Machine Learning for Intelligent IoT Networks with Edge Computing

Xiaolan Liu

Doctor of Philosophy

School of Electronic Engineering and Computer Science

Queen Mary, University of London

June 18, 2021

Abstract

The intelligent Internet of Things (IoT) network is envisioned to be the internet of intelligent things. In this paradigm, billions of end devices with internet connectivity will provide interactive intelligence and revolutionise the current wireless communications. In the intelligent IoT networks, the unprecedented volume and variety of data is generated, making centralized cloud computing inefficient or even infeasible due to network congestion, resource-limited IoT devices, ultra-low latency applications and spectrum scarcity. Edge computing has been proposed to overcome these issues by pushing centralized communication and computation resource physically and logically closer to data providers and end users. However, compared with a cloud server, an edge server only provides finite computation and spectrum resource, making proper data processing and efficient resource allocation necessary. Machine learning techniques have been developed to solve the dynamic and complex problems and big data analysis in IoT networks. Specifically, Reinforcement Learning (RL) has been widely explored to address the dynamic decision making problems, which motivates the research on machine learning enabled computation offloading and resource management.

In this thesis, several original contributions are presented to find the solutions and address the challenges. First, efficient spectrum and power allocation are investigated for computation offloading in wireless powered IoT networks. The IoT users offload all the collected data to the central server for better data processing experience. Then a matching theory-based efficient channel allocation algorithm and a RL-based power allocation mechanism are proposed. Second, the joint optimization problem of computation offloading and resource allocation is investigated for the IoT edge computing networks via machine learning techniques. The IoT users choose to offload the intensive computation tasks to the edge server while keep simple task execution locally. In this case, a centralized user clustering algorithm is first proposed as a pre-step to group the IoT users into different clusters according to user priorities for achieving spectrum allocation. Then the joint computation offloading, computation resource and power allocation for each IoT user is formulated as an RL framework and solved by proposing a deep Q-network based computation offloading algorithm. At last, to solve the simultaneous multiuser computation offloading problem, a stochastic game is exploited to formulate the joint problem of computation offloading mechanism of multiple selfish users and resource (including spectrum, computation and radio access technologies resources) allocation into a non-cooperative multiuser computation offloading game. Therefore, a multi-agent RL framework is developed to solve the formulated game by proposing an independent learners based multi-agent Q-learning algorithm.

Declaration

I, Xiaolan Liu, confirm that the research included within this thesis is my own work or that where it has been carried out in collaboration with, or supported by others, that this is duly acknowledged below and my contribution indicated. Previously published material is also acknowledged below.

I attest that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge break any UK law, infringe any third partys copyright or other Intellectual Property Right, or contain any confidential material.

I accept that the College has the right to use plagiarism detection software to check the electronic version of the thesis.

I confirm that this thesis has not been previously submitted for the award of a degree by this or any other university.

The copyright of this thesis rests with the author and no quotation from it or information derived from it may be published without the prior written consent of the author.

Signature:

Date: June 18, 2021

Acknowledgments

Foremost, I would like to express my deepest gratitude to my primary supervisor Prof. Yue Gao for his prudent guidance, continuous support and patience during my three year PhD journey, and to my co-supervisor Dr. Yuanwei Liu for his kind help and continuous support in the final year of my PhD With their professional supervision, I could figure out research problems, deal with challenges, and then contribute a list of publications.

Secondly, I want to thank Dr. Zhijin Qin for her patient revision of my research paper. I would also like to thank my collaborators, Dr. Wenge Rong from Beihang University and Dr. Julie A McCann from Imperial College London, for their technical support and research feedback. Through our collaboration, I gained special research experience and a thorough understanding of my research works.

I would also like to express my thanks to my second supervisor, Dr. Yasir Alfadhl and my independent accessor Dr. Akram Alomainy, for their support of my PhD progression viva, and all the colleagues in antenna group and network group including Dr. Xingjian Zhang, Wei Su, Dr. Qiao Cheng, Dr. Andrea Toma, Dr. Liang Yang, Zhong Yang, Xiao Liu, Chao Shu, Zihang Song, Dr. Zixiang Ma, Xiaoshuai Zhang, Hadeel Alrubayyi, Dr. Shaker Alkaraki, Joseph Nannim Mamvong, Yihua Zhou, Tianwei Hou for their insightful comments and kind help.

I would also like to thank the committee for my viva defense: Prof. Kai-Kit Wong, and Prof. Maziar Nekovee, for their professional comments and feedback.

I also want to thank my dear friends Jiadong Yu, Haoran Qi, Yan Liu, Xiaomiao Li, Lanting Zha, Mingyang Wang, Songyan Du, Pengfei Fan, Yichen Yuan for their continuous encouragement and warm accompany, so I could stay happy and keep a positive attitude towards my PhD life.

Finally, I would like to thank my family for their support and giving me freedom to be myself, and especially express thanks to my boyfriend for his encouragement and love, I could rebuild confidence quickly every time when I met challenges and felt worried about my research.

List of Publications

Journal Papers

1. **X. Liu**, Y. Gao and F. Hu, "Optimal Time Scheduling Scheme for Wireless Powered Ambient Backscatter Communication in IoT Network," *IEEE Internet of Things Journal* 6 (2), 2264-2272, Dec. 2018.

2. **X. Liu**, Z. Qin, Y. Gao and J. A. McCann, "Resource Allocation in Wireless Powered IoT Networks" *IEEE Internet of Things* 6(3) 4935 - 4945, Jan. 2019.

3. **X. Liu**, J. Yu, J. Wang and Y. Gao, "Resource Allocation with Edge Computing in IoT Networks via Machine Learning" *IEEE Internet of Things* 7 (4) 3415 - 3426, Apr. 2020.

4. **X. Liu**, J. Yu, H. Qi, J. Yang, W. Rong, X. Zhang and Y. Gao, "Learning to Predict the Mobility of Users in Mobile MmWave Networks" *IEEE Wireless communications magazine* 27 (1) 124 - 131, Feb. 2020.

5. J. Yu, **X. Liu**, Y. Gao and X. Shen, "3D Channel Tracking for UAV-Satellite Communications in Space- Air-GroundIntegrated Network" *IEEE Journal on Selected Areas in Communications* PP(99):1-1, Jul. 2020.

6. **X. Liu**, J. Yu and Y. Gao, "Multi-agent Reinforcement Learning for Resource Allocation in IoT networks with Edge Computing" *China Communications*, 17(9): 220-236, 2020.

7. J. Yu, **X. Liu**, H. Qi and Y. Gao, "Long-term Channel Statistic Estimation for Highly-Mobile Hybrid MmWave Multi-User MIMO Systems" *IEEE Transactions on Vehicular Technology* 69(12):14277-14289, Dec. 2020.

8. J. Yu, **X. Liu**, Y. Gao and X. Shen "3D On and Off-Grid Dynamic Channel Tracking for Multiple UAVs Space-Air Communications" *IEEE Transactions on Wireless Communications* (Under Review)

Conference Papers

1. **X. Liu**, Z. Qin and Y. Gao, "Resource Allocation for Edge Computing in IoT Networks via Reinforcement Learning," *IEEE Conference on Communications (ICC)*, Shanghai, China May, 2019.

2. **X. Liu**, Y. Gao, "Reinforcement Learning Approaches for IoT Networks with Energy Harvesting ," *IEEE/CIC International Conference on Communications in China (ICCC)*, Changchun, China Aug, 2019.

3. J. Yu, **X. Liu**, H. Qi, W. Zhang and Y. Gao, "Spatial Channel Covariance Estimation for Hybrid mmWave Multi-User MIMO Systems" *IEEE Global Communications Conference (GLOBECOM)*, Hawaii, USA Dec, 2019.

Table of Contents

List of Figures

List of Tables

List of Notations

U	the number of users
$\mathcal{U}, \mathcal{L}$	user set, channel set
l_m, u_i, B_m	channel m, user i, the bandwidth of channel l_m
F, k, K	time frame, timeslot index, maximum timeslot index
$\mathcal{K}_h, \mathcal{K}_d$	the set of timeslot for energy harvesting and data transmission
$P_{m,i}^{th,k}$	required power threshold
$P_{m,i}^{A,k}, P_{max}^{A}$	the available power, the capacity of the battery
$P_{m,i}^{H,k}, P_{m,i}^{k}$	the harvested power, the transmit power
$h_{m,i}^{k}, g_{m,i}^{k}$	channel gain, small-scale fading factor
$d_{m,i}$	distance between user and gatewa
η_m, a	path loss related constant, path loss exponent
$\alpha_{m,i}$	indicate if l_m is assigned to u_i
y_m^k, σ_m^2	the received signal power at gateway, , noise power
$I_m, \gamma_{m,i}^{k}$	the number of users assigned to l_m, the uplink SINR
$R_{m,i}, R_{u_i}^{uti}, R_{l_m}^{uti}$	the transmission rate, the utility function of u_i and l_m
$\Phi, \mathcal{S}_{\mathcal{U}}$	a many-to-one matching, user set in the matching game
$s_i^k, a_i^k, r_i^k, \pi^k$	the system state, action state, reward, policy
$\mathbf{P}$	transition probability matrix
$C_{i,j}^{L}, C_{i,j}^{E}$	cost of task execution in local computing, edge computing
δ	penalty function of failed task execution

t_i^k $T_{i,j}$	task queue state, the generated j^{th} at user u_i
e_i^L, e_i^E	energy consumption per CPU cycle of user, server
$\mathcal{I}^k$, I^k	task generation set, indicator
f_i, f	computation capacity of user, server
$d_{i,j}$	task size
ρ_i^k	the remaining computation capacity state
$\mathcal{O}^k$, O	users' decisions set, indicator
$D_{i,j}^L$,$D_{i,j}^E$,$D_{i,j}^T$	delay of local, edge task execution, task transmission
$E_{i,j}^L$,$E_{i,j}^E$	energy consumption of task execution in local, edge computing
β	weight factor
R_i, h_i^k, σ^2	transmission rate, channel gain and noise power
$\mathcal{P}_i^T$	transmit power set
$D_{i,j}^{th}$	task execution latency requirement of j^{th} task
x_i	user priority
h, H	cluster number index, cluster number
$\mathcal{C}_h$, c_h	cluster h, cluster centroid of $\mathcal{C}_h$
SSE, SC_i	performance index of elbow method, Silhouette Coefficient of user u_i
$\mathcal{G}_h$	channel gain set of $\mathcal{C}_h$
d_i, $\mathbf{P}_i$	distance from gateway to user, task offloading probability of user
$\delta_z^{m,i}$	indicates if u_z takes up the same channel or not
K_s	duration of the time slot
ω, ϖ	waiting cost of the user, penalty of failed offloading
$W_{all}, \bar{W}$	the computation requirements, capacity of the server

List of Abbreviations

ADR	Adaptive Data Rates
AI	Artificial Intelligence
AWGN	Additive White Gaussian Noise
AP	Access Point
AR	Augmented Reality
BP	Blocking Pair
BS	Base Station
CSI	Channel State Information
CSS	Chirp Spread Spectrum
CPU	Central Processing Unit
CC	Cloud Center
dB	deciBel
DQN	Deep Q-Network
DL	Deep Learning
DNN	Deep Neural Network
DRL	Deep Reinforcement Learning
DP	Dynamic Programming
ECAA	Efficient Channel Allocation Algorithm
IL-based MA-Q	Independent Learners based Multi-Agent Q-learning
IoT	Internet of Things

LoRa	Long Range
LoRaWANs	Long Range Wireless Area Networks
LPWA	Low-Power Wide-Area
LTE-advanced	Long Term Evolution Advanced
MINLP	Mixed Integer Nonlinear Program
MDP	Markov Decision Process
MARL	Multi-Agent Reinforcement Learning
MEC	Mobile Edge Computing
NE	Nash Equilibrium
NLP	Natural Language Processing
NB-IoT	Narrow Band IoT
NP-hard	Non-deterministic Polynomial-time hard
PCP	Poisson Cluster Process
QoE	Quality-of-Experience
QoS	Quality of Service
RL	Reinforcement Learning
RAT	Radio Access Technology
RF	Radio Frequency
RFH	RF Energy Harvesting
ReLUs	Rectified Linear Units
SF	Spreading Factors
SINR	Signal-Interference-to-Noise Ratio
SSE	Sum of Squared Errors
SC	Silhouette Coefficient
TD	Temporal Difference
VR	Virtual Reality
WSN	Wireless Sensor Network
5G	the fifth Generation
3GPP	3rd Generation Partnership Project

2ES	Two-Sided Exchange-Stable

Chapter 1

Introduction

Along with the fifth Generation (5G) communication being deployed around the world, the 3rd Generation Partnership Project (3GPP) Release 17 and Release 18 would deal with the key technologies and features of beyond 5G networks [1]. The beyond 5G communications will revolutionise the current wireless communication by a range of new services relying on higher capacity, e.g., with peak throughput reaching a gigabits per second and ultra low latency below 1 millisecond, by leveraging the benefits of the Internet of Things (IoT) and big data analysis. This can be achieved by adopting machine learning techniques to analyse the generated massive data and to optimize the network structure of the IoT system. Besides, an ever-growing complexity due to the massive connected devices and time sensitive applications will be overcome by the evolution of network and service infrastructures from cloud computing to distributed orchestration and management paradigms, i.e., edge or fog computing. This motivates the end users (e.g., devices, sensors) to offload their data to the edge server (e.g., IoT gateway and WiFi router) for data analysis, which allows the users to meet their key performance indicators in terms of ultra low latency and low power consumption because they are geometrically closer to the edge server. The successful uplink data transmission, known as computation offloading, depends on no-collision channels achieved by efficient channel access allocation, as well as reliable transmit power allocation especially in wireless powered IoT users.

With the current end devices becoming more computationally powerful, they are able to process some data locally with their local processors while only send the intensive computation tasks to the edge server for improving data analysis performance. This means the IoT users have to decide which task is offloaded, where to offload the task and how to offload the task, which raises the challenge of jointly optimizing the computation offloading decisions and resource allocation solutions. This joint optimization problem was studied to maximize the system gain measured by a weighted sum of task execution time and energy consumption, which is conventionally formulated as a Mixed Integer Non-Linear Program (MINLP) problem and solved by decomposing it into two separate subproblems of computation offloading and resource allocation [2]. However, along with the increasing network scale, this method faces higher complexity or even becomes infeasible. The learning methods with the ability to efficiently solve dynamic, complex and large-scale computation offloading problems are needed.

Machine learning technique with its success in Natural Language Processing (NLP), robot control and speech recognition, has also been explored to solve Channel State Information (CSI) prediction and modulation recognition in wireless networks and received good performance [3]. For instance, to solve the challenges in IoT networks, unsupervised learning, like clustering can extract structures and features of raw data and be adapted to big data analysis of the data generated by the IoT devices. Specifically, Reinforcement Learning (RL) with the ability to enable an agent to learn how to take actions to maximize a long-term cumulative reward under dynamic environment, has been explored to solve decision making problems by generally describing them as a Markov Decision Process (MDP). Moreover, RL algorithms were exploited to learn the optimal strategy of computation offloading, edge server selection under uncertain wireless environment conditions, as well as transmit power allocation with the dynamic energy harvesting process [4, 5]. Here, the classical model-free RL algorithm, Q-learning, relying on a Q-table for training is most adopted but faces the curse of dimensionality. Deep Learning (DL), such as Deep Neural Network (DNN) has been exploited for function approximation, which has been demonstrated to efficiently approximate Q-value function of Q-learning, known as Deep Q-Network (DQN) algorithm. Subsequently, a DQN-based computation offloading algorithm was demonstrated to learn the optimal computation offloading policy without having a prior knowledge of the dynamic environment. However, the computation resource of both edge server and end user is limited and precious so that it needs to be allocated efficiently. Therefore, a joint strategy of computation offloading and efficient computation resource and transmit power allocation requires revolutionary design with distributed learning.

To deal with multiuser computation offloading, where multiple users are performing offloading simultaneously and competing for the limited resource from the edge server, learning the distributed computation offloading mechanism becomes more challenging. Here, each user's computation offloading decision not only depends on the environmental dynamics but also on the decisions made by other users. [6] formulated the multiuser computation offloading process under dynamic environment as a stochastic game because the selfish users take self-interested actions to maximize their own payoffs. In this case, each user independently adjusts its offloading strategy according to its currently received payoff, and [6] demonstrated the formulated game has at least one Nash Equilibrium (NE). The stochastic game is a dynamic game with probabilistic transitions played by one or more players, which is the generalization of MDP and repeated games, also called Markov game, such that it has Markov property. Multi-Agent RL (MARL) framework, in which all the agents learn how to take actions to maximize their long-term cumulative rewards over the environment, can be used to model this multiuser computation offloading problem. Compared to the single-agent RL, the big difference resides in the fact that the agents' actions are coupled and each agent's observed reward is decided by the joint action taken by all the agents, this matches the non-cooperative stochastic game problem. Therefore, MARL algorithms attract more attention to distributively learn the

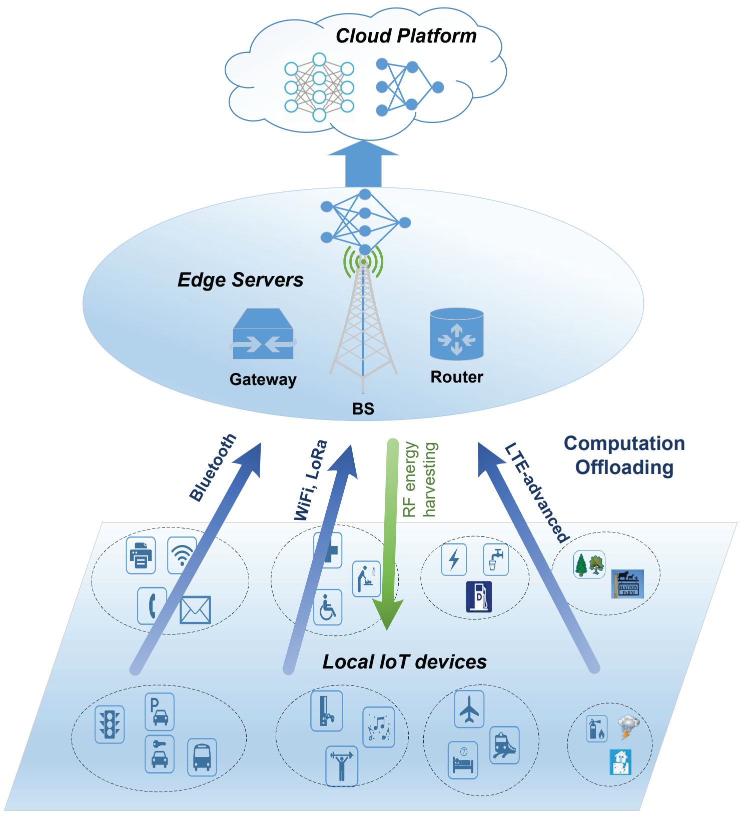


Figure 1.1: The diagram of the intelligent IoT system

optimal strategy of each user for multiuser computation offloading scenario.

1.1 Overview of IoT Networks

As shown in Fig. 1.1, an IoT system is a network of different kinds of networks. It is envisioned to be automated and intelligent, and to achieve secure and massive connectivity from end devices to the central server and until the cloud. Recently, edge computing was emerged to push the computation and communication resource physically and logically closer to users and to provide distributed data processing by introducing an extra layer, i.e., the edge layer, between the local layer and the cloud layer. Therefore, from Fig. 1.1, the IoT system has a three-layer architecture, including the cloud platform which is geometrically far from the IoT devices, edge servers which are more distributed and closer to the IoT devices. The IoT system is widely applied to smart home, smart agriculture, smart city, smart transportation and etc. All the applications have high requirements on low-power transmission due to the energy-limited batteries of the IoT devices [7]. Recently, there have been several main schemes to help form the IoT networks: radio frequency identification, Wireless Sensor Network (WSN), mobile networks and etc. Since the advance in low power circuit and wireless communication, WSN has attracted more attention due to its large-scale network structure which can sense and analyze a large amount of data [8], while mobile networks have the advantage to connect the IoT network with the core internet. As shown in Fig. 1.1, the available communication technologies for IoT networks include Low Power Wide Area (LPWA) technologies such as Long Range (LoRa) [9], Sigfox [10] and Narrow Band-IoT (NB-IoT) [11], Bluetooth and WiFi, as well as Long Term Evolution advanced (LTE-advanced). Based on these technologies, the IoT system is operating on different frequencies, such as LPWA exploring around sub-1 GHz, LTE-advanced operating on 690-960 MHz and 1710-2690 MHz.

1.1.1 Enabled Technologies

1.1.1.1 Low Power Communication Technologies

A relatively new class of communication protocols, described as LPWA, are coming to the fore as they consume lower power at the IoT device while their transmission scope covers geographically larger areas. Moreover, LPWA offers a trade-off among data transmission rate, network coverage and power consumption, to meet more various requirements on IoT applications [12–16]. LPWA systems achieve this balance of power and distance at the cost of low data transmission rate which makes it more appropriate for latency-tolerant IoT applications with smaller data requirements. The existing LPWA technologies contain LoRa, NB-IoT and Sigfox. Specially, LoRa is considered as one of the most potential LPWA techniques, and has drawn much attention from both academia and industry [17–21]. LoRa technology achieves a big success mainly because it adopts Chirp Spread Spectrum (CSS) modulation. The CSS technology contributes to flexible long range communications with low power consumption by using different Spreading

Factors (SF) [22]. Specifically, in LoRa Wide Area Networks (LoRaWANs), the gateway server configures different SFs for users and there exists a mechanism to allow Adaptive Data Rates (ADR). The server adjusts the SF and increases (reduces) the transmit power of the users according to its received Signal-Interference-to-Noise Ratio (SINR) to optimize different metrics of the network including data transmission rate, airtime, and power consumption.

1.1.1.2 RF Energy Harvesting

Since the devices in IoT networks aim to achieve self-sustainable and long-term communications but have energy-limited batteries, energy harvesting was emerged as a promising method for the proactive energy replenishment of the next generation wireless networks [23, 24]. Specially, RF energy Harvesting (RFH) draws much attention because it considers readily available transmitted energy, such as TV (radio) broadcasters, mobile BSs and handheld radios as important energy resources, which is low cost and small form factor implementation.

These ambient RF sources can be classified into static and dynamic resources. The TV signals are static RF resources since the power of the TV signal is relatively stable over time. They are promising energy resources with high transmission power [25]. In [26], the researchers designed a novel broadband Yagi-Uda antenna to harvest ambient RF power from digital TV broadcasting signals. A RFH wireless sensor network prototype was designed by harvesting signals from TV tower in [27]. Although static RF sources can provide predictable RF energy, there could be long-term and short-term fluctuations due to service schedule [28]. Moreover, TV signals are broadcasted without interruption at all the hours of day and night by the TV towers. And TV towers can transmit up to 1 milliwatt effective radiated power and can serve locations more than 100 miles away from the tower in very flat terrain and up to 45 miles in denser terrain.

Dynamic ambient RF resources (e.g., WiFi router) are produced by RF transmitters that work periodically or change transmit power over time. Therefore, making use of ambient RF resources has to be adaptive and possibly intelligent to search for available opportunities in a certain frequency range. As shown in Fig. 1.1, the edge servers can provide wireless power transfer for the IoT devices through the RF signals. Hence, the IoT devices can harvest energy from the RF signals, and adopt store-then-transmit protocol to offload the sensing data or perform local data processing.

1.1.2 Resource Management Issues in IoT Networks

1.1.2.1 Spectrum Resource

According to Ericsson mobility report, there will be 24.9 billions IoT connections by 2025 [29], which means the number of devices served by a central server will become huge. When the massive devices try to access the wireless channels simultaneously, it overloads the channel (e.g., physical random access channel in LTE networks). From study [30], to support the massive IoT connections for an exclusive spectrum use, it needs around 76 GigaHertz (GHz) spectrum resource, which is impossible to be satisfied. Therefore, efficient spectrum resource management is important due to the scarcity of spectrum resource. A few potential solutions are identified: 1) spectrum sharing [31], including licensed cellular spectrum (e.g., enhanced machine type communication and NB-IoT) and unlicensed spectrum (e.g., Bluetooth and Zigbee, LoRaWAN and SigFox), as well as exploring more available spectrum resource (e.g., millimeter wave), 2) edge computing, relying on re-designing the architecture of the network with edge devices providing distributed channel access and also reducing network congestion, as well as efficient radio resource access providing alternative spectrum choices and efficient channel allocation management reducing the network congestion probability.

1.1.2.2 Energy Resource

Since most of IoT devices are powered by energy-limited battery and charging facilities, efficient energy resource allocation is essential to provide satisfactory Quality of Service (QoS) in IoT networks. Energy harvesting provides a promising solution to power those IoT devices continuously without the need of changing the batteries frequently. However, the dynamic energy arrival process brings new challenges for energy allocation. Besides, the interference problems among users are crucial when they are trying to access to the limited channels, and coupled with the time changing physical channel and network conditions, the sophisticated dynamic transmit power allocation is required and full of challenges.

1.1.2.3 Computation Resource

With the trend of compute-intensive applications, like Apple Siri, Google Microsofts Cortana [32], the IoT devices cannot support the complicated models with their limited computation and energy resource, so the applications have to rely on cloud computing which is generally assumed to have infinite resource. However, when billions of IoT devices generate or collect data and then send to the centralized cloud center, it causes serious network congestion and unacceptable latency for IoT applications. Since the massive generated data is distributed at the network edge, edge computing is emerged with many edge servers providing distributed data processing. Compared to cloud computing, it has finite resource to support limited IoT devices requesting for computation resource simultaneously, so efficient computation resource allocation is needed for edge server. On the other hand, the increasingly powerful IoT devices are developed to have larger but still limited computation capacity, so the coordinative computation resource management between edge server and IoT devices is important.

1.2 Literature Review

1.2.1 Matching Theory for Resource Allocation

In wireless networks, the rational and selfish users are interacting with a natural propensity to solicit their maximum benefit from the network without caring about other users. To model such competitive behaviors of those players in the networks, matching theory [33] is particularly effective in developing high performance, low complexity, decentralized, and practical solutions, which can overcome some limitations of game theory and optimization. Recently, matching theory has been extensively studied to handle wireless resource allocation problems in wireless networks, such as in cognitive radio networks [34], heterogeneous cellular networks [35] and device-to-device (D2D) networks [36].

The researchers in [34, 36, 37] studied applying matching theory in wireless communications by modelling the resource allocation problems as a one-to-one marriage problem or a many-to-one matching problem. The authors in [34] studied stable spectrum allocation based on one-to-one matching framework by matching users and channels in cognitive networks, the authors proved the uniqueness of the stable matching. In [36], a context-aware resource allocation problem in wireless networks is formulated as a one-to-one matching game, which was proven to converge to a two-sided stable matching between uses and resource blocks. In [37], the authors extended the matching theory of many-to-one matching to solve the resource allocation problem in wireless communications, based on [38] that first studied stable matchings by proposing the deferred acceptance algorithm, they first developed a general framework to find stable matchings of users and resources.

In wireless networks, due to the limited resources including time, spectrum and space, resource allocation problems are extensively studied [39]. In [40, 41], matching theory was exploited to address resource allocation problems in wireless networks by assuming the users and resources as the matching players. However, the peer effects caused by the interference from other users due to channel reuse or non-orthogonal resources have not been considered. Hence, [36] implemented matching theory with peer effects to address this problem. The authors discussed a one-to-one matching model with peer effects to match users with Resource Blocks (RBs) in D2D communications.

In IoT networks, a variety of users collect data from the environment and then transmit the data to the gateway for data processing, which poses the challenges of multiple users competing for the limited wireless channels. Here, each user is taking up one channel at a time while each channel can be accessed by many users using code domain multiplexing. However, each user can be affected by the other users using the same channel (e.g., the imperfect SF orthogonality in LoRa [42] and imperfect orthogonal multiplexing with Code Division Multiple Access (CDMA) [43]). Therefore, matching theory with peer effects will be a potential way to address those challenges.

1.2.2 Dynamic Power Allocation

With the appearance of more advanced applications at IoT devices, e.g., navigation, face recognition and interactive online gaming, more embedded sensors and on-device cameras are deployed so that more powerful processors and sustainable resource supply are required [44]. However, currently resource-limited devices cannot provide satisfactory Quality of Experience (QoE) for those applications, so computation offloading was emerged to overcome this issue by migrating computation to more resourceful cloud platform or edge server [45]. Unfortunately, the conventional battery-powered devices might lead to compromised computation performance because of insufficient battery energy for computation offloading. [46, 47] explored energy harvesting technologies to continuously power mobile devices and achieved satisfactory computation performance.

Energy harvesting techniques have been proposed in IoT networks where the IoT devices can harvest energy from the RF signals and use it to transmit the collected data [48–51]. [49] has proposed a harvest-then-transmit protocol, in which a hybrid Access Point (AP) provides the users with wireless energy in the downlink transmission and then users transmit their data in the uplink by using the harvested energy before. Here, the sum of network throughput maximization problem was studied by exploring the optimal time allocation for the communication links and data transmission. In [50], the max-min problem of network throughput has been discussed, in which users harvest energy from a multi-antenna AP and transmit their data to the same AP in the uplink. The downlink-uplink time allocation, downlink energy beamforming and uplink transmit power were jointly optimized using the non-negative matrix theory. In [51], the sum throughput rate maximization problem was analyzed where mobile stations with energy harvesting capabilities communicate with a full duplex multi-antenna AP. A joint optimization problem has been formulated with optimizing channel assignment, time allocation and transmit power allocation simultaneously.

In [52], an online resource allocation algorithm has been proposed, which adopted MDP to model the stochastic environment. It considered a system which contained a single source node and multiple destination nodes, then MDP was used to model the harvested energy and channel gain guaranteeing the QoS of users. To address the throughput maximization problem, [53] has introduced a first-order MDP to model the stochastic energy arrival, and then an optimal policy of transmit power allocation was obtained by using DP method. An expected data transmission maximization problem has been investigated in [54] with the constraints of energy harvesting rate, available battery energy, buffer queue and channel gain. An infinite-horizon discounted MDP was proposed to formulate this problem and it was solved by an optimal energy allocation algorithm. In [55], online learning algorithms were adopted to exploit the MDP problem in which the rewards expectation and the state-action pairs were unknown, and then the algorithms were used to learn these rewards and develop their own policies over time. However, the dynamic power allocation while harvesting energy to achieve sustainable power supply for data transmission has not yet been addressed using both DP method and RL algorithms.

1.2.3 Computation Offloading

Edge computing has been proposed as promising solutions to handle the large volume of security-critical and time-sensitive data processing distributively [56, 57]. Compared to using distant and centralized cloud resources, edge computing employs decentralized resources which are typically limited, heterogeneous and dynamic, thereby making resource management in wireless networks more challenging that needs more effective and efficient methods [58, 59]. The IoT devices are resource-constrained, for instance, the battery capacity and local CPU computation capacity are limited [58]. Offloading computation tasks to relatively resource-rich edge computing devices can meet the QoS requirements of IoT applications and can augment the capabilities of IoT devices for running resource-demanding applications [60].

Computation offloading scheme has first been investigated to access to multiple resources efficiently in Mobile Edge Computing (MEC) networks [61–63]. Energy consumption and task execution latency are two main considerations when designing the optimal offloading scheme. The authors in [46, 64] formulated the computation offloading problem with the objective of minimizing the execution delay by proposing one-dimensional search algorithm to find the optimal offloading decision policy [64], and by proposing Lyapunov optimization-based dynamic computation offloading algorithm to jointly make the decisions on offloading strategy, the CPU-cycle frequencies for mobile execution, and the transmit power [46]. In [45, 65], the joint computation offloading and resource allocation problem with minimizing the energy consumption of the mobile device have been investigated. The online learning and pre-calculated offline solutions were proposed to solve the optimization problem [45]. And two offline solutions, deterministic and randomized, were proposed to find the optimal radio scheduling offloading policy [65].

In the aforementioned studies, the users offload all the collected raw data to the edge server for processing, which requires large bandwidth and consumes much transmit power. In [66], the authors presented that the applications could be achieved by dividing the computation task into a non-offloadable part and N offloadable parts, and then processing them locally or offloading to the edge server. The computation offloading problem is formulated as a $0-1$ programming model, where 0 represents the offloading decision and 1 indicates local computation at the user. For instance, it has been discussed in the MEC networks, in which the mobile user makes a binary decision to either offload the computation tasks to the edge device or not [67]. Moreover, an optimized trade-off between energy consumption and execution time delay was analyzed with joint allocation of radio and computation resources for partial offloading decision using the multiple input multiple output communication models [68]. Based on [68], the authors gave more in-depth theoretical analysis on a trade-off between energy consumption and execution delay in [69]. The authors in [70] formulated a power consumption minimization problem with task buffer stability constraints to investigate the power-delay trade-off.

Multiuser computation offloading is a practical problem in wireless networks and needs to be addressed properly. The authors in [61, 65, 71] extended their researches from the single-user computation offloading to the multiuser scenario. In [72], the authors provided the solutions to select the users for scheduling, hence offloading, and make decisions for the other users to either locally process or stay idle. In [62], the authors jointly optimized the radio and computation resources for multiuser MEC system. In [73], the authors jointly optimized the offloading decisions of all the users, the allocation of computation and communication resources for the multiuser multitask offloading problem. Moreover, they also studied the multiuser computation offloading problem in a multichannel wireless environment by considering a trade-off between the energy consumption and the execution delay in [74]. The authors in [75, 76] considered the computation offloading problem in the multiuser scenario based on the time division multiple access system, the computation tasks of the users were separated into small parts such that they might be able to offload one part in each time slot by considering their channel conditions, local energy consumption and the fairness among users. The authors in [77] also addressed the trade-off between execution delay and energy consumption in multiuser computation offloading scenario.

1.2.4 RL for Computation Offloading

Recently, the increase of computation capacity at edge devices contributes to a new research area, called edge learning, which crosses and revolutionizes two disciplines: wireless communication and machine learning [78–80]. Machine learning with its achievements in many disciplines provides promising solutions to solve the complicated and dynamic problems in wireless networks by exploiting different machine learning algorithms. To address these issues, RL techniques provide effective solutions in an intelligent manner to design joint computation offloading strategy and resource allocation method [81–85], which basically can be modeled as an MDP. Then it could be solved by a classical single-agent Q-learning algorithm, and to break the curse of high dimensionality, DL is used to approximate the Q-value function in Q-learning. Therefore, DRL was widely adopted to learn the optimal policy by solving the formulated computation offloading problem in MEC systems.

For example, DRL has been exploited to optimize resource allocation and computation task offloading schemes in MEC networks [5, 84–91]. The authors in [5, 84, 86] jointly optimized computation offloading, computation resource and transmit power allocation based on DRL algorithms by considering wireless charging users. In [87], the optimal task offloading policy with computation and communication resource allocation by the proposed intelligent resource allocation framework based on a multitask DRL algorithm is explored, while in [88], DRL is exploited to jointly manage the spectrum, computation and storage resources to support delay-sensitive applications in the MEC-based vehicular network. In [90], a joint optimization problem of task offloading decisions and transmission time allocation for multiuser scenario is solved in wireless powered mobile edge computing networks by the proposed DRL-based algorithm. And a DRL-based computational resource allocation policy has been addressed for edge computing network with multiple users in [91].

However, when many users in the IoT networks have collected data and require data processing, the heterogeneity of computation resource, channel gain and intensive computation tasks at users makes them have distinct offloading decisions. In [75], an offloading priority function which defined the priorities for users according to their channel gains and local computing energy consumption was derived, and the optimal policy was proved to have a threshold-based structure with respect to this defined function. This work means that the users could be classified into different groups with different priorities above or below a given threshold and perform complete and minimum offloading, respectively. It enlightened that user clustering could be performed as a pre-step to make the computation offloading decisions for a variety of users in IoT networks. Clustering algorithms, such as K-means [92], known to cluster data set into different clusters according to the characteristics of each data point, make the way for user clustering.

1.2.5 MARL for Computation Offloading

With the increasing number of IoT devices in IoT networks, the joint problem of computation offloading and resource allocation becomes challenging since it's more trendy that multiple users are simultaneously competing for the limited resources from the IoT system. RL techniques have been exploited to effectively develop computation offloading scheme with efficient resource allocation solutions in single user scenario [5, 84, 86]. Moreover, the authors in [93, 94] investigated resource allocation in IoT edge computing networks, and proposed computation offloading algorithm based on Q-learning and DQN to obtain the optimal computation offloading strategy while considering single user scenarios. However, when multiple users are trying to offload their tasks to the edge server together, each user's decisions on offloading strategy and resource management is affected by other users' decisions.

The multiuser computation offloading problem in uncertain environment has been investigated with considering power allocation, radio and spectrum resource allocation as well as computation resource allocation in [46, 76, 95–103]. The authors in [46, 95] investigated the trade-off between the power consumption of mobile devices and the execution delay of computation tasks in multiuser MEC systems, an online algorithm based on Lyapunov optimization was proposed to obtain the computation offloading strategy. Generally, the joint multiuser computation offloading and resource allocation is formulated as a non-convex optimization problem, and then different methods were tried to solve this problem, like alternating direction method of multipliers decomposition technique [96], iterative heuristic resource allocation [97] and converting to convex optimization problem [76]. Those algorithms are operated relying on the centralized controller, which will cause high system implementation complexity when the network size grows. Hence, the distributed algorithm is preferred and necessary. In [98–100], the authors have presented distributed computation offloading algorithms to investigate the multiuser computation offloading problem. A distributed greedy maximal scheduling algorithm was proposed to solve the multitask offloading schedule problem in multiuser scenario, while requiring all the mobile devices to report their true energy link weights to the wireless devices [98]. In [99], the authors formulated the multiple users' decision making problem of whether to offload or not as a prospect theory based on non-cooperative game, and proposed a distributed computation offloading algorithm to reach the NE of the game, with the need of edge server summarizing each user's threshold value and users making decisions one by one. In [100], a two-tier game-theoretic greedy offloading scheme was proposed to study the computation offloading problem in ultra dense IoT networks, while sharing messages with other devices. Besides, [102, 103] explored non-orthogonal multiple access-enabled multiuser computation offloading scheme such that the users were able to simultaneously offload their tasks to the BS over the same time/frequency resources.

Moreover, the joint optimization problem of computation offloading and resource allocation becomes aggravated when considering the randomness and dynamics of wireless networks, such as user mobility, uncertain channel quality, random task-arrival and energy resources (if considering dynamic energy harvesting). By invoking Lyapunov optimization, the authors in [46, 95, 104] studied dynamic computation offloading strategy and resource allocation, to investigate power-delay trade-off under time-varying wireless environment. Even though Lyapunov optimization was used to solve these problems, it needs to decompose the optimization problem into different suboptimal problems and solve them separately, besides that, it ignores the long-term consequences of the current decisions. To address these challenges, the authors in [6, 74, 105] studied distributed multiuser computation offloading problem by formulating it as a computation offloading game using game theory, and then to solve the formulated game problem, iterative algorithms were proposed to reach a NE. Moreover, a distributed scheme for each Cloud

Center (CC) to determine its data offloading and resource allocation strategy independently based on MARL was proposed [106], where each CC is self-motivated to learn the explicit models of other CCs.

MARL is focused on formulating models of multiple agents learning optimal strategies while interacting with the dynamic environment. Compared to a single agent scenario that the environment changes its state only based on the action of one agent, the new environment state depends on the joint action of all the agents in multi-agent scenario. The MARL has a few benefits: 1) Agents learn their strategies in distributed manners with observing local environment information; 2) Agents can share experience by communicating with each other; 3) MARL is more robust if some agents fail to operate in the multi-agent system, the rest agents can take over some of their tasks. The MARL was popularly adopted to perform resource allocation in different wireless communication networks [107, 108]. In [107], an MARL approach was proposed to explore stochastic power adaption in cognitive wireless networks. A multi-agent dynamic resource allocation algorithm based on MARL has been proposed for multi-UAV downlink networks to jointly design user, power level and sub-channel selection strategies [108].

1.3 Motivations and Contributions

Along with edge computing envisioned intelligent IoT networks, my PhD spans machine learning based computation offloading and resource management for intelligent IoT networks with particular emphasis on RL techniques. The proposed algorithms are capable of improving the efficiency of resource allocation with low complexity at IoT users and achieving distributed learning. The detailed motivations and contributions of my PhD research works are summarized as following.

1.3.1 Computation Offloading in Wireless Powered IoT Networks

To extract information from the collected data, the resource-constrained IoT devices choose to offload all the collected data to the edge server for data processing, where the number of IoT devices is always more than the access channels owned by the edge server so that multiple IoT users have to compete for one access channel. Moreover, the IoT devices are wireless powered by the RF signals to obtain continuous power supply, but the dynamic energy arrival process brings another challenge for transmit power allocation of the IoT users to offload their data. Therefore, how to allocate the access channels to IoT users is important and the dynamic transmit power allocation along with the dynamic energy arrival and environmental dynamics is needed.

Motivated by this, computation offloading in the wireless powered IoT networks is formulated into a joint optimization problem of channel allocation and dynamic power allocation. To make the formulated non-convex problem traceable, it is decomposed into two phases including assigning the IoT users with the available channels and optimizing transmit power allocation of the IoT users for its computation offloading to get solved separately. In the first phase, an ECAA algorithm is proposed to assign access channels to users, which is achieved by enabling the users to self-match with the available channels based on matching theory. A many-to-one game is adopted to solve this channel allocation problem, where each channel can be accessed by more than one user while each user can only match with no more than one channel at a time. In the second phase, the RL-based power allocation mechanism is proposed to learn the optimal transmit power strategy for each IoT user since the dynamic power allocation process under the environmental dynamics can be modeled as an MDP. Hence, both DP method and Q-learning algorithm are exploited to learn the transmit power allocation strategy for the IoT user.

1.3.2 Computation Offloading in IoT Networks via Machine Learning

With the increasing of more powerful IoT devices, they can process some simple tasks locally with their own processors and transmit the intensive computation tasks to the edge server for better processing. In IoT networks, the heterogeneity of computation resource, channel gain and intensive computation tasks at users makes them have distinct offloading decisions. Since the edge sever has limited computation and communication resource compared to the conventional cloud center, each IoT device has to make proper computation offloading decisions by accounting for its remaining computation capacity and the competition for resources from the edge server.

Inspired by the success of machine learning in NLP, gaming and robot control etc., machine learning techniques were also explored to deal with CSI prediction in wireless networks and received good performance. Hence, this joint optimization problem of computation offloading and resource allocation in the IoT networks is explored via machine learning approaches. A centralized K-means based clustering algorithm is first proposed to pre-process the computation offloading decisions of all the users, which is achieved by grouping the users into different clusters according to user priorities. The clusters with the highest and lowest user priority can be directly assigned as offloading cluster and local computing cluster, respectively. For the other clusters, the users belonging them then need more accurate computation offloading strategies and resource allocation solutions. To address this, the decision making process of each user under dynamic task arrival, its remaining computation capacity and environmental dynamics can be modeled as an MDP. Therefore, a DQN-based computation offloading algorithm is proposed to learn the computation offloading strategy and resource allocation solutions with the objective to minimize the long-term system cost.

1.3.3 MARL for Multiuser Computation Offloading in IoT Networks

Along with exploring resource management schemes and computation offloading strategies in IoT networks, both centralized algorithm and distributed algorithm are needed to address this joint problem but only works by separating resource allocation of the edge server. Another challenge of multiuser making computation offloading decisions simultaneously and competing for limited resources (e.g., spectrum and computation resource) from the edge server needs to be paid more attention. In this case, except for the environmental dynamics, the decision making process of each user is also influenced by the other users in the considered IoT network and their resource requests are coupled at the edge server.

In order to investigate this multiuser computation offloading problem, my research formulates it as a non-cooperative stochastic game where multiple selfish users are making their own computation offloading decisions simultaneously to obtain the payoffs. Then an MARL framework is exploited to provide intelligent computation offloading and resource management solutions for users with distributed learning, where each user is considered as a learning agent to learn its computation offloading strategy distributively by observing its local environment information without exchanging experience with other users. Specifically, the observed reward of the user is defined by aggregating the action effect of all the other users. Hence, an Independent Learners based Multi-Agent Q learning (IL-based MA-Q) algorithm is proposed for each user to independently run a Q-table to learn the optimal computation offloading strategy. Besides, an iterative algorithm is proposed to obtain the computation offloading strategy by learning the idea of the algorithm proposed in [6].

1.4 Thesis Outline

Chapter 2 provides an overview of the intelligent IoT networks, mainly focusing on basic mathematical techniques including matching theory and machine learning, and the basic knowledge of computation offloading schemes.

Chapter 3 investigates spectrum and power allocation for computation offloading in wireless powered IoT networks. Here, an ECAA based on matching theory is proposed to provide effective channel access schemes for IoT users. And then RL-based power allocation mechanism is explored to learn the optimal transmit power strategy for the IoT user under dynamic energy arrival and uncertain wireless communication environment.

Chapter 4 addresses the joint computation offloading and resource management problem in IoT networks, where the IoT users make decisions on computation offloading to offload the intensive computation tasks to the edge server while the rest simple tasks can be processed locally at the IoT user. Here, the centralized K-means based clustering algorithm is proposed to pre-process the computation offloading decisions of all the IoT users by grouping them into different clusters according to user priorities. Then the decision making process of each IoT user is considered in terms of the computation capacity of the IoT user and the environmental dynamics, which is described as an MDP so that a DQN-based computation offloading algorithm is proposed as the solution.

Chapter 5 focuses on the challenges of simultaneous multiuser computation offloading because the IoT users are competing for communication and computation resource in IoT networks, and formulates the simultaneous decision making process of multiple users as a non-cooperative stochastic game with each user observing its local information to minimize its long-term system cost. Then an MARL framework is presented to solve this stochastic game and an IL-based MA-Q computation offloading algorithm is proposed for each user to learn its optimal computation offloading strategy and resource allocation solutions with its reward function defined by aggregating the action effect of all the other users.

Chapter 6 draws the conclusions of this thesis and gives the possible future research directions.

Chapter 2

Background

In this chapter, the basic mathematical techniques for resource allocation in IoT networks are presented, including matching theory and machine learning techniques. Then, edge computing with the benefits of distributed data processing, less network congestion and less time delay in IoT networks are stated by introducing computation offloading schemes and machine learning for computation offloading.

2.1 Matching Theory

2.1.1 The Preliminaries

Matching theory was first proposed in economics as a mathematical framework to describe the mutually beneficial relationships of matching players in two distinct sets, depending on the individual information and preference of each player [109], which discussed the matching problems in the following three classifications, one-to-one marriage problem, many-to-one college admissions problems [38] and many-to-many firms and workers problem [110]. Matching theory can provide mathematically tractable solutions for the combinatorial problem of matching players in two distinct sets.

The classical matching problem is a stable marriage problem. In this problem, a set of women and a set of men are assumed to be the matching players. Each player in the women (men) set builds an ordered preference list based on the preferences over the men (women) set of players who she (he) finds acceptable. A matching is defined as a marriage between one woman and one man. The notion of stability is essential to the matching model, which refers to the case that no Blocking Pair (BP) exists in a matching. A BP is defined as a pair (man, woman), in which both prefer to leave their current partners and form a new marriage with each other. The Gale-Shapley algorithm was first proposed to find the stable matching of the marriage problem, which is widely deployed and has been customized to generate stable matchings in many other models.

According to the preferences of the players, matching problems can be formulated into different type of matching models. The detailed descriptions are given as follows:

- Bipartite matching problems with two-sided preferences: the participating players are divided into two distinct sets, in which the players in one set need to be matched with the players in the other set, and each player has preferences over possible matches. The example applications include labor market and new doctors in the hospital.
- Bipartite matching problems with one-sided preferences: similarly, dividing the participating players into two distinct player set, but this time only one set of players rank the subsets of the members in the other set in order of preferences. For instance, the applications include dormitory assignment and places in a public school.
- Non-Bipartite matching problems with preferences: all the participating players are forming a homogeneous set, in which any player ranks a subset of all the others in order of preferences and it can be matched with any other. This is applied in the chess tournaments and kidney exchanges.

2.1.2 Matching Theory for Wireless Resource Allocation

In wireless networks, the resource allocation problem can be formulated as a matching problem between resources (e.g., power, channels and etc.) and users (e.g., devices, BSs and etc) [111], which can be formulated into one-to-one, many-to-one or many-to-many matching model. There is a limit on the maximum number of RBs (users) can be matched for each user (RB). By given their individual objectives and learned information, e.g., transmission rate, it aims to optimally match RBs and users. Then each player (user or RB) builds a preference list in order which represents the preferable player set of the other set. Due to the uncertain and dynamic environment, applying matching theory in wireless resource management is full of challenges. According to the characteristics of the wireless environment, resource allocation problems based on matching theory can be categorized into three classes as follows:

- Canonical matching: the player (e.g., RB or user) in one player set builds its preference list only depending on the information of this player and the players that it is intending to match. This can be used to match users with the orthogonal RBs.
- Matching with externalties: here, the matching problem is formulated with "externalities", which indicates the interdependence among the players' preferences. For instance, in an IoT network, when a user is matched with one channel, the other users' preference will change since the user can cause interference to the other users using the same channel. Therefore, each user's preference depends not only on its own information but also on the other players matched to the same place. This is highlighted as "peer effects".
- Dynamic matching: in this case, the matching model with dynamics can solve the dynamic problem that adapts the matching processes to the dynamic environment, such as fast fading, mobility or time-varying traffic. Here, the preference of the players might change over time. Hence, the time dimension has to be considered

to find the matching solutions. The dynamic matching can be considered as a repeated matching game with the matching problem at each time epoch either belonging to canonical matching or matching with externalties.

2.2 Machine Learning

2.2.1 Supervised and Unsupervised Learning

Supervised learning aims to build a mathematical model by analyzing a set of labelled data that contains both the inputs and the desired outputs. With the supervised learning algorithm, the mathematical model is inferred by analyzing the known training data set in which each data sample is labeled. Hence, the learned mathematic model can be applied to predict future events on new data set. Each training data sample, known as the labeled sample, contains one or more inputs and the desired output. Generally, the data samples are presented as an array or a vector, also called feature vector, then the training data set is presented as a matrix. Through iterative optimization of an objective function, supervised learning algorithms can learn an approximate mathematic function that can be used to predict the output associated with new inputs. The classical supervised learning algorithms includes active learning, classification and regression.

In contrast, unsupervised learning aims to extract structure from a set of data that only contains inputs, such as clustering or grouping data points, here the data set is raw data and not labeled, classified or categorized. It focuses on identifying commonalities in the data and reacting based on the presence or absence of such commonalities in each new piece of data. Some common unsupervised learning algorithms are including clustering, anomaly detection, neural networks and etc. Cluster analysis is used to group or segment datasets with shared attributes in order to extrapolate algorithmic relationships.

Clustering is the task that aims to group a set of objects into different clusters, where the objects grouped into the same cluster have more similar features to each other than those objects in other clusters. Clustering represents the process of the general task to be solved, it does not denote one specific algorithm. It can be achieved by various algorithms that differ significantly in their understanding of what constitutes a cluster and how to efficiently find them. As listed above, clustering algorithms can be categorized based on their cluster model, including hierarchical clustering, centroid-based clustering (k-means clustering), distribution-based clustering and etc.

K-means clustering aims to group N observations $(x_1, x_2, ..., x_N)$ into $H \leq N$ clusters $C = \{C_1, C_2, ..., C_H\}$, in which each observation is a d-dimensional vector belonging to the cluster with the nearest mean (e.g., cluster centers or cluster centroid), serving as a prototype of the cluster. This is achieved by minimizing the within-cluster sum of squares (i.e., variance), which is to find

$$\arg\min_{C} \sum_{h=1}^{H} \sum_{x \in C_h} \|x - c_h\|^2 = \arg\min_{C} \sum_{h=1}^{H} |C_h| \operatorname{Var} C_h, \tag{2.1}$$

where c_h indicates the mean of data points in cluster C_h.

2.2.2 The Preliminaries of RL Techniques

In this section, the basic theory of RL techniques is presented. RL is a branch of machine learning algorithms, which enables agents to learn how to take actions in an interactive process with the environment so as to maximize some notion of cumulative rewards. The interactive process of agent and the dynamic environment can be described by the mathematical framework, MDP, and almost all the RL problems can be formalized using MDPs.

Specifically, a classical model-free RL algorithm, Q-learning, can deal with the stochastic problems with no need of the explicit model of the environment. Q-learning combined with function approximation makes it possible to adapt to problems with large state action space. One popular solution is to use DNN as the function approximator, known as DQN. To consider multiple agents learning, Markov games are used to formulate the multi-agent problem instead of MDPs. A significant part of the research on multi-agent learning concerns the RL techniques and raises the MARL framework.

2.2.2.1 MDP

An MDP is a stochastic control process under discrete time horizon, which provides a mathematical framework for modeling decision making problems in situations where the outcomes are partly random and partly under the control of a decision maker. It is an extension of Markov chains with the characteristics of the probability of each event depending only on the state attained in the previous event. The MDPs are appropriate to formulate the optimization control problems under dynamic environment.

An MDP problem can be described by a 4-tuple $(\mathcal{S}, \mathcal{A}, \mathbf{P}, r)$, which includes a few key parameters, $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, $\mathbf{P}(s_k, s_{k+1}) = P_r(s_{k+1} \mid s_k, a_k)$ is the probability that the agent takes action a_k in state s_k at time slot k will lead to state s_{k+1} at time slot $k+1$, and r is the immediate reward received after transition from state s_k to state s_{k+1} taking action a_k. The focus of the MDP problem is to find an optimal policy π^* that maximizes the expected long-term discounted sum of rewards given as

$$E[\sum_{k=0}^{\infty} \lambda_k r(s_k, s_{k+1})], \tag{2.2}$$

where λ is the discount factor satisfying $0 \leq \lambda \leq 1$, which is usually close to 1. The classical method used to solve the MDP problems is DP method.

2.2.2.2 DP Methods

There are two types of DP algorithms to solve the MDP problems: value iteration and policy iteration.

Value iteration: known as backward induction, iteratively approximates the optimal state value function $V^*(s)$. In each iteration k, it updates an approximation $V_k(s)$

of $V^*(s)$ by applying the following operation to each state s,

$$V_{k+1} = \max_{a \in \mathcal{A}} [r(s_k, a) + \lambda \sum_{s_k \in \mathcal{S}} \mathbf{P}(s_k, s_{k+1}) V_k(s_{k+1})]. \tag{2.3}$$

Policy iteration: it produces a sequence of policies that converge to an optimal policy. It starts with an arbitrary initial policy, π_0, and computes its value function V^{π_0}. Then the process repeats starting with π_1 until it converges. Policy iteration's computational bottleneck is the necessity to evaluate a policy at each step, which requires solving a system of $|S|$ linear equations.

It is clear that these algorithms require a complete model of the MDP problem, including the transition probabilities and the rewards. However, in practice, it is hard to obtain the complete and accurate transition probability model, the agent only knows the possible state and action set but is able to observe the current state from the environment. In this case, the agent can actively learn the optimal policy through the experience of interactions with the environment.

2.2.3 RL Algorithms

As mentioned before, if the transition probabilities and the rewards are unknown, the agent has to learn the optimal policy through successively interacting with the environment. RL algorithms are superior to solve the MDP problems without assuming the advanced knowledge of an exact mathematical model of the MDP. Moreover, the RL framework can be modelled as an MDP problem with a 4-tuple $(\mathcal{S}, \mathcal{A}, \mathbf{P}, r)$. The learning process of the agent interacting with the environment in discrete time steps is shown in Fig. 2.1. In time slot k, the agent observes a state s_k and receives a reward r_k from the environment. It then chooses an action a_k from the set of available actions, which is subsequently sent to the environment. The environment moves to a new state and generates a new reward r_{k+1} associated with the transition (s_k, a_k, s_{k+1}) is determined. There are two categories of RL algorithms: model-based learning and model-free learning.

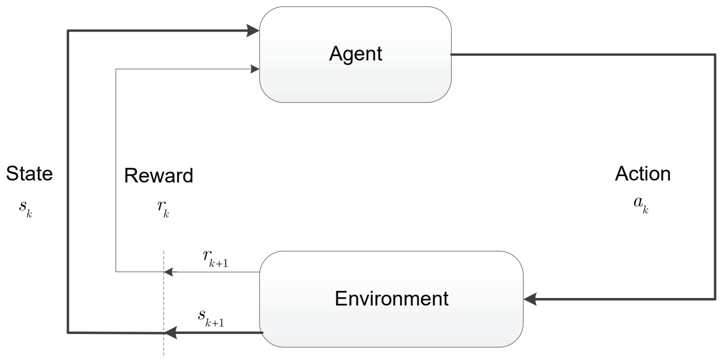


Figure 2.1: The learning process of RL framework.

Model-based Learning: the agent interacts with the environment and obtains experience from its interactions, then the agent tries to approximate the state transition and reward models. Afterwards, given the models it learnt, the agent can use value-iteration or policy-iteration to find an optimal policy.

Model-free Learning: the agent will not try to learn explicit models of the state transition and reward functions. However, it directly derives an optimal policy from the interactions with the environment.

2.2.3.1 Q-Learning Algorithm

Q-Learning is a classic model-free RL algorithm. It aims to learn a policy that tells an agent to take actions under different environmental states. In this case, the agent does not know anything about the model of the environment, and it focuses on learning what are the good or bad actions to take by using trial and error experience. The basic idea of Q-Learning is to approximate the state-action pairs, i.e., Q-function, from the samples of $Q(s, a)$ which is observed by interacting with the environment. During learning process, $Q(s, a)$ is initialized to be a possibly arbitrary fixed value. Then, at time step k, the agent takes an action a_k, observes a reward r_k from the environment, and transits to a new state $s_{(k+1)}$, so $Q(s, a)$ is updated by calculating from the received reward r_k. The core of the algorithm is a simple value iteration update, using the weighted average of

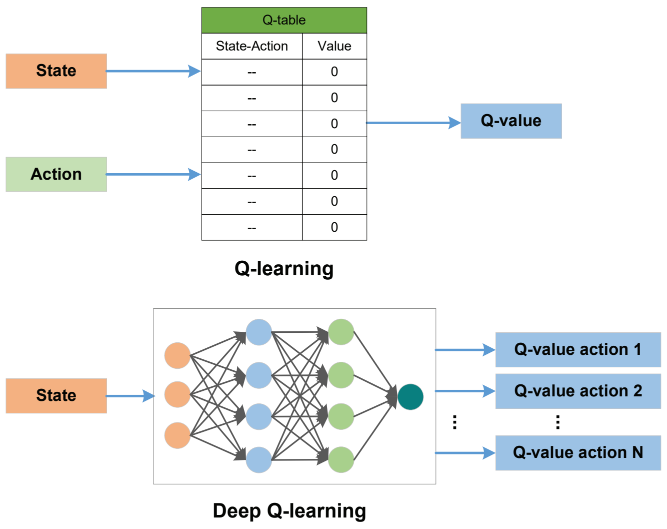


Figure 2.2: The comparisons of Q-learning and deep Q-learning algorithm.

the old value and the new information,

$$Q^{new}(s_k, a_k) \leftarrow (1-\alpha)Q(s_k, a_k) + \alpha(r_k + \lambda \max_{a} Q(s_{k+1}, a)), \tag{2.4}$$

where r_k is the reward received by the agent when moving from the state s_k to the state s_{k+1}, and α is the learning rate $(0 < \alpha \leq 1)$. Q-learning is trained depending on building a Q-table, which restricts it to adapt to the high dimensional problems with large state action space. Therefore, more efficient techniques, such as function approximation and distributed learning, are needed to address the curse of high dimensionality in Q-learning.

2.2.3.2 DRL Algorithm

DRL algorithm combines DL and RL techniques to explore efficient learning algorithms that can be applied to many practical scenarios. Specifically, deep Q-learning, defined by using DNN to approximate the Q-value function in Q-learning, is able to break the curse of high dimensionality in Q-learning. The comparison of Q-learning and deep Q-learning framework is shown in Fig. 2.2. But it is unstable with the nonlinear approximator, DNN, due to the correlations presented in the sequence of observations. Hence, the experience replay is used to remove the correlations by using a random sample of prior

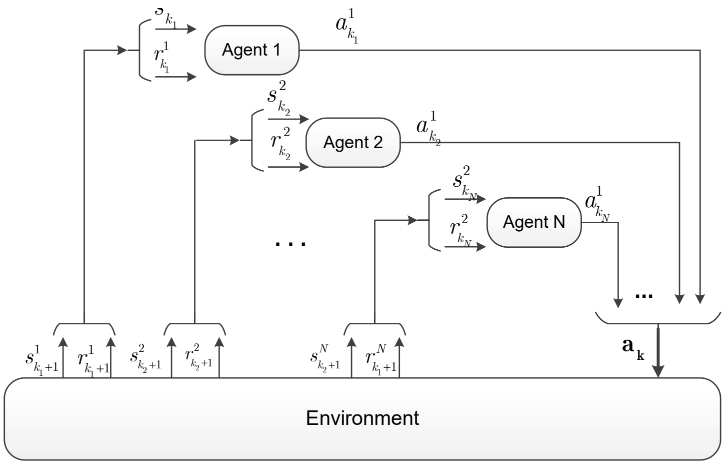


Figure 2.3: The framework of multi-agent RL framework.

actions instead of the most recent action to proceed.

2.2.3.3 MARL Algorithm

As mentioned above, an MDP is used to describe the decision making process of single-agent scenario. However, in reality, there are generally multiple users interactively making decisions together under uncertain system, this dynamic process can be described by a stochastic game, also known as Markov game, which is a general extension of single agent MDP. The RL techniques have been witnessed to be sensational in many prominent sequential decision-making problems by solving the formulated MDP problem. Therefore, to address the Markov game that including a multiuser decision making process, an MARL framework is introduced. MARL can address the sequential decision-making problem of multiple agents that operate in a common environment, each of which aims to optimize its own long-term return by interacting with the environment and other agents [112]. Compared to the single-agent scenario that the environment changes its state only based on the action of one agent, the new environment state depends on the joint action of all the agents in the multi-agent scenario as shown in Fig. 2.3. The MARL framework is exploring the multiuser decision making problem under the following three settings,

- *Cooperative setting*- a fully cooperative setting is the case that all the agents share

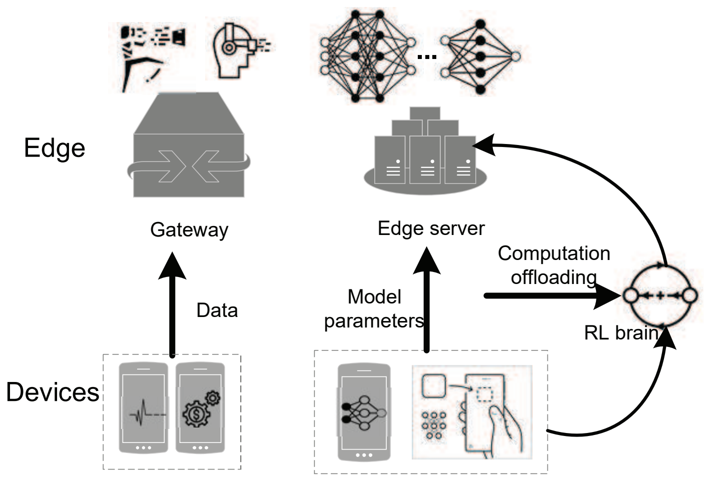


Figure 2.4: The diagram of computation offloading in IoT networks.

a common reward function, $r_1 = r_2 =, ..., = r_N = r$.

- *Competitive setting*- a fully competitive setting is typically modeled as a zero-sum Markov game, i.e., $\sum_{i \in \mathcal{N}} r_i(s, a, s\prime) = 0$ for any $(s, a, s\prime)$.
- *Mixed setting*- a mixed setting is usually modeled as a general-sum game, where no restriction is imposed on the goal and the relationship among the agents.

2.3 Edge Computing

Compared to a centralized cloud server, edge computing with its distributive characteristics can support latency-critical services and a variety of IoT applications, which requires the need of computation offloading schemes design. The diagram of computation offloading in IoT networks is presented in Fig. 2.4, the IoT devices could offload their raw data to the edge for better data processing or offload machine learning model parameters to the edge for model aggregation (e.g., federated learning). To obtain efficient computation offloading, the computation offloading schemes with efficient resource allocation need to be developed, which can be achieved by conventional optimization and machine learning techniques.

2.3.1 Computation Offloading Schemes

The increasingly intelligent applications, like autonomous driving, seamless Virtual Reality (VR) and Augment Reality (AR), surveillance and facial recognition, have high demands on computation capacity, but most of these applications are operating on resource-limited end devices, such as mobile phones are power-limited, sensors are in small sizes and have slow processors and small amount of storage. Computation offloading provides an alternative solution to enhance the end devices' capabilities by migrating computation tasks (e.g., programs) to more resourceful servers (e.g., edge servers and cloud servers) [113]. Prior to 2000, researches were mainly focused on making offloading feasible since it would take up extra bandwidth. Then time moves to early 2000s, researchers moved the focus to develop algorithms for making offloading decisions, i.e., decide whether offloading would benefit mobile users. Therefore, cloud computing with its elastic resources and multiple servers was the first enabler for computation offloading. Moreover, the development of wireless communication technology and the exploration of wideband frequency are another enablers for computation offloading by coping with the bandwidth problem. However, cloud computing imposes huge overhead on radio resource and backhaul of mobile networks, and introduces high latency since a large amount of raw data needs to be transmitted to more powerful cloud servers that are, in terms of network topology, far away from the end devices [114].

To address the problem of high time delay and network congestion, edge computing was emerged to move resourceful devices (i.e., cloud servers) closer to users, i.e., to the edge of the network. Compared to cloud computing where the cloud server is accessed by internet connection, in the case of edge computing, the edge servers that support the computation and storage resource are in proximity of the end users. Hence, edge computing can provide lower transmission delay. Moreover, cloud computing is fully centralized with the cloud server usually placed at one or few locations, edge computing with many edge servers placed at different locations is deployed in fully distributed manner. On the other hand, edge servers can only provide limited computation and storage resource with respect to cloud server. In this regard, the computation offloading brings a few challenges, such as, making decisions on computation offloading (e.g., offloading or not, when and where to offload, what to offload), accurately estimating the energy consumption, efficient resource management for simultaneous multiuser computation offloading.

To address the challenges imposed by computation offloading, computation offloading scheme has been investigated to find the right decisions, i.e., to decide whether to offload or not. Basically, a decision on computation offloading may result in different computing model.

Local execution-all the computation tasks are executed locally at the end users. So the offloading to the edge server is not performed.

Full offloading-users choose to offload all of their computation tasks.

Partial offloading-a part of computation tasks are executed locally while the others are offloaded to the edge server according to the wireless environment, the available resources of the system and the task complexity.

Furthermore, offloading computation tasks of the end users requires abundant spectrum resources or it might bring about the congestion of wireless channels [115]. Therefore, resource allocation, such as computation, power and spectrum resource allocation, is quite important for such types of resource-constrained networks. The current end users are increasingly powerful and are able to process some simple computation tasks locally, so the coordinative data processing between the end users and the edge servers needs to be addressed. This relies on the design of dynamic computation offloading scheme for the end users, i.e., the end users decide where to execute the computation tasks, at the end users or the edge server.

Computation offloading problems have been explored by exploring the offloading decisions for the users with the objective to minimize the execution delay, to minimize the energy consumption at the end user while the pre-defined delay constraint is satisfied, or to find a proper trade-off between both the energy consumption and the execution delay.

2.3.1.1 Execution Delay Minimization

Offloading computation tasks to the edge server can meet the response time and real time constraints, for instance, more complex applications are required and the processor of the end user is too slow or even impossible to support the high demand for computation capacity. Moreover, when the applications that only can be achieved by analysing multiple streams of data from different sources, computation offloading is irreplaceable. Suppose the end user has m amount of computation for offloading, and let s_l be the computation speed of the end user, so the execution time delay of those computation tasks at the end device is $\frac{m}{s_l}$.

If the computation task is offloaded to an edge server, the input data d_i from the end user u_i needs $\frac{d_i}{R}$ seconds with the transmission rate as R to be transmitted to the edge server. Let s_e be the computation speed of the edge server, so in this case, the time delay includes two parts, the transmission time delay and the execution time delay[1], is given as $\frac{d_i}{R} + \frac{m}{s_e}$.

The computation performance is improved by offloading only when this inequality is satisfied,

$$\frac{m}{s_l} > \frac{d_i}{R} + \frac{m}{s_e}, \tag{2.5}$$

when (2.5) is satisfied, the better choice for the end user is to fully offload data to the edge server.

2.3.1.2 Energy Minimization

Since most of the end users are battery-powered, and with the emergence of energy consuming applications, like VR, AR and video streaming, they are not able to support

[1]The transmission time delay of the downlink transmission from the edge server to the end device is neglected since the computing result is in a small size and the transmit power of the BS is large.

those applications on their own. Computation offloading may extend the battery life by migrating the compute-intensive computation tasks to the edge servers. Suppose P is the operating power of the end user, so the energy consumed by executing task locally is $P \times \frac{m}{s_l}$.

Let P_t be the transmit power required to transmit data from the end user to the edge server, after receiving data, the edge server processes it with the power P_e, so the total energy consumption of computation task execution in this case is $P_t \times \frac{d_i}{R} + P_e \times \frac{m}{s_e}$.

Similarly, the computation offloading saves energy only when the inequality is satisfied

$$P \times \frac{m}{s_l} > P_t \times \frac{d_i}{R} + P_e \times \frac{m}{s_e}, \tag{2.6}$$

when (2.6) is satisfied, the user chooses to offload its computation tasks to the edge server.

Moreover, the execution delay and energy minimization can be considered together as the objective function of the computation offloading problem. A weighting factor is used to balance the preference of minimizing the execution delay or minimizing the energy consumption.

2.3.2 Machine Learning for Computation Offloading

Computation offloading has been investigated as a joint optimization problem of computation offloading decision and resource allocation under the constraints of energy, execution delay and computation capacity in dynamic wireless environment. This kind of problem is generally formulated as a non-convex or an Non-deterministic Polynomial-time hard (NP-hard) problem. The traditional methods, like decomposing the problem into sub-problems and converting it to convex problem, as well as Lyapunov optimization, have been tried to solve this problem. However, they have to solve the formulated problem separately and only construct an approximately optimal solution or are infeasible for large-scale problem (e.g., massive users, fast-changing environment). Hence, the learning methods with their successful performance in the areas, like NLP and speech recognition etc. have drawn a lot of attention to be used to solve the dynamic and complex computation offloading problem.

2.3.2.1 DL for Computation Offloading

In edge computing networks, the users can generate the random and exhaustive-based optimal offloading strategies from its experience, which builds up the dataset with each data sample recorded as the network state and its corresponding optimal offloading strategy. Therefore, DL (e.g., DNN) could learn the optimal policy by training on the labelled dataset, like the supervised learning. Moreover, in the multiuser scenario, multiple parallel DNNs can be used to learn the offloading decisions for users with each user generating its own labelled dataset distributively.

2.3.2.2 RL for Computation Offloading

The design of computation offloading strategies in IoT edge computing networks should account for the wireless environmental dynamics, such as the time-varying channel quality, the task arrival and energy status of the end users, the limited computation resource of both edge server and users, as well as the limited spectrum and radio resource. Moreover, this dynamic computation offloading process is frame-based, which can be viewed as an MDP. Consequently, RL technique can enable the end user or the edge server to become an intelligent agent that learns the optimal computation offloading decisions and resource allocation solutions via a large number of trial and error interactions. Generally, the classic model-free RL algorithm, Q-learning is adopted with no need of the explicit models of the environment to solve the formulated computation offloading problem by exploring its trail and error mechanism. It's achieved by building and training the Q-table to find the optimal computation offloading strategy. However, with the increase of the network scale, like the user number and the considering resource dimensions, the large Q-table training will be inefficient or even this RL algorithm won't be able to cope with.

DNN with its ability of function approximation has been demonstrated to efficiently approximate Q-values of Q-learning algorithm, known DQN. Compared to DL-based computation offloading, DRL-based computation offloading can exploit the context of users and networks, and explore adaptive strategies to maximize the long-term system performance (e.g., network throughput, or the negative reward, system cost). Moreover, when the users want to independently take actions according to their own interests, the design of an efficient distributed multiuser computation offloading mechanism can be formulated as a Markov game. Then, the formulated multiuser computation offloading game can be solve by using the MARL framework, where all the users are considered as self-interested agents to learn their own computation offloading strategies by interacting with the environment with the goal of maximizing their individually cumulative long-term rewards.

2.3.3 Summary

This chapter presents the preliminaries of matching theory and its application in wireless networks, and machine learning techniques including RL, DRL and MARL algorithms. Moreover, the design of computation offloading schemes in IoT edge computing networks are introduced, as well as machine learning for computation offloading is presented.

Chapter 3

Computation Offloading in Wireless Powered IoT Networks

As mentioned in Chapter 2, computation offloading is a potential way to perform massive data processing on the edge (cloud) server that is believed to have sufficient computation resources. Here, the resource-limited IoT users can transmit all the collected data to the central server for data processing (i.e., full computation offloading). Therefore, the transmission process of all the IoT users has high demand on spectrum resource and the available power. The successful computation offloading (e.g., uplink data transmission) is essentially depending on efficient resource management, including spectrum, energy and computation resource management. In this chapter, the spectrum and transmit power allocation is investigated for computation offloading in wireless powered IoT networks. An ECAA based on matching theory is proposed to provide effective channel access schemes for the IoT users, which is superior to random channel assignment, and has much lower computational complexity than the brute-force exhaustive search. Moreover, the transmit power allocation is explored by both DP-based and RL-based algorithms to obtain the optimal transmit power strategy, which achieves online power allocation and has better performance than the offline scheme. The work presented in this chapter has been published as a conference paper in 2019 IEEE/CIC ICCC [116] and as a journal paper in IEEE Internet of Things Journal [117].

3.1 Objectives and Contributions

In [36], a one-to-one matching model with peer effects was discussed to match users with RBs in D2D communications. Inspired by [36], due to data transmission of computation offloading suffering limited channel access in IoT networks, matching theory could be a potential way to address the channel allocation problem. In this scenario, each user is taking up one channel at a time while each channel can be accessed by many users using code domain multiplexing, but imperfect orthogonality will cause inference among users. Therefore, many-to-one matching with peer effects could be used to address this challenge. Moreover, the data transmission requires continuous power supply at the IoT users, energy harvesting could provide promising solutions but pose new challenges due to the dynamic energy arrival process. Even though the authors in [53] introduced a first-order MDP to model the stochastic energy arrival and then obtained an optimal policy of transmit power allocation by using DP method, the online optimal policy for users to decide either on energy harvesting or data transmission with dynamic power allocation has not been addressed. The main contributions of this chapter are stated as following:

1. To address resource allocation problem of computation offloading in wireless powered IoT networks, the optimization problem of channel allocation and dynamic power allocation is formulated. To make it traceable, the considered problem is decoupled into two phases: i) assigning the IoT users to the available channels; ii) optimizing transmit power allocation of the IoT users with dynamic energy harvesting.
2. In the first phase, an ECAA is proposed to assign the access channels to the users, which is achieved by enabling users to self-match with the proper channels based

on many-to-one matching theory with peer effects.

3. Within each channel, power allocation schemes are proposed to find the optimal transmit power strategy for each IoT user, which uses MDP to model the transitions of the available power in the battery and channel uncertainties, and then the MDP problem is solved by both DP and RL approaches.

4. Numerical results demonstrate that the proposed ECAA achieves 80% of the optimal performance while it has much lower computational complexity than the exhaustive search approach. Moreover, the proposed power allocation scheme achieves online transmit power allocation with higher throughput performance than an alternative offline scheme.

3.2 Resource Allocation of Computation Offloading in Wireless Powered IoT Networks

3.2.1 System Model

In this chapter, an IoT network is considered, in which the IoT users transmit all the collected data to the gateway for data processing (i.e., full computation offloading), where the gateway is assumed to be able to provide sufficient computation resources for data processing. Assuming there are M channels can be accessed by U_T IoT users. Each user is assumed to be equipped with RF energy harvester and it is configured with a finite-capacity rechargeable battery. The user is assumed to harvest energy from ambient environment, such as wireless power beam or broadcasting TV signals, as shown in Fig. 3.1. Each user wakes up for data transmission when it starts sensing data and requests for data processing. The number of active users is U, and the users are distributed uniformly in a circle in the coverage area of the gateway. The channel set and the IoT user set are denoted as $\mathcal{L} = \{l_1, \ldots, l_m, \ldots, l_M\}$ and $\mathcal{U} = \{u_1, \ldots, u_i, \ldots, u_U\}$, respectively. The bandwidth of each channel l_m is set as B_m Hz.

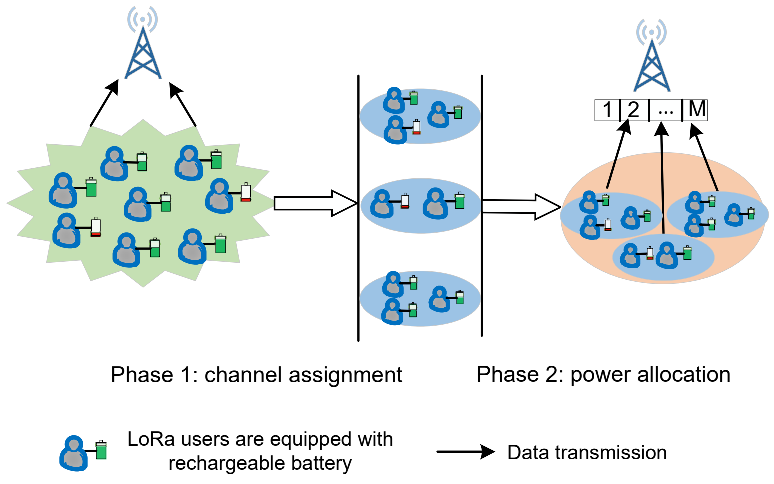

1
2
...
M
Phase 1: channel assignment
Phase 2: power allocation
LoRa users are equipped with rechargeable battery
Data transmission

Figure 3.1: The system model of wireless powered IoT networks.

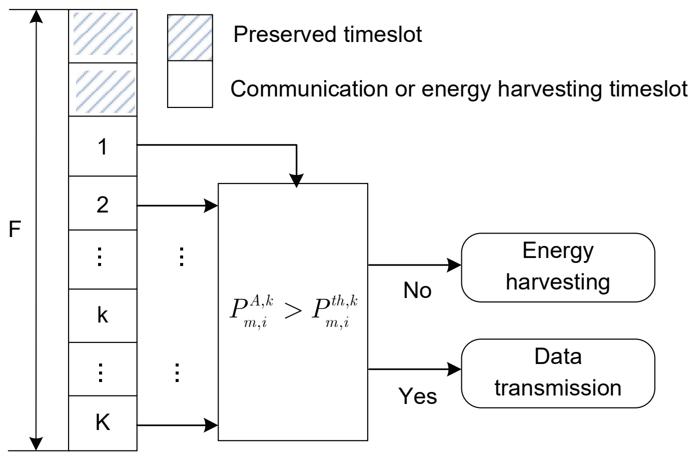

Preserved timeslot
Communication or energy harvesting timeslot
F
1
2
k
K
$P_{m,i}^{A,k} > P_{m,i}^{th,k}$
No
Yes
Energy harvesting
Data transmission

Figure 3.2: The proposed timeslot structure for each wireless powered IoT user.

In the considered scenario, both the gateway and the IoT user are equipped with a single antenna as defined in [9]. Each time frame F is partitioned into different time slots which are used for channel allocation, energy harvesting and data transmission as shown in Fig. 3.2. The considered IoT system is working on time frame structure, and all the users adopt the harvest-then-transmit protocol. Specifically, each frame is assumed to have K slots, indexing k from 1 to K. In each frame F, assuming there are t time slots used for data transmission. Except for the first several time slots used for channel allocation, the remaining slots are dynamically assigned to users for data transmission or energy harvesting.

The available battery power of user u_i in time slot k is set as $P_{m,i}^{A,k}$, $\forall m$, the required power threshold for data transmission is denoted by $P_{m,i}^{th,k}$. It is noted that only when $P_{m,i}^{A,k} > P_{m,i}^{th,k}$, the user starts transmitting data, otherwise it is harvesting energy in time slot k. Let $P_{m,i}^{H,k}$ denote the amount of power harvested by user u_i in time slot k, which is varying over time, so it is defined that $P_{m,i}^{H,k}$ can randomly get a value from a set $\mathcal{P}^{H,k} = [P_1^{H,k}, ...P_q^{H,k}..., P_Q^{H,k}]$ including Q possible values. Denote the battery capacity as P_{max}^A with the transmit power of u_i satisfying $P_{m,i}^k \leq P_{max}^A < +\infty$. Then the available battery power in time slot k is presented as

$$P_{m,i}^{A,k+1} = \min(P_{m,i}^{A,k} - P_{m,i}^k + P_{m,i}^{H,k},\ P_{max}^A). \tag{3.2.1}$$

Since the time slots are discretized and used for data transmission or energy harvesting as shown in Fig. 3.2, the update of available power at each time slot as given by (3.2.1) is reformulated into two cases. When the time slot k is used to harvest energy, the transmit power is $P_{m,i}^k = 0$, (3.2.1) can be expressed as

$$P_{m,i}^{A,k+1} = \min(P_{m,i}^{A,k} + P_{m,i}^{H,k},\ P_{max}^A). \tag{3.2.2}$$

But when it is used for data transmission, the harvested power becomes $P_{m,i}^{H,k} = 0$, (3.2.1) can be expressed as

$$P_{m,i}^{A,k+1} = \min(P_{m,i}^{A,k} - P_{m,i}^k,\ P_{max}^A). \tag{3.2.3}$$

3.2.2 Problem Formulation

In the considered IoT network, there are U active IoT users trying to transmit their data to the gateway. The channel gain $h_{m,i}^k$ over the time slot k in channel l_m is given by

$$h_{m,i}^k = g_{m,i}^k \eta_m \|d_{m,i}\|^{-a}, \tag{3.2.4}$$

where $g_{m,i}^k$ presents the small-scale fading parameter of the channel l_m which is assumed to be Rayleigh fading channel, η_m is a constant related to the path loss of communication link in l_m. The distance between the gateway and IoT user, u_i, is denoted by $d_{m,i}$, and it will not change over different time slots because all the users are assumed static in the network. The path loss exponent a is decided by the carrier frequencies and environment conditions. Therefore, the signal power received at the gateway over l_m is presented as

$$y_m^k = \sum_{i=1}^{U} \alpha_{m,i} P_{m,i}^k h_{m,i}^k + \sigma_m^2, \tag{3.2.5}$$

where the noise is assumed to be Additive White Gaussian Noise (AWGN) with the power as σ_m^2, $P_{m,i}^k$ denotes the uplink transmit power of user u_i when transmitting over channel l_m. $\alpha_{m,i}$ indicates whether l_m is assigned to user u_i, which can be defined as

$$\alpha_{m,i} = \begin{cases} 1, & u_i \text{ occupies } l_m, \\ 0, & \text{otherwise.} \end{cases} \tag{3.2.6}$$

Let I_m denote the number of users assigned to l_m, i.e., $I_m = \sum_{i=1}^{U} \alpha_{m,i}$. The uplink SINR for user u_i transmitting over channel l_m in time slot k can be calculated as

$$\gamma_{m,i}^k = \frac{P_{m,i}^k h_{m,i}^k}{\sigma_m^2 + \sum\limits_{z=1, z\neq i}^{I_m} P_{m,z}^k h_{m,z}^k}, \ \forall\, m,\ i,\ k, \tag{3.2.7}$$

where the interference is introduced due to imperfect orthogonal code domain multiplexing (e.g., adopting different SFs in LoRa).

Given any user, u_i, its achievable accumulative data rate in l_m over the time slot k can be given by

$$R_{m,i}^k = B_m \log_2 \left(1 + \gamma_{m,i}^k\right), \ \forall\, m,\ i. \tag{3.2.8}$$

The objective is to maximize the network throughput while increasing resource allocation efficiency for wireless powered IoT networks under dynamic environment. Then the problem can be formulated as

$$\textbf{(P3.1)} \max_{\alpha_{m,i},\, t,\, P^k_{m,i}} \sum_{k=1}^{t} \sum_{i=1}^{U} \alpha_{m,i} R^k_{m,i}(P^k_{m,i}), \tag{3.2.9a}$$

$$\text{s.t. } C_1: \ 0 \leq P^k_{m,i} \leq P^{A,k}_{m,i}, \ \forall\, m,\ i,\ k, \tag{3.2.9b}$$

$$C_2: \ t \in \{1, ..., K\}, \tag{3.2.9c}$$

$$C_3: \ \alpha_{m,i} \in \{0,1\}, \ \forall\, m,\ i, \tag{3.2.9d}$$

$$C_4: \ \sum_{m} \alpha_{m,i} \leq D, \ \forall\, i, \tag{3.2.9e}$$

$$C_5: \ \sum_{i} \alpha_{m,i} \leq 1, \ \forall\, m, \tag{3.2.9f}$$

where C_1 denotes value range of the transmit power for any user u_i over l_m in time slot k, $P^{A,k}_{m,i}$ is the available battery power at the beginning of time slot k, and it is decided by (3.2.1). In C_2, t is the number of time slots allocated to transmit data in the time frame F. In C_3, $\alpha_{m,i}$ is either 0 or 1. There are at most D users can be assigned to the same channel l_m, which is shown in C_4, and C_5 restricts that one user at most can be allocated to one channel.

It is noted that the problem **(P3.1)** involves integer programming as shown in C_2, the binary constraints as well as stochastic constraint in the objective function, so it is obviously a non-convex problem. As a result, there is no efficiently computational approach to solve the optimization problem **(P3.1)** directly. Thus, from Fig. 3.1, this optimization problem is decomposed into two phases and then solved separately. In the channel allocation phase, all the IoT users are self-matched with the available channels. For simplicity, the matching game is considered static in each time frame, which is performed in the beginning of the frame and is independent of the time slot k in each time frame. Then, the dynamic power allocation algorithms are proposed for the wireless powered users, in which the user either transmits data or harvests energy in the rest time slots until the end of the time frame F. In the next time frame $F+1$, the active users for data transmission are changed, then the channel-user matching problem will be explored again. This can be achieved by formulating a new matching game or modifying the current matching [118].

3.3 Matching based Channel Allocation

From Fig. 3.2, assuming the matching based channel allocation is performed in time slot $k = 0$. For simplicity, in this section, the time slot index is removed. Each user is assumed to transmit data to the gateway using the same transmit power. To guarantee the fairness of all the users, the sub-problem of channel allocation is converted to a max-min problem, which is presented as

$$(\mathbf{P3.2})\ \ \max_{\alpha_{m,i}}\ \min R_{m,i},\ \text{subject to } C_3,\ C_4,\ \text{and } C_5. \tag{3.3.1}$$

Noted that (**P3.2**) is NP-hard, so an ECAA is proposed with low complexity based on matching theory to solve it. The proposed ECAA reduces the probability of re-transmission since channel conflicts between any two users are eliminated, which extends the battery life of users. In the proposed algorithm, the IoT user set, $\mathcal{U}$, and the channel set, $\mathcal{L}$, are considered as two disjoint sets of selfish players which only focus on maximizing their own utilities. Moreover, assuming the CSI is known [1], that is, each user is aware of the CSI of other users. More details of the proposed algorithm will be discussed in Section 3.3.2.

The utility of user u_i, $R_{u_i}^{uti}$, is defined as the minimal transmission rate among those channels it occupies $\mathcal{J}_i$ at time slot k, which is expressed as

$$R_{u_i}^{uti} = \min\Big(B_m \log_2(1+\gamma_{m,i}^{k})\Big), \forall\, m \in \mathcal{J}_i. \tag{3.3.2}$$

Similarly, the utility of l_m, $R_{l_m}^{uti}$ denotes the minimal transmission rate from the set of

[1]The CSI is recorded at the gateway after it receives the requests from users and it is broadcasted to all the users in the next downlink data package.

users, $\mathcal{I}_m$, which share the same channel l_m at time slot k, can be described as

$$R_{l_m}^{uti} = \min\left(B_m \log_2(1+\gamma_{m,i}^k)\right), \ \forall\, i \in \mathcal{I}_m. \tag{3.3.3}$$

3.3.1 Many-to-One Matching

The basic theory of the many-to-one matching model is introduced to address channel allocation problem [119] in this section.

3.3.1.1 Matching Pair

In this chapter, a matching game can be considered to match channels in the channel set $\mathcal{L}$ with users in the user set $\mathcal{U}$, the formal definition is given in the following.

Definition 3.1. By giving two disjoint sets, the channel set $\mathcal{L}$ and the user set $\mathcal{U}$, a many-to-one matching Φ is defined to map the channel set $\mathcal{L}$ to the user set $\mathcal{U}$ including all subsets of $\mathcal{L} \cup \mathcal{U}$ so that for each $l_m \in \mathcal{L}$ and $u_i \in \mathcal{U}$:

1) $\Phi(l_m) \subseteq \mathcal{U}$;

2) $\Phi(u_i) \subseteq \mathcal{L}$;

3) $|\Phi(u_i)| \leq 1$;

4) $|\Phi(l_m)| \leq D$;

5) $l_m \in \Phi(u_i) \Leftrightarrow u_i \in \Phi(l_m)$.

The meanings of the aforementioned conditions are as follows: 1) lets each channel, l_m, match with a subset of users, $\mathcal{U}$; 2) lets each user, u_i, match with a subset of channels, $\mathcal{L}$. 3) implies that at most one channel can be allocated to each user; 4) restricts the number of users assigned to one channel cannot exceed D; 5) denotes the defined matching pair.

Remark 1. From **Definition 3.1**, the formulated matching game is a many-to-one problem with peer effects.

Proof. In the considered IoT network, each user can only be assigned to no more than one channel, and each channel can be assigned with more than one users, so this is a many-to-one matching game. Due to the interference component in (3.2.7), the date rate achieved by an arbitrary user, u_i, over its occupied channel, l_m, related to the set of other users sharing the same channel. Thus, each user cares not only about the channel it is matched with, but also the set of users that are assigned into the same channel. Similarly, for each channel it not only manages the individual user with which to match with, but also the subset of users that have inner-relationships through code domain multiplexing. Thus, this can be formulated as a many-to-one matching game with peer effects, in which each player tries to maximize their own utilities [120]. $\square$

3.3.1.2 Preference Relations

In this part, a preference relation, $\succ$, is defined to illustrate competition behaviors and decision making for both users and channels. Particularly, given any user, $u_i \in \mathcal{U}$, its preference $\succ_{u_i}$ between any two channels, $l_m \in \mathcal{L}$ and $l_{m'} \in \mathcal{L}$ with $m \neq m'$, is presented as

$$\left(l_m, \Phi\right) \succ_{u_i} \left(l_{m'}, \Phi'\right) \Leftrightarrow R_{m,i}\left(\Phi\right) > R_{m',i}\left(\Phi'\right), \tag{3.3.4}$$

where $l_m \in \Phi\left(u_i\right)$, $l_{m'} \in \Phi'\left(u_i\right)$. This definition implies that user u_i prefers l_m in Φ to $l_{m'}$ in Φ' if l_m can provide higher transmission rate than $l_{m'}$. Likewise, given any channel l_m, its preference $\succ_{l_m}$ between any two set of users, $\mathcal{S}_{\mathcal{U}} \in \mathcal{U}$ and $\mathcal{S}_{\mathcal{U}'} \in \mathcal{U}$, is derived as

$$\left(\mathcal{S}_{\mathcal{U}}, \Phi\right) \succ_{l_m} \left(\mathcal{S}_{\mathcal{U}'}, \Phi'\right) \Leftrightarrow R_{m,i}\left(\Phi\right) > R_{m,i'}\left(\Phi'\right), \tag{3.3.5}$$

where $\mathcal{S}_{\mathcal{U}} \in \Phi\left(l_m\right)$ and $\mathcal{S}_{\mathcal{U}'} \in \Phi'\left(l_m\right)$.

3.3.1.3 Swap Matching

From the matching game, the swapping behaviours of players are defined as that each pair of players is supposed to swap their matching but do not change any other players' assignment. Therefore, the detailed concept of *swap-matching* to better explain the interdependency of players' preference is given as in the **Definition 3.2**.

Definition 3.2. Given a matching Φ with $l_m \in \Phi(u_i)$, $l_{m'} \in \Phi(u_{i'})$, $l_m \notin \Phi(u_{i'})$, and $l_{m'} \notin \Phi(u_i)$, a swap matching $\Phi' = \{\Phi \backslash \{(u_i, l_m), (u_{i'}, l_{m'})\}\} \cup \{(u_i, l_{m'}), (u_{i'}, l_m)\}$ is defined by $l_m \in \Phi'(u_{i'})$, $l_{m'} \in \Phi'(u_i)$, $l_m \notin \Phi'(u_i)$, and $l_{m'} \notin \Phi'(u_{i'})$.

It is noticed that the swap matching defines a matching generated by a swap operation. For example, two users exchange their matched channels while keeping all other users' channel assignment the same. Based on the definition of swap matching, the swap-blocking pair is defined as follows.

Definition 3.3. Given a user pair $(u_i, u_{i'})$ that is matched for a given matching Φ, if there is $l_m \in \Phi(u_i)$ and $l_{m'} \in \Phi(u_{i'})$ such that $\forall n \in \{u_i, u_{i'}, l_m, l_{m'}\}$, $R_n^{uti}(\Phi') \geq R_n^{uti}(\Phi)$ and $\exists n \in \{u_i, u_{i'}, l_m, l_{m'}\}$ such that $R_n^{uti}(\Phi') > R_n^{uti}(\Phi)$, so the swap matching Φ' is defined, and $(u_i, u_{i'})$ is defined as a swap-blocking pair in Φ.

The success of a swap matching operation implies that the utility of an arbitrary player, i.e., $R_{u_i}^{uti}$ or $R_{l_m}^{uti}$ as shown in (3.3.2) and (3.3.3), does not decrease, moreover, at least the utility of one player is increased. Noted that both the users and the gateway can initialize the swap operation because the utilities of them are directly relevant to the data transmission rate.

Definition 3.4. If a matching Φ is not blocked by any swap-blocking pair, it is defined as two-sided exchange-stable (***2ES***) [120].

3.3.2 Efficient Channel Allocation Algorithm

In this section, an ECAA based on matching theory is proposed, which has low complexity and is able to match the users with the available channels. The ECAA aims to look for a ***2ES*** matching used for channel allocation after finishing a few swap operations. Furthermore, the battery life of users is extended since the ECAA has low computation complexity.

Algorithm 3.1: *Efficient channel allocation algorithm*

```
Initialization
    Generate the initial matching $\Phi_0$ by Algorithm 3.2.
Optimal algorithm
 1: while $\exists\,(u_i, u_{i'})$ blocks current matching do
 2:   for $\forall u_i \in \mathcal{U}$ do
 3:     for $\forall u_{i'} \in \{\mathcal{U} \backslash u_i\}$ with $l_m \in \Phi\,(u_i)$ and $l_{m'} \in \Phi\,(u_{i'})$ do
 4:       if $(u_i, u_{i'})$ is a swap-blocking pair and $C_2 - C_4$ are satisfied then
 5:         $u_i$ exchanges its match $l_m$ with $u_{i'}$'s match $l_{m'}$.
 6:         Update $\Phi$.
 7:       end if
 8:     end for
 9:   end for
10: end while
11: return final matching $\Phi$.
```

Algorithm 3.2: *Initial matching algorithm*

```
Initialization  Set of unmatched users $\ominus_{\mathcal{UM}} = \mathcal{U}$, $\alpha_{m,i} = 0$, proposal indicator $\beta_{m,i} = 0$,
    $\forall\ m,\ i$.
 1: Calculate preference list of each user $\mathcal{PL}_{u_i}$, $\forall\ u_i \in \mathcal{U}$.
 2: Calculate preference list of each channel $\mathcal{PL}_{l_m}$, $\forall\ l_m \in \mathcal{L}$.
 3: while $\ominus_{\mathcal{UM}} \neq \emptyset$ do
 4:   for $\forall u_i \in \mathcal{U}$ do
 5:     $u_i$ proposes to its first preferred channel that it has not been rejected before.
 6:     Update $\beta_{m,i} = 1$ if $u_i$ proposes to $l_m$.
 7:   end for
 8:   for $\forall\ l_m \in \mathcal{L}$ do
 9:     if $\sum_i (\alpha_{m,i} + \beta_{m,i}) \leq D$ then
10:       $l_m$ accepts all proposals from LoRa users.
11:     else
12:       $l_m$ accepts proposals from the $D$ most preferred users.
13:     end if
14:     Update $\ominus_{\mathcal{UM}}$ by removing all the matched $u_i$.
15:     Remove $l_m$ from $\mathcal{PL}_{u_i}$ if $\beta_{m,i} = 1$.
16:     Update $\Phi_0$ with $\alpha_{m,i} = 1$ for all the matched $u_i$.
17:   end for
18: end while
19: if there are vacant channels, $l_m$ then
20:   Match $l_m$ with its first preferred user.
21:   Update $\Phi_0$.
22: end if
23: return $\Phi_0$.
```

From **Algorithm 3.1**, the ECAA is proposed including initialization algorithm and swap matching algorithm. In the initialization step, an initialization algorithm is proposed to generate the initial matching, Φ_0, which is illustrated in **Algorithm 3.2**. Assuming each user transmits data using the same transmit power. The preference list, i.e., the available channels, of each user is constructed based on the CSI. For example, given u_i, the channel l_m with the best CSI, i.e., $m = \arg\max\limits_{\forall m} h_{m,i}$ is defined as the user's first preference. The preference list is initialized at the gateway for each user by calculating the distance between them and the gateway, that is, the highest preferred user is the closest one. This is because in the considered IoT networks, the achieved transmission rate is mainly affected by the large-scale channel fading. Therefore, each user chooses its first preferred channel from its preference list, and only the proposals of the first D users in the preference list are accepted by each channel. Only when all the users are matched with a channel, this process can be stopped. Specifically, if the channel is not matched with any user, we force it to match with its first preferred user, which is used to improve the minimal achievable data transmission rate. We return the initial matching Φ_0 and consider it as an input for **Algorithm 3.1**.

In the swap-matching algorithm, users keep looking for swap-blocking pairs, while the utilities of both players are guaranteed not to decrease and at least the utility of one player, i.e., the user or the channel, will increase. The swap operation carries out if only one swap-blocking pair appears in the present matching game. The process of searching and swap operation is stopped when reaching the ultimate matching state. Then after **Algorithm 3.1** returning the channel access decision, the preferred channel lists of active users are renewed.

A few theorems have been derived from the proposed ECAA, which are as follows.

Theorem 1. Stability: in the ECAA, the ultimate matching is a ***2ES*** matching.

Proof. For the proposed ECAA, if the final matching Φ contains at least one more swap-blocking pair, at least the utility of one player can be reformative and the utilities of other arbitrary players will not be reduced. Nevertheless, if any pair of players blocks the current matching, Φ, the ECAA will not stop, that is, Φ can not be considered as the final matching, so it will cause conflicts between players. As a result, the final matching Φ is defined as $\boldsymbol{2ES}$. □

Theorem 2. Convergence: after performing swap operations a finite number of times, ECAA is converged to the final matching, $\boldsymbol{2ES}$.

Proof. For the proposed ECAA, it performs finite swap operations since there are limited number of players and each channel is restricted to accept finite number of users. Furthermore, each swap operation will make the achieved minimal transmission rate of each channel increase. The stop condition of the swap operations is defined as when the transmission rate is satisfied in the worst case. This is because there is an upper bound for the achievable transmission rate of each channel since the spectrum resources is limited. As a result, the proposed ECAA is converged to the stable state after performing swap operations a finite number of times. □

Theorem 3. Complexity: there exists an upper bound of the computational complexity for the proposed ECAA, which is calculated by $\mathrm{O}\left(MU + \frac{1}{2}IDU\left(M-1\right)\right)$.

Proof. From **Algorithm 3.1** and **3.2**, the ECAA is proposed including the initial matching as well as the swap operations, so its computational complexity is made up of two parts. In the first part, the worse case that all the users are proposed to all the available channels is considered. In this case, the computational complexity is calculated by $\mathrm{O}\left(MU\right)$.

In the second step, both the number of iterations in the algorithm, ECAA, and the swap operations in each iteration are contributed to the computational complexity. Nevertheless, it takes finite iterations, I, to reach the final matching $\boldsymbol{2ES}$ though there are no closed-form expressions, which is proved in Theorem 2. In each iteration, given any user u_i, $M-1$ possible swap-blocking pairs are generated since the gateway has M

channels to access and each user only occupies one channel at most. There are at most D users are allocated to the selected channel l_m. Thus, there are at most $D\left(M-1\right)$ possible combinations of a swap matching Φ giving any user u_i. The proposed ECAA needs to consider at most $\frac{1}{2}DU\left(M-1\right)$ swap matchings in each iteration. As a result, the upper bound of the complexity in the second step is denoted by $\mathrm{O}\left(\frac{1}{2}IDU\left(M-1\right)\right)$.

In conclusion, the upper bound of the computational complexity of our proposed ECAA is calculated as $\mathrm{O}\left(MU+\frac{1}{2}IDU\left(M-1\right)\right)$. □

It is easily noticed that the ECAA has much lower complexity than the brute force exhaustive-search approach in which the computation becomes more and more complicated, even is increasing exponentially with the increasing number of active users, U.

3.4 RL-based Power Allocation

After performing the ECAA, the users are matched with one of the available channels. For those users allocated to the same channel, the optimal orthogonal code domain multiplexing scheme is performed to eliminate the interference among them [121]. Therefore, the interference component is removed from (3.2.7), so the SNR of the uplink transmission for user u_i can be computed as

$$\gamma_{m,i}^{k}=\frac{P_{m,i}^{k}h_{m,i}^{k}}{\sigma_{m}^{2}},\ \forall\, m,\ i,\ k. \tag{3.4.1}$$

Then, the optimization problem **(P3.1)** is converted as

$$(\mathbf{P3.3})\ \max_{P_{m,i}^{k},\ t}\sum_{k=1}^{t}\sum_{i=1}^{U}R_{m,i}(P_{m,i}^{k}), \tag{3.4.2a}$$

$$\text{s.t. } C_1:\ 0\leq P_{m,i}^{k}\leq P_{m,i}^{A,k},\ \ \forall m,\ i,\ k, \tag{3.4.2b}$$

$$C_2:\ t\in\{1,...K\}. \tag{3.4.2c}$$

Since there is no interference for the uplink data transmission of the users, the network throughput maximization can be converted to maximize the achievable rate of each individual user. Take u_i as an example, to maximize its transmission rate, the dynamic power allocation considering both uncertain channel conditions and random energy harvesting process is explored. The data transmission and energy harvesting process are assumed to be performed separately in different time slots, that is, the transmit power $P_{m,i}^{k} = 0$ when the time slot $k \subseteq \mathcal{K}_h$ is allocated to harvest energy; correspondingly, the harvested power $P_{m,i}^{H,k} = 0$ when the time slot $k \subseteq \mathcal{K}_d$ is used to transmit data. Moreover, $\mathcal{K}_h \cup \mathcal{K}_d = \mathcal{K}$, where $\mathcal{K}$ presents the set of all the time slots in each time frame F except for the reserved time slots for channel allocation. In this chapter, the transmit power allocation is optimized over all the time slots while the time slot used for energy harvesting can be considered as fixed power allocation with the transmit power $P_{m,i}^{k} = 0$. A pre-defined transmit power threshold $P_{m,i}^{th}$ is used to enable data transmission at the user. So the optimization problem is converted into

$$(\mathbf{P3.3.1}) \max_{P_{m,i}^{k}} \; R_{m,i}(P_{m,i}^{k}), \tag{3.4.3a}$$

$$\text{s.t. } C_1: \; P_{m,i}^{th} \le P_{m,i}^{k} \le P_{m,i}^{A,k}, \;\; \forall m, \; i, \; k \subseteq \mathcal{K}_d, \tag{3.4.3b}$$

$$C_2: \; P_{m,i}^{k} = 0, \;\; \forall m, \; i, \; k \subseteq \mathcal{K}_h, \tag{3.4.3c}$$

where C_1 indicates the value range of the transmit power when the time slot is used for data transmission, and C_2 denotes the case that the time slot is used to harvest energy. Since it is hard to know the complete information about channel conditions and available battery power in the future time slots, here, assuming that only some stochastic information of the harvested power $P_{m,i}^{H,k}$ and the channel gain $h_{m,i}^{k}$ for future time slots are available. In this case, the IoT user makes decisions on data transmission or energy harvesting as well as how much power for data transmission over time slots, during this process, the transitions of channel conditions and available battery power have Markov property. Therefore, a finite-horizon MDP is used to model the joint random process of $P_{m,i}^{H,k}$ and $h_{m,i}^{k}$, and then two power allocation algorithms are proposed by exploiting DP method and Q-learning algorithm.

Assuming the channel information is only available in current time slot, that is, $h_{m,i}^{k}$ is known at time slot k. Note that the amount of harvested energy in time slot k is unavailable until the end of the time slot, i.e., it will be known at time slot $k+1$. Thus, the channel gain and the energy arrival process are modelled by first order Markov model. The transition probabilities of them are defined as $\mathbf{P_r}(P_{m,i}^{H,k}|P_{m,i}^{H,k-1})$ and $\mathbf{P_r}(h_{m,i}^{k}|h_{m,i}^{k-1})$, respectively. Since the energy harvesting process and the channel gain transition are independent, the joint probability density function is given by

$$\mathbf{P}(P_{m,i}^{H,k-1},\ h_{m,i}^{k}) = \mathbf{P_r}(P_{m,i}^{H,k-1}|P_{m,i}^{H,k-2})\mathbf{P_r}(h_{m,i}^{k}|h_{m,i}^{k-1}). \tag{3.4.4}$$

As shown in (3.2.1), the available power $P_{m,i}^{A,k}$ in time slot k depends on its last state $P_{m,i}^{A,k-1}$, which also can be modelled as a first order Markov model. Then, the system states in time slot k is defined as

$$s_{m,i}^{k} = (P_{m,i}^{A,k},\ P_{m,i}^{H,k-1},\ h_{m,i}^{k}), \tag{3.4.5}$$

where $s_{m,i}^{k}$ indicates the state of user, u_i, assigned into channel, l_m, it consists of the available battery power, $P_{m,i}^{A,k}$, and the channel gain, $h_{m,i}^{k}$, in the current time slot k, as well as the harvested power, $P_{m,i}^{H,k-1}$, in the previous time slot $k-1$. Based on the current state, $s_{m,i}^{k}$, at time slot k, the user decides to transmit data with power $P_{m,i}^{k}$. That is an action taken at time slot k from its feasible set $\mathcal{A}_{m,i}$

$$\mathcal{A}_{m,i} = \{P_{m,i}^{th} \leq P_{m,i}^{k} \leq P_{m,i}^{A,k}\ and\ P_{m,i}^{k} = 0\},\ \forall m,\ i,\ k. \tag{3.4.6}$$

The threshold power $P_{m,i}^{th}$ is set to enable data transmission at user u_i, which makes the action space reduced. This means the time slots that are waiting for data transmission become available for energy harvesting. And the action $P_{m,i}^{k} = 0$ means the user u_i chooses to harvest energy in time slot k.

The objective is to obtain an optimal policy π^* that maximizes the accumulative expected network throughput over the K time slots for each user. By giving an initial state, $s_{m,i}^0$, the optimal value function can be calculated as

$$V^* = \max_{\pi \in \prod} \sum_{k=1}^{K} \mathbb{E}\{R_{m,i}^k | s_{m,i}^0,\ \pi\}, \tag{3.4.7}$$

where $\mathbb{E}$ indicates the statistical expectation of channel gain and the harvested energy under the transition probabilities. Then the optimization problem is represented as

$$(\mathbf{P3.3.2}) \max_{\pi^k} \quad \mathbb{E}\{[\sum_{k=1}^{K} R_{m,i}^k] | \mathbf{P}\}, \tag{3.4.8a}$$

$$\text{s.t. } C_1 : \ P_{m,i}^{th} \leq P_{m,i}^k \leq P_{m,i}^{A,k}, \ \forall m,\ i,\ k \subseteq \mathcal{K}_d, \tag{3.4.8b}$$

$$C_2 : \ P_{m,i}^k = 0, \ \forall m,\ i,\ k \subseteq \mathcal{K}_h, \tag{3.4.8c}$$

where $\mathbf{P}$ is the state transition probability matrix. In general, this optimization problem is impossible to be solved independently in each time slot because of the causality constraints on the variables. Then, the DP method is used to solve this problem in the next section.

3.4.1 DP-based Power Allocation Algorithm

In practice, the state transition probability matrix $\mathbf{P}$ can be obtained by experiments, which is assumed to be known here. Therefore, the optimization problem (**P3.3.2**) can be solved by formulating it as a DP problem. To solve the DP problem, Bellman's equations are introduced [122]. Then it is expressed as the backward recursive equations, which starts from $k = K$ to $k = 1$.

For $k = K$,

$$V_K(s_{m,i}^K)) = \max_{a_{m,i}^K} \ R_{m,i}^K, \tag{3.4.9}$$

and for $k = K - 1, ..., 1$,

$$V_k(s_{m,i}^k)) = \max_{a_{m,i}^k} r_{m,i}^k + \bar{V}_{k+1}(s_{m,i}^k,\ P_{m,i}^k), \tag{3.4.10}$$

where the second term contains the future reward information from the time slot $k+1$ to K, which is presented as

$$\bar{V}_{k+1}(s_{m,i}^k,\ P_{m,i}^k) = \mathbb{E}_{s_{m,i}^k}\{V_{k+1}(s_{m,i}^{k+1})|\mathbf{P},\ s_{m,i}^k\}, \tag{3.4.11}$$

where $\mathbb{E}_{s_{m,i}^k}$ presents the statistical expectation of all the possible states in future time slot $k+1$ given the current state $s_{m,i}^k$ and the transition probability matrix $\mathbf{P}$. (3.4.11) is considered as the optimal system throughput obtained from the future time slots. It means this problem is to maximize the cumulative network throughput from the current time slot to the last time slot resulted from the current state and the current policy.

In order to solve this problem, the backward induction method based on the Bellman's equations is adopted. Then, a DP-based power allocation algorithm is developed for each IoT user. In **Algorithm 3.3**, the initialization phase initializes user set allocated to the same channel according to the ECAA. Then the DP-based power allocation algorithm is performed, which includes two steps: planning step and transmission step. In the planning step, a look up table is built up to record the optimal policy π^*, that is, the optimal sequence of transmission power from the time slot K to 1 over all the possible states $s_{m,i}^k = (P_{m,i}^{A,k},\ P_{m,i}^{H,k-1},\ h_{m,i}^k)$. By using backward induction method, it starts from the final time slot K, in which all the available power in the battery should be used, i.e., $P_{m,i}^K = P_{m,i}^{A,K}$. Then (3.4.9) is solved for all the possible values of $P_{m,i}^{A,K}$, $h_{m,i}^K$. Afterwards, for $k = \{K-1, ..., 2, 1\}$, the (3.4.10) is calculated recursively for all the possible states. In the transmission phase, the current state $s_{m,i}^k$ is known at time slot k, then comparing the available battery power with the pre-defined threshold to determine if the data transmission is enabled in this time slot. As a result, the optimal policy π^* during the time frame is obtained, that is, each user knows the optimal transmit power

Algorithm 3.3: *DP-based power allocation algorithm*

```
Inputs:
    IoT users are assigned with one of the available channels according to ECAA.
Optimal algorithm
 1: Planning phase
 2: Set $k = K$, calculate $V_K(s_{m,i}^K)$,
    $\forall s_{m,i}^K = \{P_{m,i}^{A,K}, P_{m,i}^{H,K}, h_{m,i}^K\}$, using (3.4.9)
 3: for $k = K - 1 : 1$ do
 4:     Calculate $V_k(s_{m,i}^k)$ using (3.4.10), $\forall P_{m,i}^{A,k}, P_{m,i}^{H,k}$
 5: end for
 6: Transmission phase
    Initializing $P_{m,i}^{A,k}$, $h_{m,i}^k$, $P_{m,i}^{H,k-1}$
 7: for $k = 1 : K$ do
 8:     if $P_{m,i}^{A,k} > P_{m,i}^{th}$ then
 9:         Finding $a_{m,i}^{k*} \in \mathcal{A}_{m,i}$ that maximizes $V_k(s_{m,i}^k)$ from the planning phase, that is,
            $P_{m,i}^{H,k} = 0$
10:     else
11:         The user harvests $P_{m,i}^{H,k}$ amount of power, and the optimal policy in time slot $k$
            $a_{m,i}^{k*} = 0$
12:     end if
13:     IoT user $u_i$ consumes transmit data to the gateway with the transmit power from
        $\pi^*$ in time slot $k$
14:     Update the available power $P_{m,i}^{A,k+1}$ in the battery through (3.2.1)
15: end for
16: return $\pi^*$, $\forall k$
```

at each time slot.

3.4.2 Q-Learning based Algorithm

To solve MDP problem, RL techniques can find the optimal policies without knowing an explicit model of the environmental dynamics, that is, the state transition probability $\mathbf{P}(s_{m,i}^{k+1}|s_{m,i}^k, P_{m,i}^k)$ is unknown or even non-stationary. Therefore, the agent, each IoT user can learn from the interactions with the environment to make decisions on transmit power, data transmission or energy harvesting in each time slot. The formulated dynamic power allocation problem is a classic single-agent finite-horizon MDP problem. In the last section, the optimization problem is solved by DP method using backward induction method which has to know the explicit model of the environment dynamics in advance. In this section, a classic model-free RL algorithm, Q-learning algorithm, is considered to explore the dynamic power allocation under uncertain channel conditions and dynamic energy harvesting process by maximizing the long-term network throughput.

The Q-value, $Q(s,a)$, is defined as the expected cumulative discounted throughput when taking an action $a^k \in \mathcal{A}$ following a policy π for a given state-action pair (s,a). Therefore, the action-value function $Q(s,a)$ is given by

$$Q(s,a) = \mathbb{E}_\pi[r^{k+1} + \lambda Q_\pi(s^{k+1},\ a^{k+1}) | s^k = s, a^k = a]. \tag{3.4.12}$$

To solve the dynamic power allocation problem with Q-learning algorithm, let $R^k_{m,i}$ denote the obtained reward in time slot k, and $Q(s^k_{m,i}, a^k_{m,i})$ denote the calculated throughput for any given state $s^k_{m,i}$ and action $a^k_{m,i}$. A Q-table is built up to save all the possible $Q(s^k_{m,i}, a^k_{m,i})$. And $Q(s^k_{m,i}, a^k_{m,i})$ is updated in each time slot if the new $Q_{new}(s^k_{m,i}, a^k_{m,i})$ is larger than the current value. The $Q(s^k_{m,i}, a^k_{m,i})$ is updated incrementally based on the current throughput and the cumulative discounted Q-value $Q(s^{k+1}_{m,i}, a_{m,i}), \forall a_{m,i} \in \mathcal{A}_{m,i}$ in the next time slot $k+1$.

The one-step Q-update equation is given by

$$Q_{new}(s^k_{m,i}, a^k_{m,i}) \leftarrow (1-\alpha) Q(s^k_{m,i}, a^k_{m,i}) + \alpha(R^k_{m,i} + \lambda \max_{\mathcal{P}_{m,i}} Q(s^{k+1}_{m,i}, a_{m,i})), \tag{3.4.13}$$

where $R^k_{m,i}$ is the throughput calculated from the current state, α is the learning rate $(0 < \alpha \leq 1)$. Q-learning is an online action-value function learning with an off-policy, in each time slot, the Q-value in the next time slot is calculated with all the possible actions that it can take, then the maximum Q-value is chosen that can obtain and record the corresponding action.

Therefore, the optimization problem (**P3.3.2**) is solved by using Q-learning algorithm, and to explore the unknown states instead of trusting the learning values of $Q(s^k_{m,i}, a^k_{m,i})$ completely, the ϵ-greedy Q-learning algorithm is proposed, where the agent picks a random action with a small probability ϵ, or with $1-\epsilon$ it chooses an action that

Algorithm 3.4: *Q-Learning based power allocation algorithm*

```
InPut
    Initialization parameters: discount factor $\lambda$, learning rate $\alpha$, exploration rate $\epsilon$,
    $Q_{\mathcal{S}_{m,i}\times\mathcal{A}_{m,i}}$ table
    Initialize states: set $h^1_{m,i}$, $P^{H,1}_{m,i}$ randomly, pick $P^{A,1}_{m,i}$ from $\mathcal{P}^A_{m,i}$, $k = 1$,
Procedure
 1: while $k \leq K$ do
 2:     $h^k_{m,i}$, $P^{H,k}_{m,i}$ is changed according to a random matrix.
 3:     $e \leftarrow$ random number from [0,1]
 4:     if $e < \epsilon$ then
 5:         Choose action $a^k_{m,i}$ randomly.
 6:     else
 7:         Choose action $a^k_{m,i}$ according to $\arg\min_{a^k_{m,i}\in\mathcal{A}_{m,i}} Q(s^k_{m,i}, a^k_{m,i})$
 8:     end if
 9:     Set $s^{k+1}_{m,i} = (h^{k+1}_{m,i}, P^{H,k+1}_{m,i}, P^{A,k+1}_{m,i})$, where
        $P^{A,k+1}_{m,i} = P^{A,k}_{m,i} - P^k_{m,i} + P^{H,k}_{m,i}$.
10:     calculate the reward $R^k_{m,i}$
11:     update $Q(s^k_{m,i}, a^k_{m,i})$ by (4.5.6)
12:     Set $k = k + 1$
13: end while
```

maximizes the $Q(s^{k+1}_{m,i}, a_{m,i})$ as shown in (3.4.13) in each time slot. Then the Q-learning-based power allocation algorithm is proposed as shown in **Algorithm 3.4**.

3.5 Numerical Results

In this section, the proposed ECAA is verified with simulations and comparing it with the baseline approaches. After performing ECAA, the performance of the proposed DP-based and RL-based power allocation algorithms are shown versus the number of time slots. LoRa communication technology is simulated to demonstrate the proposed algorithms. The simulations are conducted on the MATLAB and use the MDP toolbox.

In the simulations, the LoRa users are randomly distributed around the gateway, and they are located in a circle with the radius $r_c = 10$ km. The LoRa technology works at 868 MHz with $M = 3$ available channels, and the bandwidth of each channel is set the same $B_m = 125$ KHz. Each channel allows at most $D = 6$ users with different SFs ranging from 7 to 12. The noise is set as $\sigma^2 = -174 + 10\log_{10}(B_m)\ dBm$. The path loss exponent is set as $\alpha = 3.5$ for all the communication links. From **Algorithm 3.3**, the maximum battery capacity is $P_{max}^{A} = 30\ dBm$, and the pre-defined threshold to enable data transmission is $P_{m,i}^{th} = 12\ dBm$. A three-state Markov chain is considered to model the channel gain of the uplink transmission between the LoRa users and the gateway, that is, the channels have three possible values: "good", "normal" and "bad". Then the channel gain set is $h_{m,i} = \{0.5 \times 10^{-4},\ 1 \times 10^{-4},\ 1.5 \times 10^{-4}\}$, with the transition matrix

$$\mathbf{P}_h = \begin{bmatrix} 0.3 & 0.7 & 0 \\ 0.25 & 0.5 & 0.25 \\ 0 & 0.7 & 0.3 \end{bmatrix}. \tag{3.5.1}$$

For simplicity, the energy arrival process is modelled as a finite-horizon MDP with four states, the possible values are taken from $\{0, aH_e, bH_e, cH_e\}$. Let one possible value be 0, which is the value of the harvested power of the LoRa user when the time slot k is used to transmit data. H_e is the energy harvesting rate with the value of 15 dBm. The transition probability matrix [54] is given as

$$\begin{aligned} \mathbf{P}_e &= \begin{bmatrix} P_{H_1H_1} & P_{H_1H_2} & P_{H_1H_3} & P_{H_1H_4} \\ P_{H_2H_1} & P_{H_2H_2} & P_{H_2H_3} & P_{H_2H_4} \\ P_{H_3H_1} & P_{H_3H_2} & P_{H_3H_3} & P_{H_3H_4} \\ P_{H_4H_1} & P_{H_4H_2} & P_{H_4H_3} & P_{H_4H_4} \end{bmatrix}, \\ &= \begin{bmatrix} 0.3 & 0.7 & 0 & 0 \\ 0.25 & 0.5 & 0.25 & 0 \\ 0 & 0.25 & 0.5 & 0.25 \\ 0 & 0 & 0.7 & 0.3 \end{bmatrix}. \end{aligned} \tag{3.5.2}$$

where $P_{H_iH_j}, i, j \in \{1, 2, 3, 4\}$ indicates the transition probability of the harvested energy state from state H_i to state H_j.

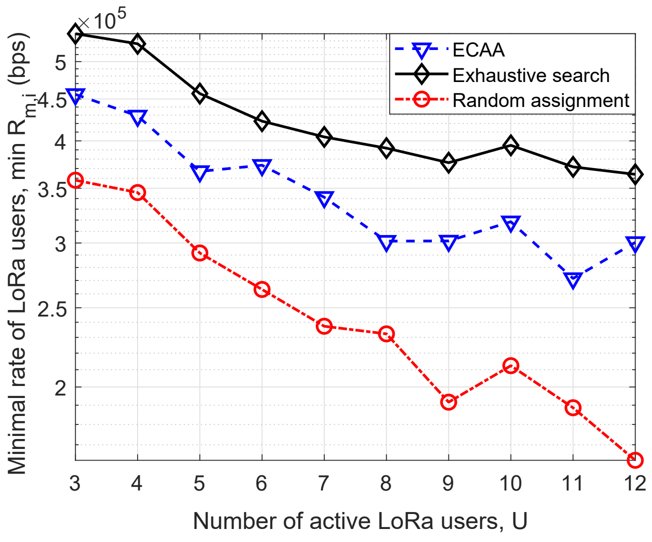


Figure 3.3: Comparison of minimal transmission rates.

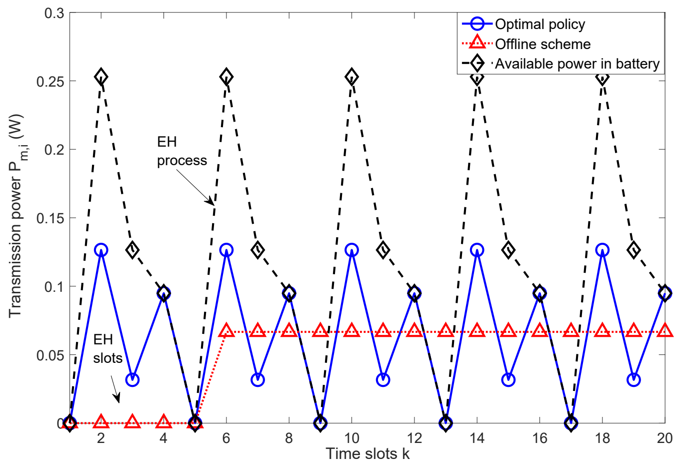


Figure 3.4: The optimal power policy π^* over the time slots.

In Fig. 3.3, the feasibility of the proposed ECAA is validated with different numbers of active LoRa users. For comparison, another two baseline approaches are considered as well, including random channel assignment and brute-force exhaustive search. In this case, each LoRa user has fixed transmit power which is set as $P^{A}_{\max}$, which can remove the effects of different transmit power. It is observed that the proposed algorithm, ECAA, achieves 80% of the optimal performance in exhaustive search. However, ECAA has much lower computational complexity as illustrated in Theorem 3.

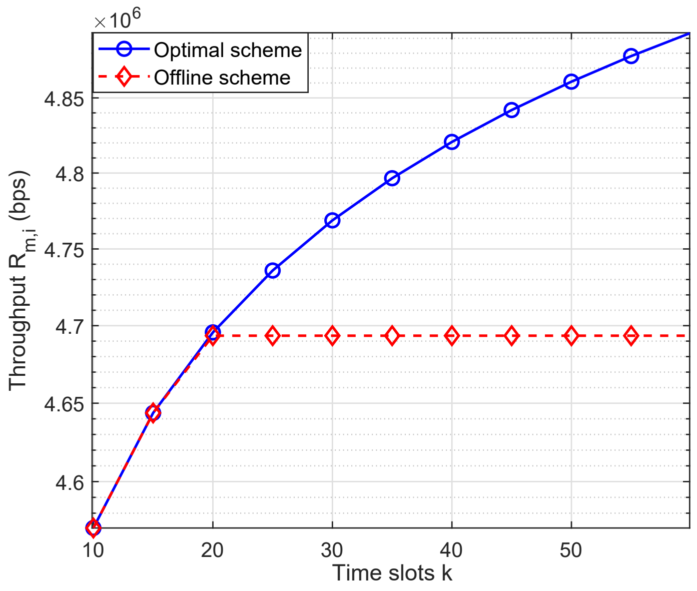


Figure 3.5: Performance comparison of the optimal and offline scheme.

3.5.1 DP-based Power Allocation

Given the transition probability matrix in (3.5.1) and (3.5.2), the power allocation problem can be solved by the DP method. Fig. 3.4 shows the optimal policy over the time slots obtained by the DP-based power allocation algorithm. The energy harvesting rate is set as $\{0, 2H_e, 5H_e, 8H_e\}$. The optimal policy for each LoRa user presents the optimal transmission power in each time slot, moreover, LoRa user switches to energy harvesting when the transmit power $P_{m,i}^k = 0$. The offline scheme is presented for comparison, here, the LoRa user harvests energy in the first few time slots, then uses all the energy stored in the battery to transmit data in the remaining time slots. As a comparison, the number of time slots used for energy harvesting is set the same as the proposed optimal scheme as shown in the Fig. 3.4. It is demonstrated that the dynamic transmit power allocation is achieved over time slots by updating the available battery power in real time through the proposed algorithm.

Fig. 3.5 shows the performance comparison of the proposed optimal scheme and the offline scheme in terms of the achieved network throughput. It is noted that the network throughput of the optimal scheme is increasing with the increasing number of time slots.

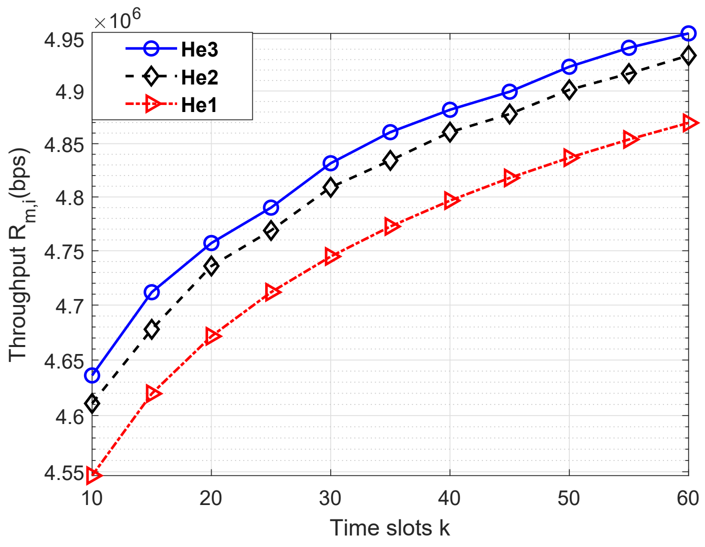


Figure 3.6: The network throughput comparison under different energy harvesting rates.

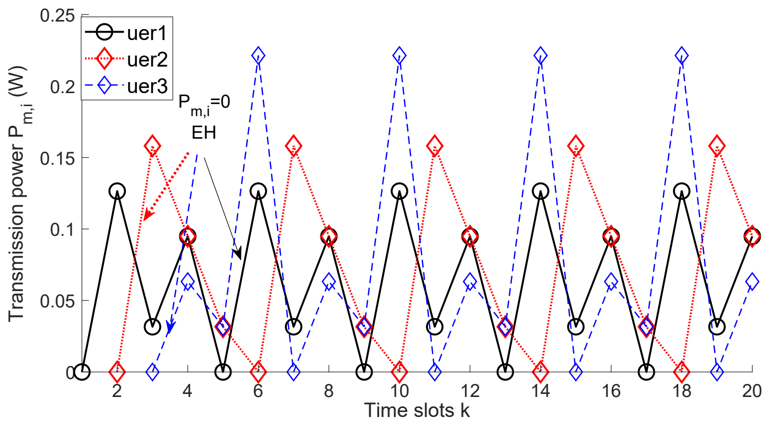


Figure 3.7: The optimal policies for different users allocated to the same channel.

However, the network throughput of the offline scheme only increases before the number of the time slots reaches 20, then it remains the same. This is because that the battery capacity is limited, that is, the maximum amount of the harvested power cannot exceed the battery capacity, i.e., the total available power for the rest time slots are equal to the battery capacity. In Fig. 3.6, the three energy harvesting value vectors are set as: $\mathbf{He_1} = [0, 5H_e, 8H_e, 11H_e]$, $\mathbf{He_2} = [0, 4H_e, 7H_e, 10H_e]$, $\mathbf{He_3} = [0, 2H_e, 5H_e, 8H_e]$. It is observed that the network throughput increases with the increasing of the energy harvesting rate value vectors.

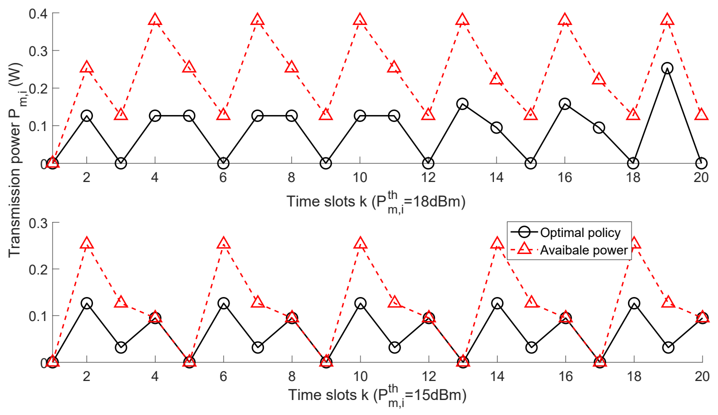


Figure 3.8: The optimal policies under different pre-defined power threshold.

Fig. 3.7 illustrates the optimal policies for different LoRa users allocated to the same channel. The different energy harvesting value vectors are used to distinguish different LoRa users while their channel gains are taken from the same channel gain value vector. In Fig. 3.7, energy harvesting value vectors for different LoRa users are set as user1 $= [0, 1\text{He}, 4\text{He}, 7\text{He}]$, user2 $= [0, 2\text{He}, 5\text{He}, 8\text{He}]$, user3 $= [0, 3\text{He}, 6\text{He}, 9\text{He}]$, respectively. Noted that there are at most 6 users transmitting data at the same channel as the SF ranges from 7 to 12. In this work, by enabling LoRa users to start performing their optimal policies in different time slots, The capacity of each channel is increased by including more users in the same channel. This is because the extra users are allowed to harvest energy which will not cause interference to the users that are transmitting data.

In Fig. 3.8, it shows that the optimal power policy with different pre-defined power threshold is obtained. It is noted that the battery will not run out of its power with increasing the power threshold. This means that the rechargeable battery do not have to discharge all of its power frequently, which will help extend its operation life. Moreover, the maximum transmit power will increase as shown at the bottom figure of Fig. 3.8 after 13^{th} time slot. This is because the user starts transmitting data with more available battery power.

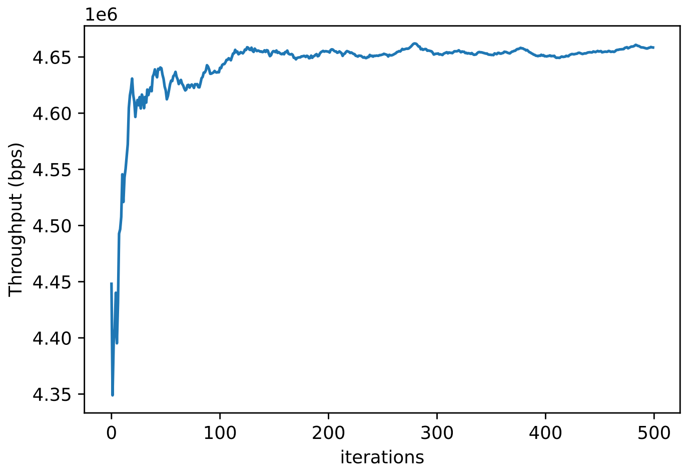


Figure 3.9: The convergence performance of Q-learning based power allocation algorithm.

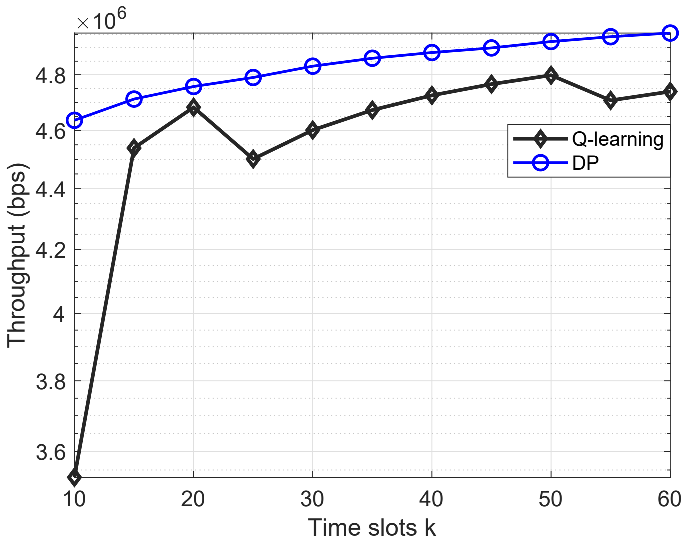


Figure 3.10: The network throughput performance comparison of DP-based and Q-learning based power allocation algorithm

3.5.2 Q-learning based Power Allocation

Q-learning algorithm is a popular model-free RL approach which can interact with the environment to maximize the reward of the system. Here,it does not need any explicit models of the dynamic environment, that is, the transition matrices of channel gain and energy harvesting are no longer needed. At first, the Q table and states are initiated, then to train the Q-table following the Q-update function in (3.4.13), the convergence performance of the Q-learning based power allocation algorithm is obtained in Fig. 3.9. To train the Q-table, let $K = 15$ steps in each iteration and run 5000 iterations. Therefore, the convergence performance of the network throughput is obtained.

In Fig. 3.10, the network throughput performance comparison of both DP-based and Q-learning based power allocation algorithm is presented. It is noted that Q-learning based power allocation algorithm has achieved approximate 90% network throughput performance comparing to the DP-based algorithm that is optimal with the need of transition probability model. Moreover, the Q-learning algorithm needs less environment information, which is superior for the practical implementation.

3.6 Summary

In this chapter, the resource allocation problem in wireless powered IoT networks has been investigated to extend the network lifespan and maximize the network throughput. At first, an ECAA was proposed with low complexity to match the IoT users with the available channels. Then, the transmit power allocation problem under dynamic energy harvesting conditions coupled with uncertain channel conditions was formulated as MDPs, so that both DP method and RL techniques were exploited to solve this problem by proposing DP-based and Q-learning based power allocation algorithms. Simulations have demonstrated that the proposed ECAA achieves 80% of the optimal performance in brute-force exhaustive search but with much lower complexity, and better network throughput performance than the random channel assignment. The DP-based algorithm has achieved dynamic power allocation for each user by maximizing the accumulative network throughput over finite time slots, which has been proved to outperform the offline scheme. Moreover, Q-learning algorithm has been demonstrated to have approximate 90% network throughput performance of the DP-based algorithm with no need of environmental information model. Therefore, the proposed resource allocation methods have achieved a good performance considering system performance and computational complexity.

In this chapter, efficient channel allocation and power allocation have been considered in wireless powered IoT networks. This causes heavy network congestion with all the IoT users trying to transmit all the collected data to the central server, especially for increasingly data-intensive IoT networks. In the next chapter, data processing in IoT networks will be explored with edge computing.

Chapter 4

Computation Offloading in IoT Networks via Machine Learning

As discussed in chapter 3, matching based efficient channel allocation and RL-based power allocation have been investigated to address the challenge of spectrum scarcity and learn the optimal transmit power strategy for the users in wireless powered IoT networks. Here, the IoT users transmit all the collected data to the central server for data processing, which puts heavy burden on the spectrum resource and is very challenging for data-intensive IoT networks. Moreover, compared to the conventional cloud platform, the edge server has finite computation and power resource. Recently, the development of smart chips has given the IoT devices powerful computation capabilities, which means some simple computation tasks can be processed locally at the IoT users such that partial computation offloading is emerged. However, this raises the challenges of which task is processed locally, how to allocate power resource while offloading computation tasks, how to manage spectrum resource for multiuser computation offloading and how to manage the limited computation resource of IoT users. In this chapter, the joint problem of computation offloading and resource allocation in IoT edge computing network is investigated to find the efficient computation offloading strategies for IoT users by deciding which user should offload, which task should offload and how much transmit power should be allocated while offloading through machine learning approaches. At first, the centralized user clustering algorithm is explored to group the IoT users into different clusters according to user priorities. The cluster with the highest priority is assigned to offload computation tasks and perform task execution at the gateway, while the lowest priority cluster is selected to execute tasks locally. For the other clusters, the computation offloading process of each IoT user under dynamic environment is formulated as a RL framework, where each IoT user is considered as an agent which makes a series of decisions on computation offloading by minimizing the long-term system cost. Moreover, this RL framework can be modelled as an MDP, which can be solved by classical model-free RL algorithm, Q-learning. To deal with the curse of high dimensionality, DQN algorithm is adopted to learn the computation offloading strategy in which the DNN is used to approximate the Q-table in Q-learning algorithm. The work presented in this chapter has appeared in IEEE ICC [93] and IEEE Internet of Things Journal [94].

4.1 Objectives and Contributions

The authors in [90, 91] adopted DRL algorithms to address the computation offloading with resource allocation problem, which needs to frequently communicate with the users to train the agent. Inspired by the threshold-based structure of the optimal task offloading policy [75], in this chapter, a centralized clustering algorithm is developed to group the users in IoT networks into different user clusters corresponding to different computation offloading decisions, with defining user priority as the clustering feature. For each user, its computation offloading scheme with resource allocation could be investigated by using RL and DRL algorithms in [84, 85]. However, the computation resource of the edge server and the users was not considered, moreover, the computation offloading scheme was designed to minimize either the energy consumption or the time delay. The major contributions of this paper are stated as following:

1. The joint optimization problem of computation offloading and resource allocation

is addressed for the IoT edge computing networks via machine learning approaches, which is cooperatively solved by centralized user clustering and designing distributed computation offloading strategies.

2. In the user clustering algorithm, the clustering feature is defined by user priority, and based on it, the IoT users are grouped into different clusters by proposing the K-means based clustering algorithm. The clusters with the highest and lowest user priority are assigned as edge computing and local computing model, respectively.

3. Considering a typical IoT user in the remaining clusters, its dynamic computation offloading process while interacting with the uncertain environment is modeled as an MDP, where the objective is to minimize the long-term system cost. Moreover, a DQN-based computation offloading algorithm is proposed for the IoT user to learn the efficient computation offloading strategy.

4. Numerical results show that the IoT users are grouped into different user clusters and the optimal cluster number is validated. Also, the DQN-based computation offloading scheme is demonstrated to be superior to the other baseline schemes in terms of system cost.

4.2 System Model

As shown in Fig. 4.1, an IoT framework is consisted of a variety of independent IoT networks, which has a three-layer hierarchical architecture including the cloud layer (e.g., cloud platform), the edge layer (e.g., the edge server and the IoT gateway) and the local layer (e.g., the IoT users). Each independent IoT network provides services for a large number of IoT users, which is achieved by the gateway collecting data from the IoT users in its coverage area and processing it with its equipped edge server or sending compute-intensive tasks to the cloud layer for further data processing.

An IoT network is considered as an example to explore the joint problem of computation offloading and resource allocation between the local and edge layer. Here, the

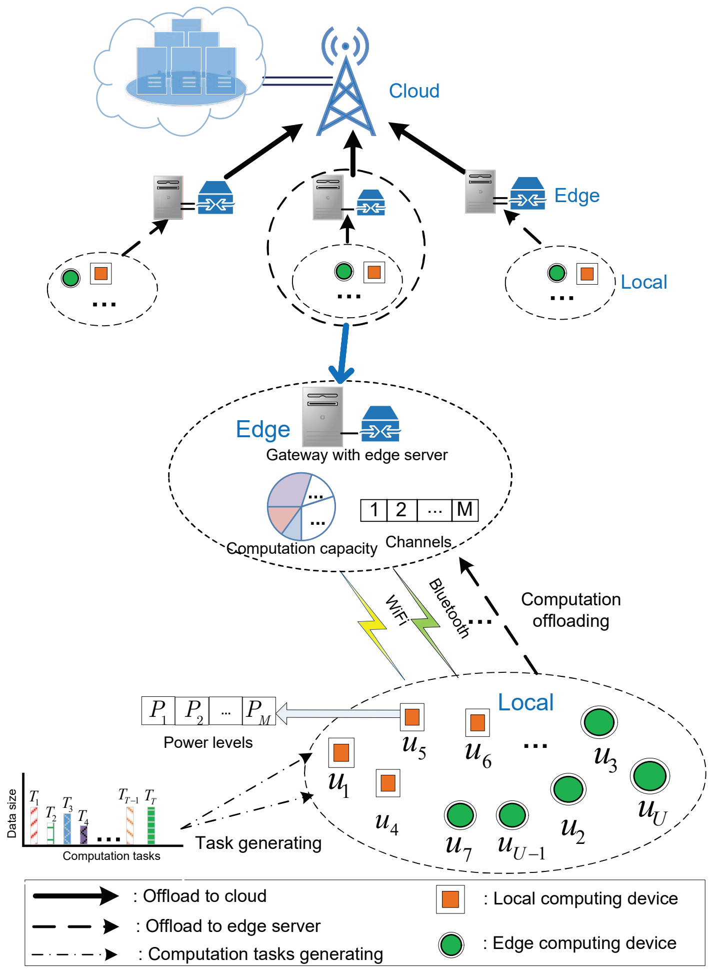


Figure 4.1: The framework of IoT system.

location distribution of the users $\mathcal{U} = \{u_1, ..., u_U\}$ in the considered IoT network is modelled by a Poisson Cluster Process (PCP). From Fig. 4.1, the computation tasks can be either executed locally at the IoT user or offloaded to the gateway and executed at the edge server. However, it is hard for the IoT users to make decisions on computation offloading since it cannot know the perfect knowledge of the environment, like channel gains and other users' decisions. Machine learning algorithms are exploited to design the computation offloading schemes by learning the stochastic model of the dynamic environment. The computation offloading scheme is designed with two steps: centralized user clustering and distributed computation offloading strategy design. Hence, the

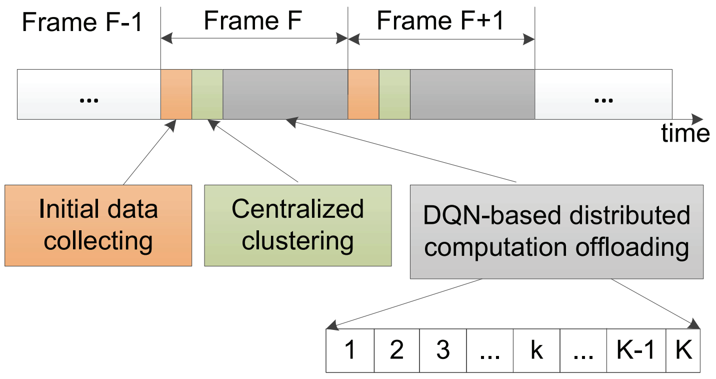


Figure 4.2: Illustration of the timeslot structure for the computation offloading scheme.

proposed scheme is assumed to operate on the time slot structure presented in Fig. 4.2, and the time horizon is discretized into time slots with each time slot indexed by k in each time frame F.

In the considered IoT network, the offloading users transmit their data to the gateway by accessing to a limited number of channels with the bandwidth of each channel denoted by B_m. Assuming each IoT user continuously generates independent computation tasks in different sizes. Each IoT user can store a task queue with the maximum number of tasks not exceeding T. Hence, the possible number of tasks stored at each IoT user in time slot k, i.e., the possible length of task queue t_i^k, is denoted by $t_i^k \in \{T_1, ..., T_T\}$. The task generation is assumed to be an independent and identically distributed sequence of Bernoulli random variables with a common parameter $\tau^k \in [0, 1]$. The indicator of task generation is presented as $\mathcal{I}^k = \{0, 1\}$, where $I^k = 1$ indicates the j^{th} task $T_{i,j}(d_{i,j}, D_{i,j}^{th})$ is generated at user u_i with the task size as $d_{i,j}$ and the task execution delay as $D_{i,j}^{th}$; otherwise, $I^k = 0$ means there is no task generated at current time slot k. Here, $\tau^k = \mathrm{P_r}\{I = 1\} = 1 - \mathrm{P_r}\{I = 0\}$.

4.3 Problem Formulation

The benefits of distributively processing computation tasks are releasing the computation burden on the cloud server, shrinking the task execution latency and reducing the waste of spectrum resource. Edge computing harnesses these benefits by distributing the powerful computation devices to closer to the IoT devices in the IoT networks. Moreover, the more powerful smart chips are deployed in IoT devices, which makes it possible for the users to perform local computing and run machine learning algorithms. In this section, two possible computing models to execute the computation tasks generated in IoT networks are first presented, and then the system cost under each computing model is calculated.

The computation offloading decision and transmit power allocation are jointly considered for each IoT user. Firstly, the possible decisions of each IoT user on computation offloading in time slot k are $\mathcal{O}^k = \{1\} \cup \{0\}$, which indicates whether the task is offloaded or not. Secondly, if the IoT user is selected to offload its computation task, its transmit power level is selecting from a discrete set $\mathcal{P}^T = \{P_1, ..., P_X\}$. Noted that the IoT user does not offload the computation task when $O^k = 0$, then the system cost only contains local computation energy consumption and local task execution latency, and the transmit power is defined as $P^{T,k} = 0$ in this case. In contrast, $O^k = 1$ indicates that the IoT user decides to offload the computation task to the gateway, with the transmit power $P^{T,k} \in \mathcal{P}^T$. In both cases, the computation task is executed successfully, but there is a possibility that the task execution is failed. For example, the task transmission might suffer from communication outage between the IoT user and the gateway, so a penalty function is introduced to cope with this situation.

4.3.1 Local Computing Model

Let $O^k = 0$ if the computation task is executed locally at the IoT user. And let ν denote the fixed CPU frequency of the device, which presents the number of CPU cycles required to compute 1 bit of input data. The energy consumption of per CPU cycle is denoted by e_i^L. Then νe_i^L indicates the energy consumption of per bit data at the IoT user. The total energy consumption of executing the computation task $T_{i,j}$ at the IoT user is denoted by $E_{i,j}^L = \nu e_i^L d_{i,j}$. Moreover, let f_i denote the computation capacity of the IoT user, which is measured by the number of CPU cycles per second. The remaining CPU resource of the IoT user u_i is denoted by the remaining percentage of its computation resource $\rho_i \in [0, 1]$. The local computation latency $D^L_{i,j}$ at the IoT user is defined as $D^L_{i,j} = (\nu d_{i,j})/f_i$. The system cost of choosing the local computing model is defined as

$$C^L_{i,j} = E^L_{i,j} + \beta D^L_{i,j}, \tag{4.3.1}$$

where β indicates the weight factor between the energy consumption and the task execution latency, which combines different types of functions with different units into a universal cost function.

4.3.2 Edge Computing Model

Let h_i denote the channel gain from the IoT user to the gateway, which is assumed to be constant during each time slot. P^T_i is the transmit power of the IoT user. Then the achievable transmission rate (bit/s) of the user u_i by transmitting the task $T_{i,j}$ is denoted by

$$R_i = B_m log_2(1 + \frac{P^T_i h_i}{\sigma^2}), \tag{4.3.2}$$

where σ^2 indicates the power of the AWGN noise. Then the energy consumption of the IoT user consumed by the data transmission is calculated as $P^T_i D^T_{i,j}$, where the transmission delay is given by $D^T_{i,j} = d_{i,j}/R_i$. Similarly, let ν denote the CPU frequency and e^E denote the energy consumption per CPU cycle at the gateway. f indicates the computation capacity of the gateway. The computation energy of the IoT user at the gateway is given by $E^E_{i,j} = d_{i,j}\nu e^E$, and the computation latency is calculated by $D^E_{i,j} = (d_{i,j}\nu)/f$. Therefore, the system cost of choosing the edge computing model can be derived as

$$C^E_{i,j} = E^E_{i,j} + P^T_i D^T_{i,j} + \beta(D^E_{i,j} + D^T_{i,j}). \tag{4.3.3}$$

As discussed above, the task execution is assumed to be successful in both cases. But there is a possibility that the task execution is failed, like the task transmission might suffer transmission outage. Then a penalty value δ is defined as the cost when the task execution fails. Therefore, the system cost is reformulated into a piece-wise function as

$$C_i^k = \begin{cases} C_{i,j}^L & if\ local, \\ C_{i,j}^E & if\ edge,\ not\ failed, \\ \delta & if\ edge,\ failed. \end{cases} \tag{4.3.4}$$

4.4 Centralized User Clustering

4.4.1 User Priority Initialization

The considered IoT network is consisted of a set of IoT users as $\mathcal{U} = \{u_1, ..., u_i..., u_U\}$, where users are requesting for computation services from the resource-constrained gateway, but it is noted that those users have different priorities of task execution. In this section, a centralized user clustering method is investigated to group the IoT users into different user clusters according to user priorities which are then assumed to have similar values across users in the same cluster. Here, the user priority of the IoT user u_i is defined by its channel gain h_i (mainly depending on the large-scale fading) to the gateway and its computation offloading probability $\mathbf{P}_i$. The definition of computation offloading probability $\mathbf{P}_i$ for IoT user u_i is given in **Definition 4.1**.

Each task is generated with its requirement on task execution delay at the IoT user u_i, if the local computing at the user cannot meet the delay requirement, the task has to be offloaded to the gateway for execution. An initial process is allocated in the clustering algorithm for the gateway to initialize the users' priorities, which is shown in Fig. 4.2. It contains that the users calculate and send their computation offloading probabilities to the gateway, and then the gateway records the channel gain of all the users. After that, the gateway performs the clustering algorithm to group the IoT users into a bunch of clusters based on the initialized user priorities..

Definition 4.1. *Considering the IoT user u_i, the task $T_{i,j}$ is generated with its task* *execution delay requirement as $D_{i,j}^{th}$. Assuming there are M_T tasks are generated in each time frame F. The task execution delay with the local computing model at the IoT user u_i is calculated as $D_{i,j}^{L} = d_{i,j}\nu/f_i$. In time frame F, the computation offloading probability of user u_i is defined as the frequency of offloading tasks from time frame $\iota = 1$ until time frame $\iota = F-1$,*

$$\boldsymbol{P}_i = P_r(D_i^L > D_i) = P_r(\sum_{\iota=1}^{F-1}\sum_{j=1}^{M_T} D_{i,j,\iota}^{L} > \sum_{\iota=1}^{F-1}\sum_{j=1}^{M_T} D_{i,j,\iota}^{th}). \tag{4.4.1}$$

Here, the generated tasks are independent. This is calculated in the initial process in the clustering algorithm shown in Fig. 4.2, and later it's improved and updated in each time frame F.

4.4.2 Priority-driven Clustering based on K-means

As discussed above, let x_i indicate the user priority of user u_i, which is defined by the feature vector including the channel gain h_i from user u_i to the gateway and the computation offloading probability $\mathbf{P}_i$, so that $x_i = \{h_i, \mathbf{P}_i\}$. Therefore, a priority-driven user clustering algorithm based on the standard K-means algorithm is proposed to group IoT users into H clusters according to user priorities by minimizing the sum of squared error

$$\min_{\mathcal{C}_h, c_h} \sum_{h=1}^{H} \sum_{x_i \in \mathcal{C}_h} \|x_i - c_h\|_2^2, \tag{4.4.2}$$

where H is the initial number of clusters, $\mathcal{C}_h$ indicates the cluster h, x_i is the user priority feature of user u_i. Here, the centroid of the cluster $\mathcal{C}_h$ is calculated as

$$c_h = \frac{1}{|\mathcal{C}_h|} \sum_{x_i \in \mathcal{C}_h} x_i, \tag{4.4.3}$$

where $|\cdot|$ is the cardinality and the detailed description is provided in **Algorithm 4.1**.

It is noted that the optimal cluster number H_{op} is hard to get. One good method to validate the optimal cluster number is called the elbow method. The basic idea is to run

Algorithm 4.1: *Priority-driven Clustering based on K-means*

```
1: Input:
2: The number of observed tasks $J$, clusters $H$ and clustering feature vectors, user
   priorities $\mathbf{X} = \{x_1, ...., x_i, ..., x_U\}$
3: Initialization:
4: Randomly initialize $H$ cluster centroids $c_1, ..., c_H$
5: Repeat:
6: for $i = 1 : U$ do
7:    for $\hat{h} = 1 : H$ do
8:       Calculate cluster index $\hat{h}_i$ for $x_i$ as :
         $\hat{h}_i = \arg\min_{\hat{h}} \|x_i - c_h\|_2^2$
9:    end for
10: end for
11: for $\hat{h} = 1 : H$ do
12:    Update the cluster centroids by
       $c_h := \frac{\sum_{i=1}^{U} 1\{\hat{h}_i = \hat{h}\} x_i}{\sum_{i=1}^{U} 1\{\hat{h}_i = \hat{h}\}}$
13: end for
14: Until: No change of the cluster centroids
```

K-means clustering algorithm on the dataset for a range values of cluster number H. If the selected cluster number H' is smaller than the real cluster number H, the sum of squared errors (SSE) is reduced dramatically when the cluster number increases by 1. Inversely, the SSE does not have obvious changes. Then the optimal cluster number is located at the turning point of the curve, known as the elbow point. The performance index of the elbow method, i.e., SSE, is defined as

$$SSE = \sum_{h=1}^{H'} \sum dist(x_i, c_h)^2, \tag{4.4.4}$$

where the SSE indicates the sum of squared errors between the sample point x_i and the cluster centroid c_h. $\sum dist(x_i, c_h)^2$ denotes the distoration of cluster h, the smaller the distoration is, the closer the sample points in the cluster h are; otherwise the looser the sample points are. The distoration is decreasing with increasing the number of clusters, but it will receive significantly decrease at a turning point, and then it decreases slowly.

However, the elbow method is sometimes ambiguous. The average silhouette method can be an alternative to validate the optimal cluster number jointly. This method defines a Silhouette Coefficient to measure the similarity of a user to its own cluster compared to its neighboring clusters, its Silhouette Coefficient is calculated by

$$SC_i = \frac{b_i - a_i}{\max\{a_i, b_i\}}, \; if \;\; |\mathcal{C}_h| > 1, \tag{4.4.5}$$

where a_i indicates the mean distance between the user u_i and all the other users in the same cluster $\mathcal{C}_h$, b_i is the mean distance between the user u_i and all the users in its closest neighboring cluster $\mathcal{C}_l$ which is given by

$$\mathcal{C}_l = \arg\min_{\mathcal{C}_h} \frac{1}{q} \sum_{x_l \in \mathcal{C}_h} |x_l - x_i|^2, \tag{4.4.6}$$

where q indicates the number of users in the cluster $\mathcal{C}_h$. Then the average Silhouette Coefficient is obtained by calculating the mean of SC_i over all the users. The average Silhouette Coefficient is in the range $[-1, 1]$, the higher value means the cluster number is more appropriate for the clustering algorithm.

Noted that the proposed priority-driven user clustering algorithm is the first method to cluster IoT users by defining user priority of user's task execution; other explorations could be done in the future with different features, such as radio propagation, usage, user mobility pattern, etc. Moreover, the user clustering algorithm is performed periodically based on the coverage of users' location changes and the monitored task type changes.

4.4.3 Complexity Analysis

Finding an optimal solution to the clustering problem for observations in N dimensions (feature vector) is an NP-hard problem. If the cluster number H and the dimension size N are fixed, the computational complexity can be exactly calculated as $O(U^{NH+1})$, where U is the number of users that need to be clustered and N is the dimension of the cluster feature vector. It depends on the two main steps: calculate the cluster index and update the cluster centroids. There are many heuristic algorithms that can be used to find the optimal solution, like Lloyd's algorithm which has the complexity as $\mathcal{O}(IUNH)$.

Here, I is the iteration number until the algorithm has converged and it has $I = 2^{\mathcal{O}(\sqrt{U})}$ iterations in the worst case.

4.5 DRL-based Computation Offloading

After running the centralized user clustering algorithm, the IoT users are grouped into different clusters by the gateway according to their unique features, user priorities. Here, the cluster with the highest priority is directly designated as edge computing while the cluster with the lowest priority assigned as local computing. In each cluster, all the IoT users are assumed to have the same priorities to offload their tasks. For the remaining clusters, the computation offloading scheme is further explored by DRL techniques in this section. First, the basic formulations of RL are presented and then the dynamic computation offloading process of the IoT user is modelled by an MDP. At last, a DQN-based computation offloading algorithm is proposed to learn the efficient computation offloading strategy, which can address the curse of dimensionality while solving the MDP problem.

4.5.1 The RL Preliminaries

As shown in Fig. 4.3, RL is a dynamic process where the agent continuously learns optimal actions to take by interacting with the environment. In this scenario, each IoT user is considered as an agent and everything else in the IoT network is made up of the environment. In this work, through the centralized user clustering, the scale of the computation offloading problem in the IoT network is reduced. The computation offloading process is discretized into time slots with the time slot structure shown in Fig. 4.2. From Fig. 4.3, each agent, i.e., the IoT user u_i, observes a state s_i^k from state space $\mathcal{S}_i$ at time slot k, and then an action a_i^k is taken from the action space $\mathcal{A}_i$, that is, decides which computing model and how much transmit power are taken based on the policy π_i. Therefore, the environment changes with the taken action, then the new state s_i^{k+1} is observed and a reward r_i^k is obtained by the user. In this scenario, the reward

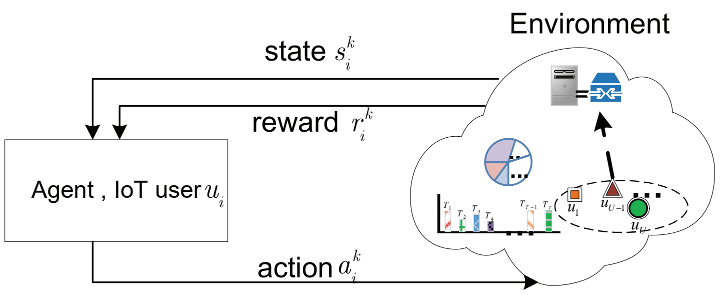


Figure 4.3: The RL framework for computation offloading in IoT networks

is designated as a negative reward, the system cost C_i^k, i.e., the weighted sum of energy consumption and task execution delay.

In a RL problem, the few key elements are $\{Action, State, Reward, Environment\}$, and which can be formulated as an MDP problem.

Action

The computation offloading decision and the transmit power of the IoT user are combined as the action space, so the action set is denoted by $\mathcal{A}_i = \{0, \mathcal{P}_i^T\}$. Here, in time slot k, $P_i^T = 0$ indicates choosing the local computing model while $\mathcal{P}_i^T$ is the discretized transmit power set in edge computing model [1].

State

The possible states observed by the agent represent the exploration information from the environment. Here, the channel gain h_i^k, task queue t_i^k stored at the IoT user and its remaining computation resource ratio ρ_i^k are consisted of the state, which presents the exploration information of the IoT user. Hence, the observed state of the user at the time step k is given as

$$s_i^k = \{h_i^k, t_i^k, \rho_i^k\}, \tag{4.5.1}$$

where $s_i^k \in \mathcal{S}_i$, $h_i^k \in \mathcal{G}_h$. $\mathcal{G}_h$ is the channel gain set of IoT users in the cluster $\mathcal{C}_h$ that has

[1]Assume that the transmit power is larger than 0 while offloading tasks to the edge

not been assigned any direct decisions on computation offloading yet, which is obtained after performing the centralized user clustering algorithm.

Reward

The function of the reward signal is to encourage the learning algorithm to reach the goal of the optimization problem. In this chapter, the negative reward is adopted to minimize the long-term system cost while making the right decisions on computation offloading and transmit power allocation. Here, the system cost is measured by the weighted sum of energy consumption and task execution latency as derived in (4.3.1) and (4.3.3). To make it clear, the reward function is defined as a piece-wise function. Therefore, the reward function of each IoT user at time slot k is given by

$$r_i^k = \begin{cases} C_{i,j}^L, & if\ local, \\ C_{i,j}^E, & if\ edge,\ 1-p, \\ \delta, & if\ edge,\ p, \end{cases} \tag{4.5.2}$$

where p is the failure rate of task transmission (considering the possibility of failed task transmission) in the edge computing model, for simplicity, which is set as a fixed value in this work, and δ is the penalty value of failed task execution. Here, the objective function is given as a negative reward, i.e., the system cost shown in (4.3.4).

In addition, the state transitions of the computation offloading process are stochastic and can be modelled as an MDP problem, where the state transition function and the reward only depend on the environment and the obtained policy. The transition probability $\mathbf{P} = (s_i^{k+1}, r_i^k | s_i^k, a_i^k)$ is defined as the transition from state s_i^k to s_i^{k+1} with the obtained reward r_i^k when the action a_i^k is taken according to the policy π_i. Generally, the MDP problem is solved by finding an optimal strategy that will maximize some cumulative function of the random rewards. In this case, the expected long-term cumulative discounted reward over a finite horizon is given by

$$V_i(s_i, \pi_i) = \mathbb{E}\left[\sum_{\tau=0}^{+\infty} \lambda^\tau r_i(k+\tau) \middle| s_i^k = s_i, \pi_i\right], \tag{4.5.3}$$

where $\lambda \in [0, 1]$ is the discount factor and $\mathbb{E}$ indicates the statistical conditional expectation with transition probabilities.

Basically, the conventional solutions, like policy iteration and value iteration [123] belonging to dynamic programming can be used to solve the MDP optimization problem with the known state transition function. But it is hard for the agent to know the prior information of the state transition function, which is determined by the environment. Therefore, a model-free RL approach is proposed to investigate this decision-making problem since the agent cannot make predictions about what the next state and cost will be before it takes each action.

In this scenario, the IoT user is trying to obtain an optimal computation offloading strategy and transmit power allocation solutions according to some stochastic information, such as the possible channel conditions, the possible remaining computation resource of the user and the possible task queue, observed from the environment. Particularly, the user is finding the optimal policy π_i^* that minimizes the long-term reward $V(s_i, \pi_i)$. This means for any given network state s_i, the optimal policy π_i^* can be obtained by

$$\pi_i^* = \arg\min_{\pi_i} V(s_i, \pi_i),\ \forall s_i \in \mathcal{S}_i. \tag{4.5.4}$$

The problem of jointly designing computation offloading strategy and transmit power allocation for the IoT user can be formulated as a classic single-agent finite-horizon MDP problem. The classic model-free RL approach, Q-learning algorithm, is an effective way to learn the optimal computation offloading strategy by minimizing the expected long-term accumulated discounted reward, $V(s_i, \pi_i)$. $Q(s_i, a_i)$ denotes the Q-value that is the expected accumulated discounted reward when taking an action $a_i^k \in \mathcal{A}_i$ following a policy π_i for a given state-action pair (s_i, a_i). Thus, the action-value function $Q(s_i, a_i)$ is defined as

$$Q(s_i, a_i) = \mathbb{E}_{\pi_i}[r_i^{k+1} + \lambda Q_{\pi_i}(s_i^{k+1},\ a_i^{k+1})|s_i^k = s_i, a_i^k = a_i]. \tag{4.5.5}$$

In the proposed algorithm, $Q(s_i, a_i)$ indicates the value calculated from cost function (4.3.4) for any given state s_i and action a_i, it is stored in the Q-table which is built up to save all the possible accumulative discounted reward. And the current $Q(s_i^k, a_i^k)$ is updated during the time slot k if the new $Q(s_i^{k+1}, a_i^{k+1})$ is smaller than the current $Q(s_i^k, a_i^k)$. The $Q(s_i^k, a_i^k)$ is updated incrementally based on the current reward function r_i^k and the discounted $Q(s_i^{k+1}, a_i), \forall a_i \in \mathcal{A}_i$ in the next time slot. This is achieved by the one-step Q-update equation

$$Q(s_i^k, a_i^k) \leftarrow (1 - \alpha)Q(s_i^k, a_i^k) + \alpha(r_i^k + \lambda \min_{a_i} Q(s_i^{k+1}, a_i)), \tag{4.5.6}$$

where r_i^k is the reward obtained in the current state, α is the learning rate. Q-learning is a model-free and off-policy solution to solve the MDP problem, in each time slot, the Q-value in the next time slot is calculated with all the possible actions that it can take, then it chooses the minimum Q-value and records the corresponding action.

Moreover, to explore the unknown states instead of trusting the learned values of $Q(s_i, a_i)$ completely, the ϵ-greedy approach is used in the Q-learning algorithm, where the agent picks a random action with small probability ϵ, or with $1 - \epsilon$ it chooses an action that minimizes the $Q(s_i^{k+1}, a_i)$ as shown in (4.5.6) in each time slot.

4.5.2 Optimality and Approximation

The agent in the RL algorithm aims to solve sequential decision-making problems by learning an optimal policy. In practice, the requirement for Q-learning to obtain the correct convergence performance is that all the state action pairs $Q(s, a)$ keep to be updated. Moreover, if the policy is explored infinitely, the Q-value $Q(s, a)$ has been validated to converge with possibility 1 to $Q^*(s,a)$, which is given by

$$\lim_{n\to\infty} \mathbf{P_r}(|Q^*(s,a) - Q(s,a)_n| \geq \varsigma) = 0, \tag{4.5.7}$$

where n is the index of the obtained sample, and $Q^*(s,a)$ is the optimal Q-value while $Q(s,a)_n$ is one of the obtained samples. Therefore, Q-learning can identify an optimal action selection policy based on infinite exploration time and a partly-random policy for a finite MDP model.

4.5.3 DQN-based Computation Offloading Algorithm

As mentioned above, the classical Q-learning algorithm can find the optimal strategy when the state-action space is small, and it depends on a Q-table to store the Q-values and has to look up for every state in the table while training. If the state-action space becomes huge, it takes a long time for the Q-table to converge. The reason is that the states will be visited infrequently and the corresponding Q-values are updated rarely for a large number of states. In the considered problem, multiple state dimensions and continuous environment variations generate a large state space. To solve this problem, a neural network is used to approximate the Q-function, which can predict the Q-values based on the input (states), that is, once the weight vector $\boldsymbol{\theta}$ is obtained through training, we can get the output Q-values $Q(s_i^k, a_i^k)$. Specifically, DNN has achieved successful performance in approximating the Q-table, known as DQN. DNN can be applied to the larger scale problems, and it's appropriate to address sophisticate mappings from states to the desired Q-values output.

The DQN follows the rule of neural network to update its weight vector $\boldsymbol{\theta}$ at each iteration by minimizing the loss function,

$$Loss(\boldsymbol{\theta}) = E[(Q'(s_i^k, a_i^k) - Q_{target}(s_i^k, a_i^k; \boldsymbol{\theta}))^2], \tag{4.5.8}$$

where $Q_{target}(s_i^k, a_i^k; \boldsymbol{\theta})$ is the target output Q-value, and the current Q-value $Q'(s_i^k, a_i^k)$

is the prediction information which is given by

$$Q'(s_i^k, a_i^k) = r_i^k + \min_{a_i \in \mathcal{A}_i} Q(s_i^k, a_i, \boldsymbol{\theta}), \tag{4.5.9}$$

where r_i^k is the corresponding negative reward, i.e., the system cost.

Here, the agent uses a replay memory with finite size to store state transitions $(s_i^k, a_i^k, r_i^k, s_i^{k+1})$. The experience replay technique is adopted to sample a mini-batch from the memory pool, which is then used to train the DQN in the direction of minimizing the loss function (4.5.8). The detailed process of training the DQN-based computation offloading algorithm is presented in **Algorithm 4.2**. After the algorithm training is finished, given an initial state, the IoT user selects each action that has the minimum estimated $Q(s_i^k, a_i, \boldsymbol{\theta}^*)$ to obtain the optimal computation offloading strategy and transmit power allocation. Hence, the test algorithm is proposed as a DQN-based computation offloading algorithm shown in **Algorithm 4.3**.

Algorithm 4.2: *DQN training for computation offloading*

```
Initialization
    Initialize the size of replay memory and the mini-batch,
    Initialize Q-network with random weight vector $\boldsymbol{\theta}$,
    Initialize parameters: discount factor $\lambda$, learning rate $\alpha$, exploration rate $\epsilon$, and the
    channel gain set $G_h$ in the clusters $\mathcal{C}_h$
Procedure
 1: while $i' \leq$ iteration do
 2:     Observe initial state $s_i^k = (h_i^k, t_i^k, \rho_i^k)$
 3:     Select an action $a_i^k$ according to $\epsilon$-greedy policy.
 4:     After taking action $a_i^k$, calculate the cost $r_i^k$ by (4.5.2), and new state $s_i^{k+1} =$
        $(h_i^{k+1}, t_i^{k+1}, \rho_i^{k+1})$ is observed.
 5:     Store the state transition $(s_i^k, a_i^k, r_i^k, s_i^{k+1})$in the replay memory
 6:     Randomly sample a mini-batch of transitions from the experience pool
 7:     Update weight vector $\boldsymbol{\theta}$ by minimizing (4.5.8)
 8:     Every C steps update the Q-network
 9:     Update time epoch: $k = k + 1$
10: end while
```

Algorithm 4.3: *DQN-based computation offloading algorithm*

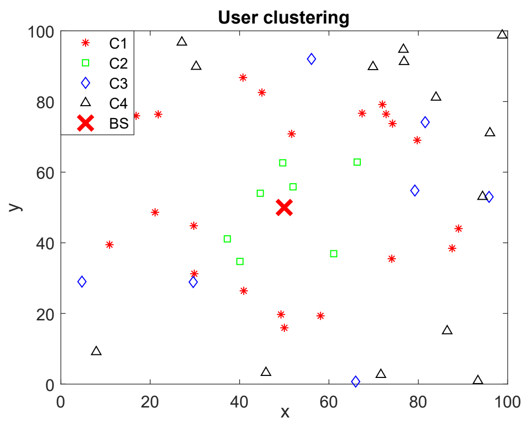
```
Initialization
    Given an initial state $s_i^1 = (h_i^1, t_i^1, \rho_i^1)$
1: while $t_i^k > 0$ do
2:     Select an action $a_i^k = \min_{a_i \in \mathcal{A}_i} Q(s_i^k, a_i, \boldsymbol{\theta}^*)$.
3:     State $s_i^{k+1} = (h_i^{k+1}, t_i^{k+1}, \rho_i^{k+1})$ is observed.
4: end while
```

4.5.4 Complexity Analysis

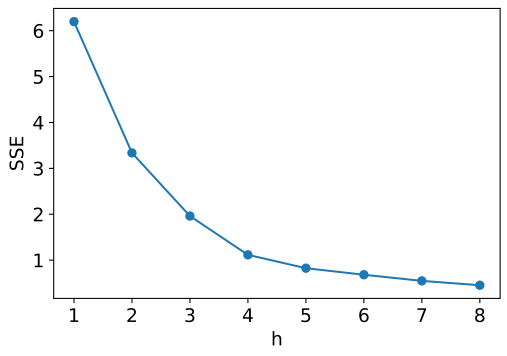
In this section, the complexity of the proposed DQN-based computation offloading algorithm is analyzed, which is mainly determined by the training algorithm shown in **Algorithm 4.2**. For Q-learning algorithm, its time complexity depends on the episodes I and the steps H_s in each episode, denoted as $O(IH_s)$. In DQN, DNN is used to approximate the Q-value function in Q-learning algorithm. Hence, its time complexity mainly comes from training the DNN, which can be calculated as $O(H_i \cdot H_1 + H_1 \cdot H_2 + ... + H_n \cdot H_o)$. Moreover, how many times required for training the DNN relies on the total episodes I and the replay memory size D, denoted by $\mathcal{W} = \lfloor (I/D) \rfloor$. Here, the input layer of the DNN is determined by the batch size $\mathcal{B}$ and state space $\mathcal{S}$, i.e., $H_i = \mathcal{BS}$, while the output layer depends on the action space $\mathcal{A}$. Therefore, the time complexity of the proposed algorithm is $O(W \times (\mathcal{BS} \cdot H_1 + H_1 \cdot H_2 + H_2 \cdot H_3 + H_3 \cdot \mathcal{A}))$.

4.6 Numerical Results

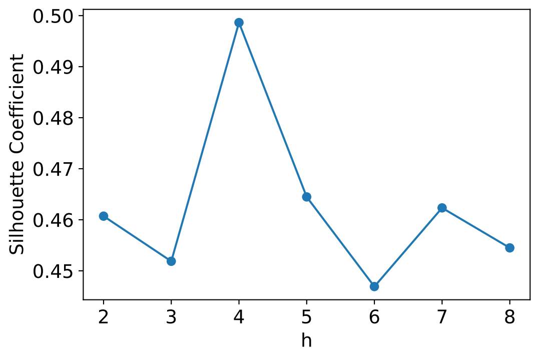
In this section, the joint design of computation offloading strategy and transmit power allocation is analyzed via the simulations. The proposed computation offloading scheme is achieved by user clustering and DRL-based computation offloading design. Here, user clustering is based on K-means clustering algorithm according to users' unique features, user priorities. After running centralized user clustering algorithm, the users are grouped into different user clusters where the cluster with the highest user priority is designated as edge computing while the cluster with the lowest user priority is specified as local computing. For the remaining clusters, a DQN-based computation offloading algorithm



(a) Priority-driven user clustering



(b) optimal cluster number by SSE



(c) optimal cluster number by SC

Figure 4.4: Priority-driven user clustering and the optimal cluster number validation

is proposed for each user to learn the optimal computation offloading strategy and power allocation solutions distributively.

4.6.1 Parameters Setup

In the simulations, the location distribution of the IoT users is modelled by PCP with the parameter $\lambda_{Possion} = 50$. The small scale fading of the channels between users and the gateway is set as Rayleigh fading, and the path-loss parameter is $a = 2.5$. The input cluster number of priority-driven user clustering algorithm is 4, the result of user clustering and the validation of the optimal cluster number are shown in Fig. 4.4.

For DQN training, the DNN with three fully-connected feed forward hidden layers is used to approximate the Q-function, the neurons of each hidden layer is 256, 256 and 512, respectively. The activation function of the first two hidden layers are rectified linear units (ReLUs), and the final hidden layer employs tanh function as the activation function. The replay memory is set to have the capacity of 1000 recent transitions due to the dramatically changing environment, and the mini-batch size is $\mathcal{B} = 300$. The action exploration is following the ϵ-greedy policy with $\epsilon = 0.9$, and it's linearly decreasing with the iteration number. The possible channel gain values between the IoT user and the gateway are set as the channel gain set obtained in the clustering algorithm. For the user $u_i \in \mathcal{C}_h$, the channel gain h_i is selecting possible values from $\mathcal{G}_h$. The other simulation parameters of the proposed DQN-based computation offloading algorithm are shown in Table 4-A. The experiments are conducted by using Tensorflow for machine learning library.

Table 4-A: Simulation Parameters

Parameters	Values
γ, α	0.9, 0.5
B_m, σ	10^6Hz, $-174 + 10log_{10}B_m$
Channel gain set	G_h
Task queue set	$\mathcal{T} = [0, 19]$
Task size set	$\mathcal{M} = [10, 30]Kbits$
ν	600 $cycles/bit$
e_i^E	10^{-8} $Joule/CPU\ cycle$
e_i^L	10^{-7} $Joule/CPU\ cycle$
f, f_i	4GHz, 500 MHz
β	100

4.6.2 DQN-based Computation Offloading Scheme

In this section, the performance of the distributed computation offloading strategy learned by the DQN-based computation offloading algorithm is quantified. For comparison, the three benchmarks: local computing, edge computing and greedy scheme are also studied,

1. Local computing: the computation tasks are locally executed at the IoT user in each time slot whatever the size of the generated task is.

2. Edge computing: all the computation tasks generated at the IoT user are offloaded to the gateway and then processed there.

3. Greedy scheme: when the task queue is not empty and the remaining computation resource of the user is still enough, the IoT user decides to execute the task locally or offload it to the gateway while minimizing the system cost, that is, $min\{C^L_{i,j}, C^E_{i,j}\}$.

Through the centralized user clustering, the IoT users are grouped into different clusters corresponding to different user priorities. A typical IoT user in the cluster $\mathcal{C}_h$ (except the cluster with the highest and lowest user priority) is considered as an example to explore the computation offloading strategy and power allocation solution learned by the DQN-based algorithm **Algorithm 4.2**.

First, the convergence performance of the DQN-based computation offloading algorithm is validated. The possibility of task generation is set as $\tau^k = 0.5$. From Fig. 4.5, it is observed that the training loss is converged to a stable value by simulating the loss function (4.5.8). Based on the convergence performance of the proposed DQN-based computation offloading algorithm, the following simulation results are analyzed from the trained DQN running for 2×10^4 iterations.

With the trained DQN, the computation offloading scheme is tested through the proposed algorithm in **Algorithm 4.3**. The task queue is initialized as 15 and the terminal state is set as the task queue equalling to zero in the first time. The per-

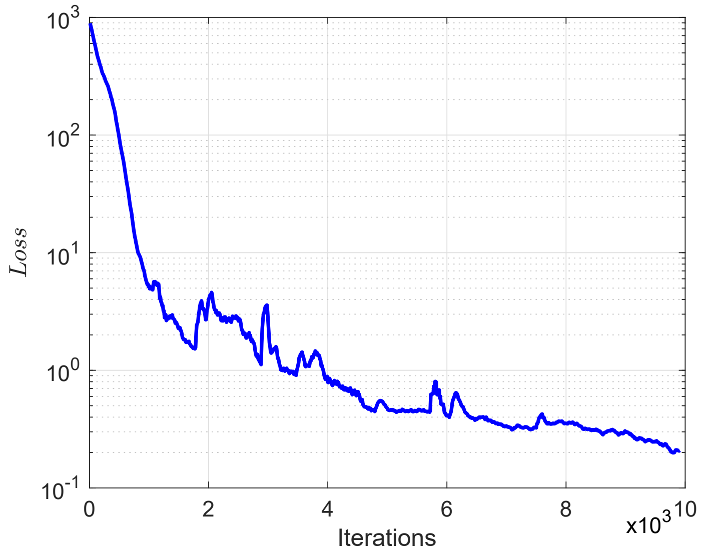


Figure 4.5: Convergence performance of the proposed DQN-based computation offloading algorithm measured by the loss function, the weight factor $\beta = 100$.

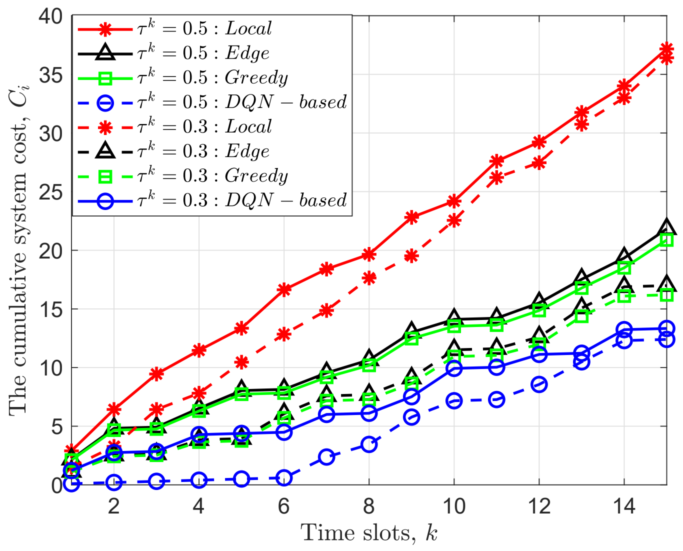


Figure 4.6: Performance comparison of the cumulative system cost C_i with the proposed DQN-based computation offloading scheme and the other three baseline schemes versus the time slots, $\beta = 100$.

formance of the cumulative system cost C_i over the experienced time slots is analyzed in Fig. 4.6. Moreover, it demonstrates the comparisons of the cumulative system cost among local computing, edge computing, greedy scheme and DQN-based computation offloading scheme over the time slots. The proposed DQN-based computation offloading

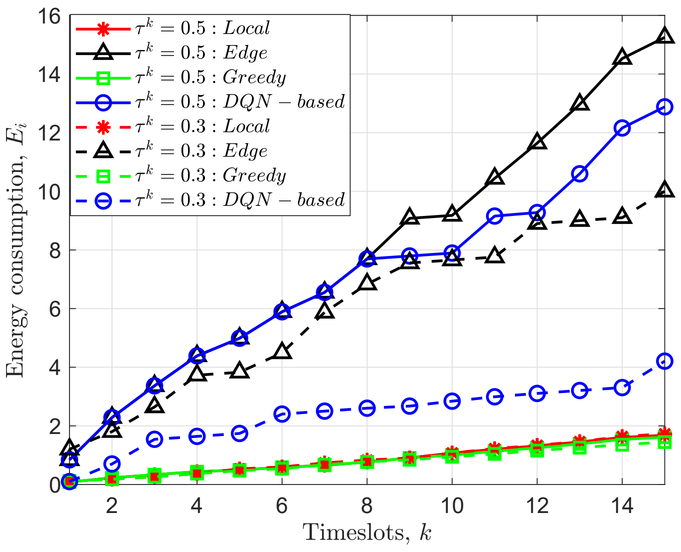


Figure 4.7: Performance comparison of cumulative energy consumption E_i with the proposed DQN-based task offloading scheme and the other three baselines versus the time slots. $\beta = 100$

scheme has lower cumulative system cost than the other three schemes. The reason is that the proposed DQN-based scheme can learn the computation offloading strategy to select the computing model and transmit power by minimizing the long-term system cost while the greedy scheme only considers the current minimum system cost. Here, the higher task generation probability, $\tau^k = 0.5$ means more computation tasks are generated and executed at the user compared to $\tau^k = 0.3$, so the system cost is higher.

Fig. 4.7 shows the comparisons of the energy consumption E_i among edge computing, local computing, greedy scheme and the proposed DQN-based computation offloading scheme over the time slots. It is shown that the edge computing has the highest energy consumption because the IoT user consumes transmit power to offload the computation task to the gateway. But for the local computing model, it consumes the least energy since all the energy consumption only comes from the task execution. In the proposed DQN-based scheme, the decisions on computation offloading are made by observing environment information. Compared to local computing, it has higher energy consumption because it may choose edge computing model according to the learned computation offloading strategy. Here, the greedy scheme has the lowest energy consumption since it

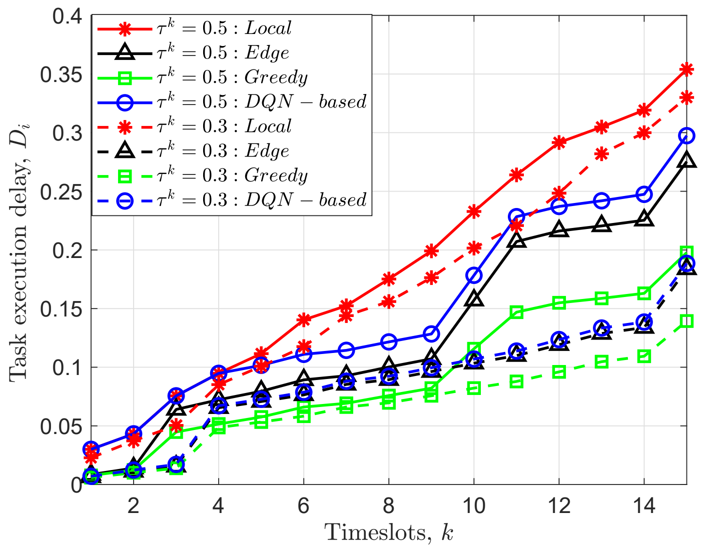


Figure 4.8: Performance comparison of cumulative task execution delay D_i with the proposed DQN-based task offloading scheme and the other three baselines versus the time slots. $\beta = 100$

considers the current minimum system cost, the energy consumption.

The performance of the task execution delay D_i among the three baseline schemes and the proposed DQN-based computation offloading scheme is shown in Fig. 4.8. Local computing has the highest execution delay since the computation ability of the IoT user is much weaker than the gateway. Specifically, the task with large task size takes long time to be executed locally. However, the computation task can be executed more quickly when it is offloaded to the gateway for execution. Similarly, the computation offloading strategy learned by the proposed DQN-based scheme includes both actions: local computing model and edge computing model, hence it has neutral performance of task execution delay.

Fig. 4.9 presents the performance comparisons over different computation capacity D_s of the gateway in terms of the average cost. It is demonstrated that the average cost is smaller when the gateway has larger computation capacity $D_s = 20$ GHz compared to $D_s = 4$ GHz. This is because more powerful computation capacity can provide faster task execution, that is, smaller task execution delay by executing tasks at the gateway.

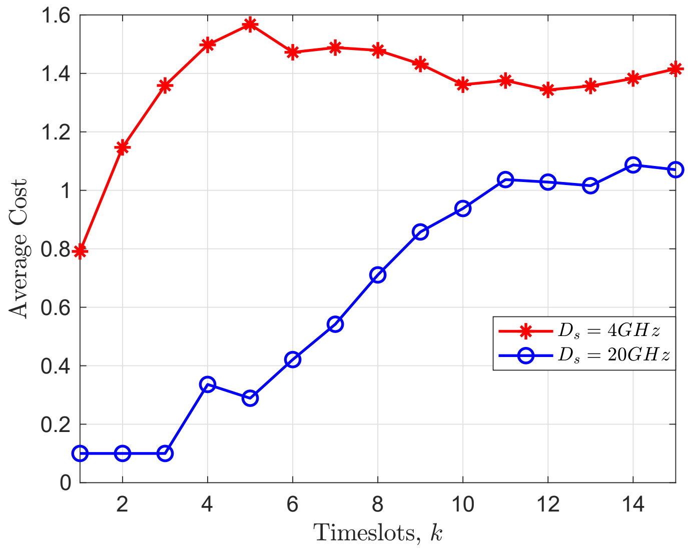


Figure 4.9: Performance comparison of average cost under different edge server computation capacity versus time slots k. $\beta = 50$

4.7 Summary

In this chapter, the joint problem of computation offloading scheme and resource allocation has been explored in IoT edge computing networks by considering the statistics information of environment, including the time-varying channel conditions, the dynamic task queue and the remaining computation resource of the IoT users via machine learning approaches. This problem was solved by two steps: centralized user clustering and distributed computation offloading strategy. The centralized user clustering was achieved by the proposed priority-driven clustering algorithm by defining user priorities as the clustering features. By running user clustering algorithm, the users were grouped into different clusters, where the clusters with the highest and lowest user priority were assigned as edge computing model and local computing model, respectively. For the users in the remaining clusters, a DQN-based computation offloading algorithm was proposed for each user to distributively learn the computation offloading strategy and power allocation solutions by using DQN framework with minimizing the long-term system cost. The simulations demonstrate the priority-driven user clustering algorithm and the optimal cluster number was validated. Moreover, the DQN-based computation offloading algorithm has been simulated, it has lower system cost compared to the other three baseline schemes and a neutral performance was obtained with separately considering energy consumption and task execution delay.

In this chapter, the computation offloading problem of users in IoT edge computing networks was investigated from the aspect of user clustering and distributed computation offloading scheme. Here, the distributed computation offloading scheme was learned only considering single user computation offloading scenario. In the next work, the multiuser computation offloading scenario where multiple users making their computation offloading decisions together will be discussed, which can be formulated as a stochastic game that makes it more suitable for practical applications.

Chapter 5

MARL for Multiuser Computation Offloading in IoT Networks

As discussed in Chapter 3 and 4, edge computing provides a front-end distributed computing archetype of centralized cloud computing with low latency, but where the edge server has finite resources to support all the users offloading their computing tasks. Therefore, the joint optimization problem of computation offloading and resource allocation in IoT edge computing network was studied in Chapter 4, where the computation offloading scheme adopting machine learning techniques has been proposed. A centralized user clustering was first investigated to pre-process the computation offloading decisions for users, and then a DQN-based computation offloading algorithm was developed for the user to learn efficient computing offloading strategies and resource allocation solutions distributively with considering single user computing offloading scenario. However, the challenges of multiple users offloading their computation tasks simultaneously need to be addressed since the uses are competing for spectrum, computation and radio access technologies resources during the offloading process. In this Chapter, the computation offloading mechanism of multiple selfish users is investigated in IoT edge computing network by formulating it as a non-cooperative stochastic game. Each user is a learning agent observing its local network environment to learn optimal decisions on either local computing or edge computing and optimal resource allocation solutions including transmit power level, radio access technology and sub-channel with a goal of minimizing long-term system cost while without knowing any information from the other users. Since the users' decisions are coupling at the gateway, the reward function of each user is defined by considering the aggregated effect of other users. Therefore, an MARL framework is developed to solve the stochastic game by proposing an IL-based MA-Q algorithm. The convergence of the proposed IL-based MA-Q algorithm is shown under different learning parameters by the simulations, and then this algorithm is tested to be feasible to solve the formulated problem and achieve distributed computation offloading decision making. Finally, compared with the benchmark algorithms, including the random and the iterative algorithm studied from [6], it has lower system cost performance. Moreover, it achieves around 80% performance of the centralized algorithm but has less complexity. The work presented in this chapter has been published in China Communications [124].

5.1 Motivation and Contributions

In IoT edge computing networks, due to its characteristics of distributed nature and random channel conditions, as well as limited spectrum resource, to find the computation offloading scheme with efficient resource allocation solutions for simultaneous multiuser computation offloading problem is full of challenge. To address these challenges, the authors in [6, 74, 105] formulated this problem as a computation offloading game using game theory and then the iterative algorithm was proposed and proved to reach a NE [6]. To the best of my knowledge, multiuser computation offloading problems have not been well investigated with MARL that can provide promising solutions for intelligent resource management with distributed learning. The major contributions of this chapter are presented as:

1. The joint optimization problem of multiuser computation offloading and resource allocation aiming to minimize the long-term system cost is investigated in IoT edge computing network by jointly selecting the transmit power level, the sub-channel and the radio access technology.

2. The multiuser computation offloading problem with resource allocation is formulated as a stochastic game, where each user becomes a self-interested learning agent to learn its computation offloading strategy and resource allocation solutions distributively under the dynamic wireless environment.

3. The MARL framework is exploited to develop intelligent computation offloading scheme and resource management solutions with distributed learning for users. Specifically, each user's reward function is defined by considering the aggregated effect of other users.

4. an IL-based MA-Q algorithm is proposed to explore the optimal computation offloading strategy with efficient resource allocation solutions for users. Here, each user runs an independent Q-learning algorithm with considering other users as part of the environment, and there is no information exchange among users.

5. Numerical results demonstrate that the proposed IL-based MA-Q computation offloading algorithm is superior to the random and the iterative algorithm proposed in [6]. Moreover, it achieves around 80% performance of the centralized algorithm but has less complexity, and it is more energy efficient without extra cost on channel estimation.

5.2 System Model

As shown in Fig. 5.1, considering a multiuser IoT edge computing network, which includes U single antenna IoT users and one single antenna edge server. Here, the IoT

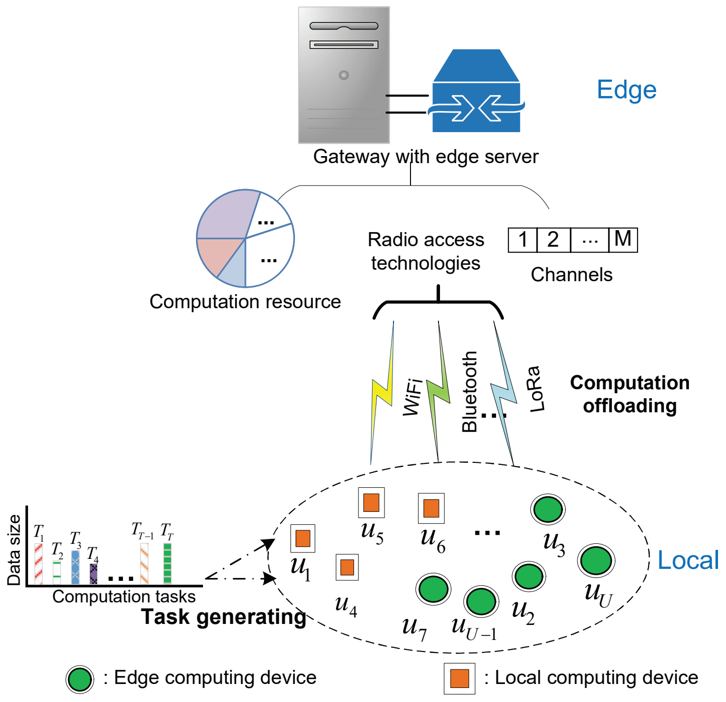


Figure 5.1: Multiuser computation offloading model in IoT networks.

users (e.g., sensors or mobile devices) can process some small computation tasks, while the edge device (e.g., gateway or the access point) can provide data processing for more intensive computation tasks from the IoT users with its higher computation capacity. Assuming that the IoT users are stationary and randomly distributed around the gateway, they collect data and process it with two possible ways: 1) local computing: process computation tasks locally at the IoT users; 2) edge computing: offload computation tasks to the gateway and process them at the edge server. The IoT users have limited computation capacity and the generated computation tasks are delay-constrained, so to improve the performance of data processing, they may choose to offload some compute-intensive computation tasks to the gateway by selecting the possible radio access technologies denoted by $\mathcal{RA} = \{RA_1, ..., RA_N\}$ with each radio access technology having M orthogonal sub-channels, denoted by $\mathcal{CH} = \{CH_1, ..., CH_M\}$. Noted that each radio access technology is operated on different radio frequencies. From Fig. 5.1, different computation tasks are continuously generated at each IoT user. The gateway in the considered

IoT edge computing network serves a set of IoT users $\mathcal{U} = \{u_1, ..., u_U\}$. Each IoT user u_i continuously generates computation tasks with the j^{th} task denoted as $T_{i,j}(d_{i,j}, D^{th}_{i,j})$, where $d_{i,j}$ is the task size and $D^{th}_{i,j}$ is the delay constraint of executing the task $T_{i,j}$ at user u_i. The generated task sequence at each user is indicated by $\{T_1, T_2, ...T_T\}$ shown in Fig. 5.1

This chapter is focused on dynamic multiuser computation offloading with efficient resource allocation in the considered IoT edge computing network by selecting transmit power levels, radio access technologies and sub-channels distributively without sharing information among users. Here, each IoT user only can observe its local information, such as the CSI between the IoT user and the gateway, and the feedback from the gateway. The system is assumed to operate on a time slotted structure with discretized time slots indexed by an integer $k \in \mathcal{K} = \{1, 2, ..., K\}$. At each time slot k, each IoT user makes its own decisions on computation offloading and resource allocation solutions distributively according to the observed local environment information.

5.2.1 Local Computing Model

In the considered multiuser IoT edge computing network, each IoT user is possible to execute its tasks locally or offload to the edge server and perform task execution there. Let the IoT user u_i choose to execute its task locally in time slot k. ν presents the number of CPU cycles required to process 1 bit data, and e^L_i denotes the computing energy consumption of each CPU cycle at the IoT user. Hence, the computing energy consumption of the task $T_{i,j}$ is $E^L_{i,j}$, and the time delay of local execution is $D^L_{i,j}$.

5.2.2 Edge Computing Model

Compared to the IoT user, the gateway has much more powerful computation capacity f and more stable power supply. Then the computation time delay of the task execution at the edge server is $D^E_{i,j}$, and the energy consumption by processing the offloaded computation task $T_{i,j}$ is $E^E_{i,j}$.

As mentioned before, the IoT users could offload their computation tasks to the gateway and which is able to support efficient computation task execution with its powerful computation ability. Assuming that the channels between each IoT user and the gateway follow Rayleigh fading distribution, each user can select one radio access technology from $\mathcal{RA}$ and one sub-channel from $\mathcal{CH}$ in the selected RA to offload the computation task $T_{i,j}$ to the gateway. The transmission rate can be presented as

$$R_i^{m,n} = B_{m,n}\log_2(1 + \frac{P_i^T h_i^{m,n}}{\sigma^2 + \sum\limits_{z=1, z\neq i}^{Z} \delta_z^{m,n} h^{z\to o} P_z^T}), \tag{5.2.1}$$

where P_i^T is the transmit power of the user u_i, $h_i^{m,n}$ is the channel gain of the radio access technology RA_n in sub-channel $CH_{m,n}$ [1], and $h^{z\to o}, z \in \mathcal{Z}$ is the channel gain from the user u_z to the gateway O. The noise is assumed to be AWGN with its power as σ^2 and P_z^T is the interference from the IoT user u_z that chooses the same channel $CH_{m,n}$, and $\delta_z^{m,n}$ indicates if the user u_z takes up the channel $CH_{m,n}$ or not, which is defined as

$$\delta_z^{m,n} = \begin{cases} 1, & u_z \; in \, CH_{m,n}, \\ 0, & otherwise. \end{cases} \tag{5.2.2}$$

Then the transmission delay of offloading the task $T_{i,j}$ of user u_i to the gateway is given by

$$D_{i,j}^T = d_{i,j}/R_i^{m,n}. \tag{5.2.3}$$

Moreover, the consumed energy for offloading the task $T_{i,j}$ is calculated as

$$E_{i,j}^T = P_i^T \cdot D_{i,j}^T. \tag{5.2.4}$$

Here, the observed SINR from user u_i to the gateway over channel $CH_{m,n}$ by offload-

[1]For simplicity, let $CH_{m,n}$ indicate the channel CH_m of radio access technology RA_n.

ing task $T_{i,j}$ is given by

$$\gamma_i^{m,n} = \frac{P_i^T h_i^{m,n}}{\sigma^2 + \sum\limits_{z=1,\, z\neq i}^{Z} \delta_z^{m,n} h^{z\to o} P_z^T}. \tag{5.2.5}$$

In this chapter, each IoT user adopts discrete transmit power control, with the transmit power values discreted as in a set $\mathcal{P}_i^T = \{P_1, ..., P_X\}$. At time slot k, each IoT user selects its transmit power $P_i^T = P_x, \forall x = \{1, ..., X\}$ from $\mathcal{P}_i^T$ when it chooses edge computing model; otherwise, $P_i^T = 0$ indicates the user chooses local computing model. Hence, the possible transmit power levels can be selected by the IoT user u_i are defined as a finite set,

$$\mathcal{P}_i^T = \{P_0, P_1, ..., P_X\}, P_x \neq 0, \forall x = \{1, ..., X\}, \forall i \in \mathcal{U}, \tag{5.2.6}$$

where $P_0 = 0$ indicates the user chooses to execute task locally. Similarly, the possible radio access technologies and the possible sub-channels under each radio access technology can be selected by the IoT user u_i are defined as the finite sets, respectively.

$$\begin{aligned} \mathcal{RA}_i &= \{RA_1, RA_2, ..., RA_N\}, \forall i \in \mathcal{U}, \\ \mathcal{CH}_i &= \{CH_1, CH_2, ..., CH_M\}, \forall i \in \mathcal{U}. \end{aligned} \tag{5.2.7}$$

The considered multiuser IoT edge computing network is assumed to operate on the discrete time horizon with each time slot equalization and non-overlapping. Moreover, assuming that the communication parameters keep unchanged during each time slot. The timeslot structure is shown in Fig. 5.2, with each time slot lasting K_s duration. During the time slot k, each IoT user executes its computation tasks according to the computation decisions made in the last time slot $k-1$, and then receives some feedbacks from the gateway, at last, it has to make decisions on computation offloading and resource allocation by the end of time slot k, i.e., $k = (k-1) + K_s$.

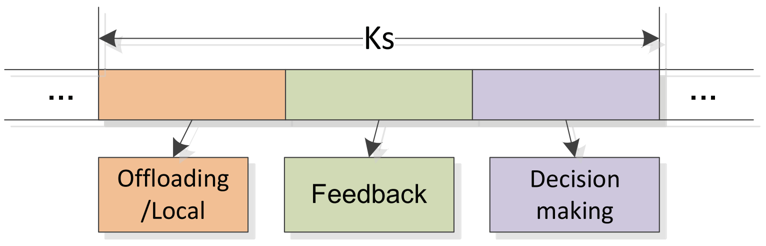


Figure 5.2: The timeslot structure of the multiuser computation offloading scheme.

5.3 Problem Formulation with Stochastic Game

In this section, the joint problem of computation offloading and resource allocation for multiuser IoT edge computing network is firstly formulated, and then the stochastic game is used to model the formulated problem with multiple self-interested users simultaneously choosing computing model, transmit power level, radio access technology and sub-channel.

5.3.1 Problem Formulation

In IoT edge computing networks, energy consumption and time delay are two main concerns for successful computation task execution. If the IoT user chooses to offload its computation task, it has to request spectrum and computation resource from the gateway, which in turn reduces the resource that other users can be allocated. Moreover, from (5.2.5), larger transmit power means higher transmission rate, smaller transmission delay, but causes more interference to other IoT users. Therefore, it is necessary to design a joint computation offloading scheme with efficient resource allocation for the IoT users. Here, the system cost, defined as the weighted sum of energy consumption and time delay, is considered as an index to evaluate the decisions on computation offloading and resource allocation made by the IoT user u_i, given by

$$C_i = E_i + \beta D_i, \tag{5.3.1}$$

where E_i is the energy consumption, and D_i is the time delay. Specifically, the system cost is considered as a negative reward function in the formulated problem, which can indicate what are good or bad decisions for the user.

At the time slot k, the IoT user u_i has two possible computing models, executing task locally or transmitting task to the gateway for execution. Based on the observed state from the environment, each IoT user selects its transmit power level, radio access technology and sub-channel. Then it receives a reward to evaluate the performance of the selected actions. Hence, the design of the reward function directly guides the learning process. In this chapter, the reward function r_i^k of each IoT user u_i is defined by considering the aggregated effect of other users, which is given by

$$r_i^k = \begin{cases} (a)\; C_{i,j}^L, \quad \gamma_i^{m,n} = 0, \\ (b)\; C_{i,j}^E + C_{i,j}^T, \qquad \gamma_i^{m,n} > \bar{\gamma}^n,\; W_{all} < \bar{W}, \\ (c)\; C_{i,j}^E + C_{i,j}^T + \omega, \;\; \gamma_i^{m,n} > \bar{\gamma}^n,\; W_{all} > \bar{W}, \\ (d)\; C_{i,j}^T + \varpi, \;\; \gamma_i^{m,n} < \bar{\gamma}^n, \end{cases} \tag{5.3.2}$$

where (a) indicates user u_i chooses local task execution and receives the reward $C_{i,j}^L$, which obviously indicates that the gateway will not receive any data transmission from this user, $\gamma_i^{m,n} = 0$. (b) indicates user u_i chooses to offload task to the gateway for execution and gets the reward $C_{i,j}^E + C_{i,j}^T$, which means the offloaded task is successfully received by the gateway, that is, $\gamma_i^{m,n} > \bar{\gamma}^n$, where $\bar{\gamma}^n$ denotes the threshold of the SINR. Also, the offloaded task from user u_i can be successfully executed, that is, $W_{all} < \bar{W}$, which means the computation capacity of the gateway is enough to support the data processing of all the offloaded tasks, W_{all} indicates the required computation capacity of all the offloaded tasks while $\bar{W}$ is the computation capacity of the gateway. (c) presents that user u_i chooses edge task execution and the offloaded task is successfully received by the gateway, but the computation capacity requirements from all the offloading users exceed the computation capacity of the gateway, that is, $W_{all} > \bar{W}$, so the obtained reward is $C_{i,j}^E + C_{i,j}^T + \omega$, where ω denotes the waiting cost received by the offloading users. (d) means the user u_i chooses edge task execution, but due to the interference from other users, the computation offloading process fails (e.g., the SINR is smaller than the SINR threshold). Therefore, it receives the reward $C_{i,j}^T + \varpi$, where ϖ denotes the received penalty because of failed computation offloading process. $C_{i,j}^L$, $C_{i,j}^E$, and $C_{i,j}^T$ present the cost of local task execution, edge task execution and computation offloading, which are calculated by

$$\begin{aligned} C_{i,j}^L &= E_{i,j}^L + \beta D_{i,j}^L, \\ C_{i,j}^E &= E_{i,j}^E + \beta D_{i,j}^E, \\ C_{i,j}^T &= E_{i,j}^T + \beta D_{i,j}^T. \end{aligned} \tag{5.3.3}$$

Noted that the instant reward of the IoT user u_i in any time slot k mainly relies on its observed information. 1) the available information: the taken actions including the transmit power level $P_i^{T,k}$, the radio access technology RA_i^k and the sub-channel CH_i^k, and it relates to the current channel gain $h_i^{m,n,k}$ and the remaining computation capacity of the edge server. 2) the unavailable information: the actions taken by the other IoT users in the same IoT network and the channel gains between them and the gateway.

Next, to consider the long-term system cost, which is given by

$$v_i^k = \sum_{\tau=0}^{+\infty} \lambda^\tau r_i^{k+\tau}, \tag{5.3.4}$$

where $\lambda \in [0, 1]$ is the discount factor. v_i^k presents the discounted sum of future rewards from the time slot k, which can be used to measure the taken action by IoT user u_i. And τ is the time slot index from the time slot k.

The action space of each IoT user contains the set of possible transmit power level $\mathcal{P}_i^T$, radio access technologies $\mathcal{RA}_i$ and sub-channels $\mathcal{CH}_i$, which can be denoted as $\mathcal{A}_i = \mathcal{P}_i^T \times \mathcal{RA}_i \times \mathcal{CH}_i$. Moreover, at any time slot k, the goal of each IoT user is to take an optimal action $a_i^*(k) = (P_i^{T*}, RA_i^*, CH_i^*) \in \mathcal{A}_i$ with maximizing the long-term reward in (5.3.4). Specifically, the reward function is defined by the system cost, which is a negative reward, so it converts to minimize the long-term reward here. Therefore, the optimization computation offloading problem of the IoT user u_i can be formulated as

$$a_i^{k*} = \underset{a_i \in \mathcal{A}_i}{\arg\min}\, v_i^k. \tag{5.3.5}$$

However, the design of multiuser computation offloading scheme for the considered IoT edge computing network consists of U sub-problems as discussed above, which corresponds to U IoT users. Moreover, each IoT user has no information of other IoT users and it is hard or even impossible to share their local information, like the taken actions and the received rewards, to each other. It is difficult to get a direct tool to solve this problem accurately, so a non-cooperative stochastic game is exploited to model this problem, and then an MARL framework is proposed as the solution.

5.3.2 Stochastic Game

In this section, a stochastic game is investigated to model the design of multiuser computation offloading strategies while minimizing the long-term system cost. In the considered IoT edge computing network, each IoT user is considered as a player to take actions including computation offloading decisions and resource allocation solutions to maximize its own payoff by interacting with the environment. This is an n-player game, in which multiple self-interested and selfish users learn their own computation offloading strategies without any cooperation with the other IoT users. Each player observes its local environment state $s_i^k \in \mathcal{S}_i$, then independently takes the action $a_i^k \in \mathcal{A}_i$. Consequently, each player receives a reward $r_i^k = r_i(s_i^k, a_1^k, ...a_U^k)$, and observes a new state $s_i^{k+1} \in \mathcal{S}_i$ depending on the actions of all the involved players.

A stochastic game is the generalization of MDP in the multi-agent case, also named as a Markov game, which is denoted by a tuple $< \mathcal{S}, U, \mathcal{A}, \mathbf{P}, \mathcal{R} >$.

- $\mathcal{S}$ is the environment states that include the states of each player, and $\mathcal{S} = \mathcal{S}_1 \times \mathcal{S}_2 \cdots \times \mathcal{S}_U$;
- U is the player number;

- $\mathcal{A}$ is the joint action set $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \cdots \times \mathcal{A}_U$;
- $\mathbf{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \in [0, 1]$ is the state transition probability function;
- $\mathcal{R} = \{r_1, \cdots r_U\}$ contains all the reward functions of the players.

5.3.3 Computation Offloading Game Formulation

Based on the definition of the stochastic game, the considered multiuser computation offloading problem can be formulated as a stochastic game with each item presented as follows.

5.3.3.1 Action

In the considered IoT edge computing network, each IoT user u_i is considered as an intelligent agent, at any time slot k, it takes an action including selecting transmit power level $P_i^{T,k}$, radio access technology RA_i^k and sub-channel CH_i^k to complete task execution. $a_i^k \in \mathcal{A}_i = \mathcal{P}_i^T \times \mathcal{RA}_i \times \mathcal{CH}_i$ is denoted as the IoT user u_i's action at time slot k. Hence, the action space of the computation offloading game is $\mathcal{A} = \prod_{i \in \mathcal{U}} \mathcal{A}_i$.

5.3.3.2 State

There is no cooperation among the competitive IoT users, so the environment state is defined based on each IoT user's local observations. At time slot k, the state observed by the IoT user u_i is given by

$$s_i^k = (\mathcal{L}_i^k, \mathcal{I}_i^k, \mathcal{J}_i^k), \tag{5.3.6}$$

where $\mathcal{L}_i^k \in \{0, 1\}$ indicates the user u_i chooses local computing or edge computing model, that is, whether the transmit power $P_i^{T,k}$ of the user u_i is equal to zero or not, denoted as

$$\mathcal{L}_i(k) = \begin{cases} 0, \ if \ P_i^{T,k} = P_0 = 0, \\ 1, \ otherwise. \end{cases} \tag{5.3.7}$$

$\mathcal{I}_i^k \in \{0,1\}$ indicates whether the u_i's computation offloading can be recognized by the gateway, that is, the received SINR $\gamma_i^{m,n,k}$ of user u_i is above or below the SINR threshold $\bar{\gamma}^n$,

$$\mathcal{I}_i^k = \begin{cases} 1, \; if \; \gamma_i^{m,n,k}(a_i^k, \mathbf{a}_{-i}^k) > \bar{\gamma}^n, \\ 0, \; otherwise, \end{cases} \tag{5.3.8}$$

where $\mathbf{a}_{-i}^k = (a_1^k, ..., a_{i-1}^k, a_{i+1}^k, ..., a_U^k) \in \mathcal{A}_{-i} = \mathcal{A}_1 \times ... \times \mathcal{A}_{i-1} \times \mathcal{A}_{i+1} \times ... \times \mathcal{A}_U$ indicates the action vector of the other IoT users.

Moreover, $\mathcal{J}_i^k \in \{0,1\}$ denotes the broadcast information from the gateway that presents if the gateway's computation capacity is enough to support the offloaded computation tasks from users at time slot k, given by

$$\mathcal{J}_i^k = \begin{cases} 1, \; if \; W_{all} \leq \bar{W}, \\ 0, \; otherwise. \end{cases} \tag{5.3.9}$$

5.3.3.3 Reward

The reward $r_i^k(s_i^k, a_i^k, \mathbf{a}_{-i}^k)$ of IoT user u_i in state s_i^k presents the immediate return by user u_i taking action a_i^k while the other IoT users taking actions $\mathbf{a}_{-i}^k$ at time slot k. It's rewritten as

$$r_i^k(s_i^k, a_i^k, \mathbf{a}_{-i}^k) = \begin{cases} C_{i,j}^L, \quad if \; \mathcal{L}_i^k = 0, \\ C_{i,j}^E + C_{i,j}^T, \quad if \; (\mathcal{L}_i^k, \mathcal{I}_i^k, \mathcal{J}_i^k) = (1,1,1), \\ C_{i,j}^E + C_{i,j}^T + \omega, \; if \; (\mathcal{L}_i^k, \mathcal{I}_i^k, \mathcal{J}_i^k) = (1,1,0), \\ C_{i,j}^T + \varpi, \; if \; (\mathcal{L}_i^k, \mathcal{I}_i^k) = (1,0). \end{cases} \tag{5.3.10}$$

Here, the reward function of user u_i is defined by considering the aggregated effect of other users' actions on user u_i. Since only the users that choose edge computing model may have coupling actions including sub-channels selection and computation resource competition at the gateway, the reward function $r(s_i^k, a_i^k, \mathbf{a}_{-i}^k)$ (hereinafter shorten $r_i^k(s_i^k, a_i^k, \mathbf{a}_{-i}^k)$ as $r(s_i^k, a_i^k, \mathbf{a}_{-i}^k)$) depends on the actions taken by the users choosing edge computing model. This means that the user u_i requesting for the computation resource from the gateway is affected by the other users that have the same requests at the same time slot k, also, its channel access is affected by the users competing for the same channel. For these considerations, two different penalties, ω and ϖ, are defined in the reward function to mitigate the completely unknown information from the other offloading users.

Recall that a policy, $\pi_i(s_i, a_i)$, is a mapping from each state, $s_i \in \mathcal{S}_i$ and $a_i \in \mathcal{A}_i$ to a probability $\pi_i(s_i, a_i) = \Pr(a_t = a \mid s_t = s) \in [0, 1]$. It gives the probability of taking action a_i in state s_i. Specifically, for IoT user u_i in the state s_i, it has mixed strategy as $\pi_i(s_i) = \{\pi_i(s_i, a_i) \mid a_i \in \mathcal{A}_i\}$. Hence, in a stochastic game, a joint strategy for U agents is defined as a strategy vector $\boldsymbol{\pi} = (\pi_1(s_1), \pi_2(s_2), ..., \pi_U(s_U))$ with each strategy belonging to each agent. Based on the probabilistic policies, we formulate the optimization goal of each IoT user u_i in (5.3.4) into its expected discounted form as

$$V_i(s_i, \boldsymbol{\pi}) = \mathbb{E}\left[\sum_{\tau=0}^{+\infty} \lambda^{\tau} r_i(k+\tau) \middle| s_i^k = s_i, \boldsymbol{\pi}\right], \tag{5.3.11}$$

where $\mathbb{E}[\cdot]$ is the expectation operation, which calculates the state transition under strategy $\boldsymbol{\pi}$ in state s_i. The state transition from the state s_i^k to the new state s_i^{k+1} is determined by the joint strategy of all the IoT users. Moreover, in the non-cooperative game, at each time slot k, each IoT user u_i chooses its strategy $\pi_i(s_i^k)$ independently in state s_i^k to maximize its discounted reward $V_i(s_i, \boldsymbol{\pi})$, and then it receives its current individual reward based on the joint strategy $\boldsymbol{\pi}$. Here, the goal of each IoT user is to learn the optimal strategy π_i^* from any state $s_i \in \mathcal{S}_i$, i.e., the optimal strategies of other IoT users are learned as $\boldsymbol{\pi}_{-i}^* = (\pi_1^*, ..., \pi_{i-1}^*, \pi_{i+1}^*, ..., \pi_U^*)$, and the expected reward is reformulated as

$$\begin{aligned} &V_i(s_i, \pi_i, \boldsymbol{\pi}_{-i}) = \mathbb{E}\left[\sum_{\tau=0}^{+\infty} \lambda^{\tau} r_i(s_i^{k+\tau}, \pi_i(s_i^{k+\tau}), \boldsymbol{\pi}_{-i}(s_i^{k+\tau})) \middle| s_i^k = s_i\right], \\ &\boldsymbol{\pi}_{-i}(s_i^k) = (\pi_1(s_1^k), ..., \pi_{i-1}(s_{i-1}^k), \pi_{i+1}(s_{i+1}^k), ...\pi_U(s_U^k)). \end{aligned} \tag{5.3.12}$$

Hence, the solution of the multiuser computation offloading game may reach a Nash equilibrium (NE) at one stage, which is defined by **Definition 5.1**.

Definition 5.1. *A Nash equilibrium is a set of U optimal strategies $(\pi_1^*, ..., \pi_U^*)$, in which no IoT user can receive any lower reward by only changing their own strategies. That is, for each IoT user $u_i \in \mathcal{U}$ in each state $s_i \in \mathcal{S}_i$,*

$$V_i(s_i, \pi_i^*, \boldsymbol{\pi}_{-i}^*) \leq V_i(s_i, \pi_i, \boldsymbol{\pi}_{-i}^*),\ \forall \pi_i \in \Pi_i, \tag{5.3.13}$$

where Π_i is the set of possible strategies can be taken by IoT user u_i.

Definition 5.2. *1) Every finite stochastic game has a Nash equilibrium if it has a finite number of players, U, finite action set, $\mathcal{A}$, and set of states, $\mathcal{S}$.*
2) A game with infinite stages if the total payoff is the discounted sum, $V_i(s_i, \pi_i, \boldsymbol{\pi}_{-i})$.

This means, there always exists a NE in our formulated computation offloading game. In a NE, each IoT user has obtained its optimal strategy, no IoT user can get any better strategies by changing only their own strategies. Therefore, in this chapter, each IoT user u_i is aiming to find an NE strategy for any state s_i.

5.4 MARL based Computation Offloading Algorithm

In this section, the multiuser computation offloading in IoT edge computing network is formulated by the MARL framework. Then a MA-Q learning based computation offloading algorithm is proposed to address the formulated multiuser computation offloading game.

5.4.1 MARL Framework for Multiuser Computation Offloading

Fig. 5.3 illustrates an MARL framework for a multiuser computation offloading problem in the IoT edge computing network. Here, in time slot k, each IoT user u_i at state s_i^k takes the action $a_i^k \in \pi_i(s_i)$, and then the environment is changed to a new joint state $S^{k+1} = \{s_1^{k+1}, ..., s_i^{k+1}, ..., s_U^{k+1}\}$ based on the joint action $A^k = \{a_1^k, ..., a_i^k, ..., a_U^k\}$ taken

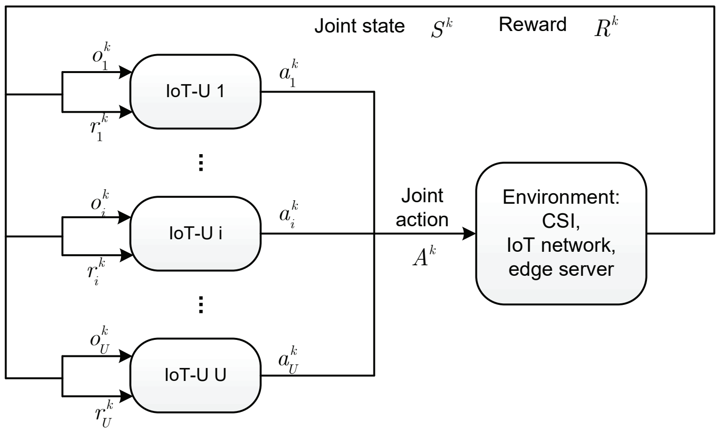


Figure 5.3: The MARL framework for multiuser computation offloading in IoT networks.

by all the IoT users. Finally, each IoT user observes its local information o_i^k as the new state s_i^{k+1} and gets its own reward r_i^k. Since the future state S^{k+1} only depends upon the present state S^k and the taken action A^k, this dynamic MARL process has Markov property such that it can be formulated into a stochastic game, i.e., Markov game. Specifically, the stochastic game with a single player is modelled as an MDP, moreover, the decision problem faced by a player in a stochastic game when all the other players choose a fixed profile of stationary strategies is equivalent to an MDP [125].

In the non-cooperative game, each IoT user u_i chooses the strategy $\pi_i(s_i)$ independently to maximize its total expected discounted reward, defined in the value function, from (5.3.12). The value function can be decomposed into two parts as shown in the Bellman equation:

$$V_i(s_i, \pi_i, \boldsymbol{\pi}_{-i}) = \mathbb{E}[r_i(s_i, \pi_i(s_i), \boldsymbol{\pi}_{-i}(s_i)) + \sum_{\tau=1}^{+\infty} \lambda^\tau r_i(s_i', \pi_i(s_i'), \boldsymbol{\pi}_{-i}(s_i')) \mid s_i^k = s_i], \tag{5.4.1}$$

where for simplicity, let $s_i^k = s_i$ and $s_i^{k+1} = s_i'$, so the Bellman equation is formulated as

$$\begin{aligned} V_i(s_i, \pi_i, \boldsymbol{\pi}_{-i}) &= \mathbb{E}[r_i(s_i, \pi_i(s_i), \boldsymbol{\pi}_{-i}(s_i)) + \lambda V_i(s_i', \pi_i, \boldsymbol{\pi}_{-i}) \mid s_i^k = s_i] \\ &= \mathbb{E}[r_i(s_i, \pi_i(s_i), \boldsymbol{\pi}_{-i}(s_i))] + \lambda \mathbb{E}[V_i(s_i', \pi_i, \boldsymbol{\pi}_{-i})], \end{aligned} \tag{5.4.2}$$

where the expectation of the immediate reward is defined as

$$\mathbb{E}[r_i(s_i, \pi_i(s_i), \boldsymbol{\pi}_{-i}(s_i))] = \sum_{(a_i, \mathbf{a}_{-i}) \in \mathcal{A}_i} r_i(s_i, a_i, \mathbf{a}_{-i}) \prod_{z \in U} \pi_z(s_z, a_z), \tag{5.4.3}$$

where $\pi_z(s_z, a_z)$ denotes the probability of the IoT user u_z choosing action a_z in state s_z. Moreover, the expectation of the discounted value of successive state is calculated with the state transition function

$$\mathbb{E}[V_i(s_i', \pi_i, \boldsymbol{\pi}_{-i})] = \sum_{s_i' \in \mathcal{S}_i} \mathbf{P}_{s_i s_i'}(\pi_i(s_i), \boldsymbol{\pi}_{-i}(s_i)) V_i(s_i', \pi_i, \boldsymbol{\pi}_{-i}). \tag{5.4.4}$$

Therefore, at the state $s_i^k = s_i$, the IoT user takes action a_i^k and then gets the expectation value of the cumulative return under policy $\boldsymbol{\pi}$ defined as the state-action value function

$$\begin{aligned} Q_i^{\boldsymbol{\pi}}(s_i, a_i) &= \mathbb{E}[\sum_{\tau=0}^{+\infty} \lambda^\tau r_i(s_i, a_i, \boldsymbol{\pi}_{-i}(s_i)) \mid s_i^k = s_i, a_i^k = a_i] \\ &= \mathbb{E}[r_i(s_i, a_i, \boldsymbol{\pi}_{-i}(s_i)) + \lambda Q_i^{\pi}(s_i', a_i') \mid s_i^k = s_i, a_i^k = a_i], \end{aligned} \tag{5.4.5}$$

similarly, the first part of the Q-value function presents the current return of the IoT user by taking action a_i in state s_i, which is defined as

$$\mathbb{E}[r_i(s_i, a_i, \boldsymbol{\pi}_{-i}(s_i))] = \sum_{\mathbf{a}_{-i} \in \mathcal{A}_{-i}} r_i(s_i, a_i, \mathbf{a}_{-i}) \prod_{z \in U \setminus \{i\}} \pi_z(s_z, a_z). \tag{5.4.6}$$

Hence, (5.4.5) is reformulated as

$$\begin{aligned} Q_i^{\pi}(s_i, a_i) &= \mathbb{E}[r_i(s_i, a_i, \boldsymbol{\pi}_{-i}(s_i))] + \lambda \sum_{s_i' \in \mathcal{S}_i} \mathbf{P}_{s_i s_i'}(a_i, \boldsymbol{\pi}_{-i}(s_i)) V_i(s_i', \pi_i, \boldsymbol{\pi}_{-i}) \\ &= \mathbb{E}[r_i(s_i, a_i, \boldsymbol{\pi}_{-i}(s_i))] + \lambda \sum_{s_i' \in \mathcal{S}_i} \mathbf{P}_{s_i s_i'}(a_i, \boldsymbol{\pi}_{-i}(s_i)) \sum_{a_i'} \pi_i(s_i', a_i') Q_i^{\pi}(s_i', a_i'). \end{aligned} \tag{5.4.7}$$

As discussed in Section III, there always exists an NE in the formulated computation offloading game. Hence, the optimal strategy satisfies the Bellman's optimality equation, given as

$$V_i(s_i,\pi_i^*,\boldsymbol{\pi}_{-i}^*) = \max_{a_i\in\mathcal{A}_i}\{\mathbb{E}[r_i(s_i,a_i,\boldsymbol{\pi}_{-i}^*(s_i))] + \lambda\sum_{s_i'\in\mathcal{S}_i}\mathbf{P}_{s_is_i'}(a_i,\boldsymbol{\pi}_{-i}^*(s_i))V_i(s_i',\pi_i^*,\boldsymbol{\pi}_{-i}^*)\}. \tag{5.4.8}$$

Similarly, the optimal Q-value function Q^* is the maximal action-value function over all the possible strategies, that is, when all the IoT users follow the NE strategies given by

$$Q_i{}^*(s_i,a_i) = \mathbb{E}[r_i(s_i,a_i,\boldsymbol{\pi}_{-i}^*(s_i))] + \lambda\sum_{s_i'\in\mathcal{S}_i}\mathbf{P}_{s_is_i'}(a_i,\boldsymbol{\pi}_{-i}^*(s_i))V_i(s_i',\pi_i^*,\boldsymbol{\pi}_{-i}^*). \tag{5.4.9}$$

By combining (5.4.8) and (5.4.9), the optimal Q-function is reformulated as

$$Q_i{}^*(s_i,a_i) = \mathbb{E}[r_i(s_i,a_i,\boldsymbol{\pi}_{-i}^*(s_i))] + \lambda\sum_{s_i'\in\mathcal{S}_i}\mathbf{P}_{s_is_i'}(a_i,\boldsymbol{\pi}_{-i}^*(s_i))\max_{a_i'\in\mathcal{A}_i}Q_i^*(s_i',a_i'). \tag{5.4.10}$$

From (5.4.10), the optimal strategy means each IoT user u_i chooses the optimal action which maximizes the corresponding Q-value function for the current state s_i, which forms an optimal policy over each time step. Moreover, the optimal Q-function is determined by the joint policy of all the users $\boldsymbol{\pi}$, which makes it difficult to obtain the optimal strategy. In this chapter, independent learning is used to solve the formulated computation offloading game. Each IoT user is considered as an independent learner to learn its individual strategy without sharing information with other IoT users, that is, for the IoT user u_i, the other users are just forming of one part of the environment. Actually, it is more practical because it is hard for the IoT user to be aware of the existence of the other IoT users, or to reduce complexity, it may choose to ignore the action information from the other IoT users.

5.4.2 IL-based MA-Q learning Algorithm

In this section, Q-learning is applied to solve the independent MDP problems, and an IL based MA-Q learning algorithm is proposed to solve the joint problem of multiuser computation offloading and resource allocation in the IoT edge computing network. In the proposed algorithm, each IoT user runs an independent Q-learning algorithm and simultaneously learns an individual optimal strategy for their own MDPs. Specifically, the selection of an optimal action depends on the Q-function, by following (5.4.10), the optimal Q-function is defined as the optimal expected value of state s_i when taking action a_i and then proceeding optimally. Since the state transition probability function $\mathbf{P}_{s_i s_i'}$ and the reward function $r_{s_i s_i'}$ are hard or even impossible to be obtained in practice, the mean return with multiple sampling is used to approximately calculate the expected cumulative reward. This is achieved by using Monte-Carlo (MC) Learning method, with sampling the same Q-function $Q_i(s_i, a_i)$ over different strategies. However, MC learning is complicated by calculating mean return with sampling complete episodes, so Temporal Difference (TD) learning is used to recursively update Q-value function with learning the estimates on the basis of the old values, which is presented as

$$Q_i'(s_i, a_i) \leftarrow Q_i(s_i, a_i) + \alpha(r_i' + \lambda \min_{a_i' \in \mathcal{A}_i} Q(s_i', a_i') - Q_i(s_i, a_i)), \tag{5.4.11}$$

where $r_i' + \lambda \min\limits_{a_i' \in \mathcal{A}_i} Q(s_i', a_i')$ indicates the optimal cumulative returns at time slot $k+1$, which is called TD target. α is the learning rate ($0 < \alpha \leq 1$), for ensuring the convergence of Q-learning, the learning rate α_k is set as

$$\alpha_k = \alpha_{k-1} * (\frac{\alpha_{end}}{\alpha_{ini}})^{\frac{1}{episodes}}, \tag{5.4.12}$$

where $\alpha_{ini}, \alpha_{end}$ are the given initial and last values of α, respectively, and *episodes* is the maximum iterations of the learning algorithm.

In this chapter, an IL-based MA-Q learning algorithm is proposed for multiple users to independently learn their optimal computation offloading strategies for the formulated computation offloading game. In the proposed algorithm, each IoT user runs a Q-learning procedure independently, and it only maintains its own Q-table. The detailed process of the proposed algorithm is shown in **Algorithm 5.1**.

Algorithm 5.1: *IL-based MA-Q learning computation offloading algorithm*

```
Initialization:
    Initialize parameters: discount factor $\lambda$, learning rate parameters $\alpha_{ini}, \alpha_{end}$, exploration rate $\epsilon$.
    Set $k = 0$.
 1: for $i$ to $U$ do
 2:     Initialize action-value function $Q_i^k(s_i, a_i)$
 3:     Initialize the resource allocation strategy $\pi_i^k(s_i, a_i)$
 4:     Initialize the state $s_i^k$
 5: end for
Learning:
 6: while $k \le K$ do
 7:     for $i$ to $U$ do
 8:         Choose an action $a_i$ according to the strategy $\pi_i(s_i)$
 9:         Measure the received SINR $\gamma_i^{m,n}$ at the receiver and the computation capacity
            of the gateway by identifying the transmit power, radio access technology and
            sub-channel
10:         Observe the current state $s_i^{k+1}$
11:         Obtain a reward $r_i^k$ according to the measured information
12:         Update $Q_i^{k+1}(s_i, a_i)$ by (5.4.11)
13:         Update the strategy $\pi_i^{k+1}(s_i, a_i)$ according to $\epsilon$-greedy method
14:         Set $k = k + 1$
15:     end for
16: end while
```

In the proposed learning algorithm, the ϵ-greedy method is adopted as the strategy of action selection, which focuses on solving the important problem of reinforcement learning, the trade-off between exploration and exploitation. This gives a guide that the agent reinforces the best decision given information or explores new actions to gather more information. With the ϵ-greedy method, the agent selects the optimal action corresponding to the largest Q-function with probability $1 - \epsilon$, and chooses a random action with probability $\epsilon \in [0, 1]$.

Table 5-A: Simulation Parameters

Learning parameters	
$\alpha_{ini}, \alpha_{end}$	0.9, 0.001
λ, β	0.9, 5
Channel parameters	
$B^L_{m,n}, \sigma^L$	$1.25 * 10^5$Hz, $-174 + 10log10B^L_{m,n}$
$B^W_{m,n}, \sigma^W$	$5 * 10^6$Hz, $-160 + 10log10B^W_{m,n}$
$\bar{\gamma}^L, \bar{\gamma}^W$	$-15dB, 10dB$
CH^L, CH^W	$10, 13$
$\mathcal{P}^L, \mathcal{P}^W$	$[0.05, 0.3]W$, $[0.1, 1]W$
Computation parameters	
$d_{i,j}$	$[10K, 4M]bits$
ν	500 $cycles/bit$
e^E, e^L_i	$10^{-7}, 10^{-8}$ $W/CPU\ cycle$
f, f_i	10GHz, [500MHz, 1GHz]

5.5 Simulation Results

In this section, the performance of the proposed IL-based MA-Q learning algorithm for multiuser computation offloading with the resource allocation problem is presented in the simulations. In the considered IoT network, each IoT user runs a Q-table and independently interacts with the environment to learn its own optimal computation offloading strategy and resource allocation solutions. The distances between the IoT users and the gateway are following the normal distribution with $\mu = 1000$, $\xi = 3$, that is, the users are located in a circle with its radius $r_c = 1km$. The small scale fading between the users and the gateway is set as Rayleigh fading, and the path-loss parameter is set as $a = 2.5$. Two radio access technologies, WiFi and LoRa, are considered as an example to illustrate the proposed algorithm. The detailed simulation parameters are given in Table 5-A. The experiments are conducted by using Tensorflow for machine learning library.

Here, the maximum iteration episode for all the simulation experiments is set as $episodes = 10000$. To verify the proposed IL-based MA-Q learning algorithm, without loss of generality, an IoT user u_1 is considered as an example. As shown in Fig. 5.4, it shows the convergence performance of the proposed algorithm under different exploration

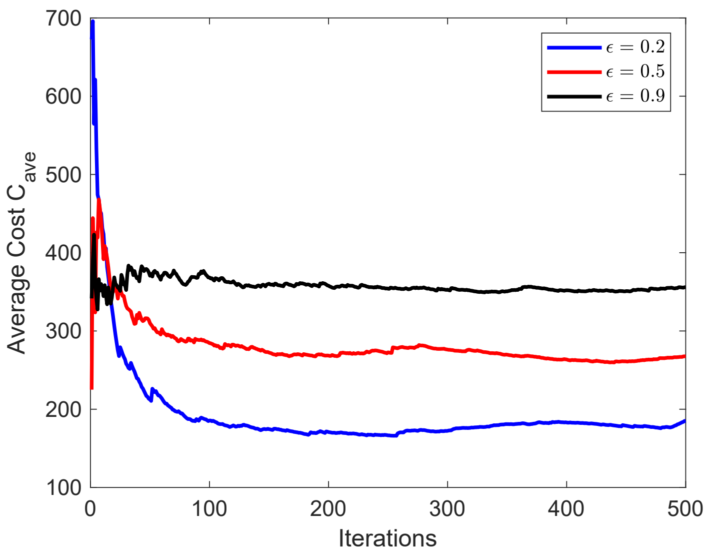


Figure 5.4: Convergence performance of the proposed IL-based MA-Q learning algorithm measured by the average Cost C_{ave}, achieved by u_1, $Ag = 30$.

rates ϵ. The average cost per time slot is converged with the iterations increasing, which indicates the Q-table of user u_1 has been trained stably. Moreover, it is observed that the larger $\epsilon = 0.9$ causes the worse average cost, that is, the user explores too many random actions instead of exploiting the optimal action. From Fig. 5.4, the user has to pay more attention to exploiting its optimal action in the considered scenario.

Fig. 5.5 shows the convergence performance comparison over different number of users. It is observed that the average cost, C_{ave}, per time slot of user u_1 is higher with more users in the considered IoT network. This is because with more users, it is harder for each user to access the gateway due to the limited number of channels. Furthermore, with the trained Q-tables, the IoT users can make their computation offloading decisions simultaneously. Considering there are 30 IoT users in the IoT network with 23 available channels including 10 LoRa channels and 13 WiFi channels. From Fig. 5.6, 20 users choose to offload their computation tasks while the other 10 users choose to execute their computation tasks locally. Here, 8 users offload their computation tasks using the LoRa channels while the others access to the gateway with WiFi channels. It is noted that the IoT users can access the channels reasonably without collisions.

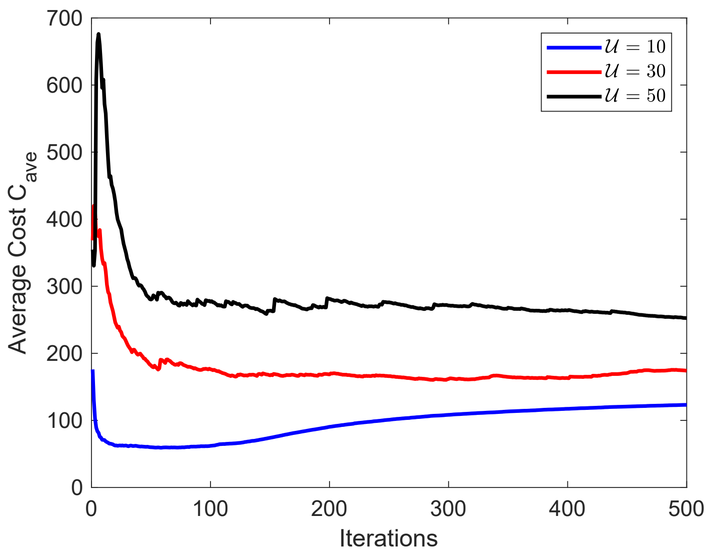


Figure 5.5: Performance comparison with different number of users $\mathcal{U}$ measured by the average Cost C_{ave}, achieved by u_1, $\epsilon = 0.2$.

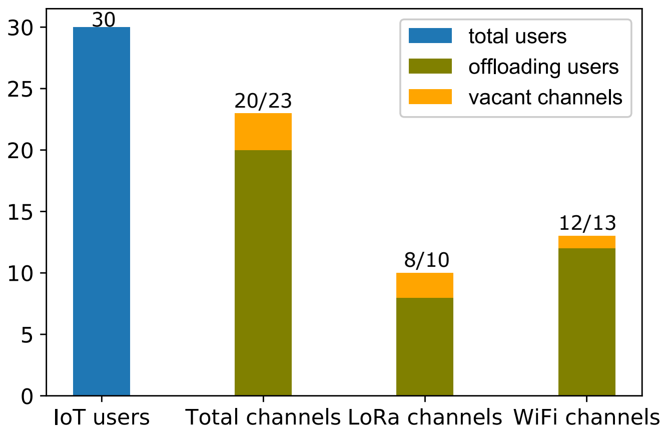


Figure 5.6: Access channels allocation with LoRa and WiFi access technologies, $Ag = 30, \epsilon = 0.2$.

Moreover, the average cost per time slot per IoT user under different computation offloading algorithms is investigated: the proposed IL-based MA-Q, Centralized, Iterative presented in **Algorithm 5.2** and Random. Here, Centralized, Iterative and Random algorithm are proposed as three benchmark algorithms for the proposed algorithm.

1. IL-based MA-Q computation offloading: each user independently runs its own Q-

Algorithm 5.2: *Iterative computation offloading algorithm*

```
Initialization:
    Set $k = 0$
1: for $i$ to $U$ do
2:     Initialize the strategy selection probability of user $u_i$ as $\frac{1}{U+1} \in \mathbf{w}_i^k = (\frac{1}{U+1}, ..., \frac{1}{U+1})$
3: end for
Learning:
4: for $k$ to $K$ do
5:     for $i$ to $U$ do
6:         User $u_i$ selects an offloading strategy $\pi_i^k$ according to $\mathbf{w}_i^k$
7:         Measure the received SINR $\gamma_i^{m,n}$ at the receiver and the computation capacity of the gateway by identifying the transmit power, radio access technology and sub-channel. Evaluate a reward $r_i^k$.
8:         Update $\mathbf{w}_i^k$ by $\mathbf{w}_i^{k+1} = \mathbf{w}_i^k + bo_i^k(\mathbf{e}_{\pi_i^k} - \mathbf{w}_i^k)$ where $0<b<1$ is the learning step size, $\mathbf{e}_{\pi_i^k}$ is $(U+1)$-dimensional unit vector with the $\pi_i^k$-th element equal to 1, $o_i^k = 1 - \zeta_i r_i^k$, $\zeta_i$ is a scaling factor and $\zeta_i \leq \frac{1}{\max_k r_i^k}$
9:         Until all the users do not change their offloading strategies.
10:    end for
11: end for
```

table to learn the efficient computation offloading strategy by interacting with the environment.

2. Centralized computation offloading: first, the gateway makes channel estimation over each user to obtain their channel information and computation task sizes, then it allocates users for local computing, or offloading computation tasks with LoRa channels or WiFi channels.

3. Iterative computation offloading: it is based on the multi-agent stochastic learning algorithm proposed in [6] to solve the multiuser computation offloading problem, which initializes a strategy selection probability vector $\mathbf{w}_i^k$ as a uniform distribution for each user, then performs a loop by *a*) each user updates its computation strategy according to $\mathbf{w}_i^k$; *b*) each user measures its instantaneous payoff; *c*) each user updates its $\mathbf{w}_i^k$ with an updating rule. The loop ends until all the users do not adjust their respective computation offloading strategies. The detailed procedure is illustrated in **Algorithm 5.2**.

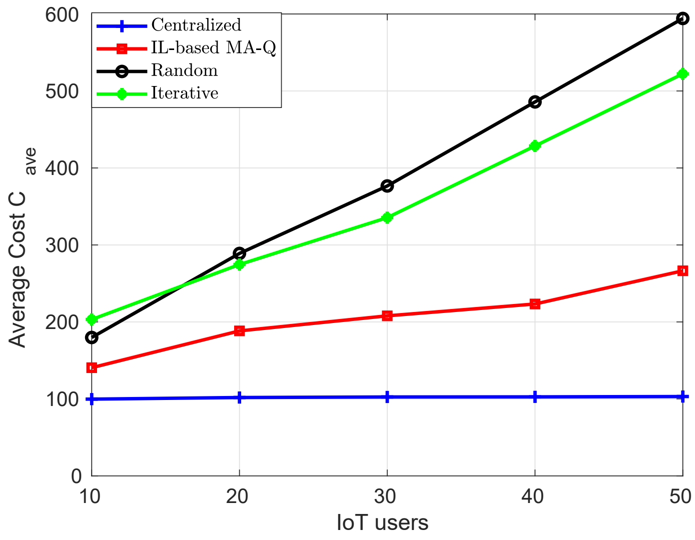


Figure 5.7: Comparisons for average cost C_{ave}, with different IoT users under algorithms Centralized, IL-based MA-Q and Random.

4. Random computation offloading: each user randomly chooses to offload its computation tasks using LoRa channels or WiFi channels, or locally execute its computation tasks.

Fig. 5.7 shows that the average cost is increasing with the number of IoT users increasing. However, the centralized algorithm does not show obvious upward trend, this is because it gets the solution with knowing the complete information of the environment, that is, there is no competition among users such that the interference will not increase with the increasing number of users. The proposed IL-based MA-Q algorithm has lower average cost than the random and iterative based computation offloading, and the iterative algorithm takes a long time to converge with a large action space. It is observed that even though the centralized algorithm has around 20% less cost than the proposed algorithm, it has higher complexity. Moreover, the proposed algorithm can be achieved in a distributed way, which reduces the computation burden on the gateway, and it saves the extra cost with no need of performing channel estimation.

5.6 Summary

Towards addressing the multiuser computation offloading problem with limited resources, like power, computation and spectrum, in IoT edge computing networks, a joint problem of multiuser computation offloading and resource allocation is formulated. Then, a non-cooperative multiuser computation offloading game is formulated with multiple selfish and self-interested users making decisions on computation offloading simultaneously while competing for resources from the gateway. Moreover, an IL-based MA-Q algorithm was developed to solve the formulated problem. The proposed algorithm enabled each IoT user to independently learn their computation offloading strategies by considering the aggregated effect of the other users in the defined reward function. The feasibility of the proposed IL-based MA-Q algorithm applied to different scale of IoT networks has been verified by simulation results. Compared to the iterative and random benchmark algorithms, it achieved less system cost, and improved computation performance for the multiuser IoT edge computing networks with distributed learning.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

This thesis focused on solving the resource allocation problem of computation offloading in the Internet of Things (IoT) networks by adopting efficient resource management methods and especially exploiting machine learning techniques, which is considered as one of the biggest challenges in the future intelligent IoT networks aiming to achieve connected intelligence. To deal with data processing in the intelligent IoT networks, compared to the conventional cloud computing, edge/fog computing with its distributive property can reduce network congestion and shrink communication latency by exploring intelligent computation offloading mechanisms based on Artificial Intelligence (AI). By invoking machine learning techniques, like clustering and Reinforcement Learning (RL) algorithms, the complex and dynamic computation offloading problem can be converted to the data-driven framework or formulated by the Markov Decision Process (MDP), which is of great significance in IoT edge computing networks since the conventional approaches are restricted by large scale problem and specific application scenarios. During my PhD study, the fundamental research was carried out to design optimal resource allocation schemes and computation offloading strategy by exploiting matching theory and machine learning algorithms, with particular effort to achieve distributed learning to improve network throughput, energy efficiency and data processing latency.

It has been an effective way for the resource-limited IoT users to transmit their collected data to a central server for data processing. In this case, the IoT users offload all the collected data to the central server to get data processing, but which raises the challenge of spectrum allocation. Besides, the IoT users need sufficient power to transmit the collected data to the gateway. The use of energy harvesting techniques provides continuous power supply for data transmission but makes the power allocation more challenging. Therefore, these challenges were addressed in chapter 3 by proposing matching theory based Efficient Channel Allocation Algorithm (ECAA) and online power allocation framework. The proposed ECAA efficiently assigned the IoT users to the available access channels without collisions, achieved 80% of the optimal performance but with much lower complexity and achieved better performance than the random channel assignment. Moreover, the online power allocation framework achieved dynamic transmit power control for each IoT user by the proposed Dynamic Programming (DP) and RL based power allocation algorithms. The comparison of network throughput performance between both proposed power allocation algorithms and the offline power allocation scheme was carried. The DP-based power allocation has highest network throughput, and Q-learning-based algorithm could reach 90% of the highest performance but with no need of pre-known network state transition probability function.

However, offloading all the data to a central server consumes large communication bandwidth and transmit power. Moreover, compared to cloud computing, edge server has finite computation and power resource. The development of more powerful chips enables the IoT users to execute partial computation tasks locally, so partial computation offloading is more reasonable to be considered in practical IoT networks. Nevertheless, it brings the concern that which task should be offloaded and how it should be offloaded, how many users should be chosen to offload their tasks. To address these issues, machine learning approaches with two steps were explored in chapter 4. Firstly, a centralized K-means based clustering algorithm was proposed to group IoT users into different clusters in IoT edge computing network, which roughly decided the computation offloading strategies for the IoT users in the clusters with highest and lowest user priority. Next, for the users in the remaining clusters, a distributed computation offloading strategy was learned by exploiting Deep Q-Network (DQN) algorithm to minimize the long-term system cost by considering the single user computation offloading scenario. It was demonstrated that the clustering algorithm could eliminate redundant processing for those users with relatively obvious decisions on computation offloading. Moreover, the DQN-based computation offloading mechanism achieved lower system cost than the other three benchmark schemes and had a neutral performance separately considering energy consumption and execution latency.

Machine learning approaches have received good performance to learn computation offloading strategy and efficient resource allocation schemes for IoT users from chapter 4. The distributed computation offloading strategy based on DQN was learned by only considering the single user scenario and ignoring other users' interference and resource competition. In reality, multiple users are likely offloading computation tasks simultaneously, which causes competition for limited system resources and interference among users. Chapter 5 investigated this problem as a joint computation offloading and resource allocation optimization problem, and formulated it as a non-cooperative stochastic game. Next, a Multi-Agent Reinforcement Learning (MARL) framework was explored to solve the formulated problem and then an Independent Learners based Multi-Agent Q-learning (IL-based MA-Q) algorithm was developed to learn the efficient computation offloading strategy and resource allocation solutions, with special effort on the design of each user's reward function by aggregating the effect of all the other users. Simulations have demonstrated the feasibility of the proposed IL-based MA-Q learning algorithm applied to different scale of IoT networks, and it achieved lower system cost compared to the iterative and random benchmark algorithms. Moreover, it was shown 20% more cost than the centralized algorithm but with less complexity. Consequently, it achieved efficient resource management and improved computation performance for the users in IoT edge computing networks with distributed learning.

To sum up, in this thesis, the efficient resource allocation methods and computation offloading mechanisms have been proposed for computing offloading in IoT networks based on matching theory and machine learning techniques with an emphasis on improving network throughput, energy efficiency and network latency, as well as achieving distributed learning. An amount of simulations have been performed. It demonstrated the feasibility of the proposed matching theory-based channel allocation algorithm, the joint RL-based computation offloading strategies and resource allocation schemes. Moreover, they were proved to be superior to the benchmark algorithms and have less complexity than the optimal scheme, and have achieved distributed learning.

6.2 Future Work

A few research issues have been identified and will be addressed in the future work, for achieving edge intelligence enabled IoT networks which contributes an important part in 6G networks.

6.2.1 Federated Learning for Edge Intelligence

Based on the research work in chapter 4, each IoT user has to run an independent deep Q-network to learn the optimal computation offloading strategy and resource allocation solutions. However, each user has limited data source, and is always resource-constrained, including power-limited battery and capacity-limited processor. Federated learning enables collaborative training with the master-slave architecture where multiple IoT devices are slavery to the edge server by offloading their local training model parameters and then getting the global training model update from the edge server, which has received much attention recently. As mentioned in section 2.3, edge intelligence requires the edge server to provide both machine learning enabled data processing and distributed AI model integration. This raises the challenge in computation offloading, i.e., the joint design of data offloading and model parameters offloading, especially for offloading data and the model parameters of the learning model for computation offloading scheme itself.

6.2.2 Distributed Intelligence

As mentioned before, federated learning generally enhanced distributed AI model training with a central parameter server to integrate the local AI models from the IoT devices. In this case, network congestion is still challenging because many users need to offload their model parameters to the central server. The partially/fully decentralized AI model training method, in which users might rely on their closest neighbours or more powerful devices to support their AI model training, is attracting the attention from researchers. Moreover, according to research in chapter 5, to deal with the non-cooperative multiuser computation offloading game, each user has to learn the actions taken by the other users since it is observing local information from the varying environment due to the joint action of all the users. Therefore, the design of fully distributed learning method when it is hard or impossible to know the coupled actions from other users and the environmental changes is full of challenges, at the same time it is essential to be addressed for 6G networks.

6.2.3 Embedded Intelligence for IoT Applications

Along with the development of AI services in our lives, like augmented reality and virtual reality, and autonomous driving, they require data-driven machine learning methods to analyse the massive amount of data and then make real-time decisions. More critical and efficient communication technologies via machine learning algorithms are needed to satisfy the extreme requirements, including ultra-low latency and ultra-low energy consumption. Specifically, embedding machine learning algorithms into the resource-constrained IoT devices raises a big challenge for algorithm design and hardware implementation. Here, the design of machine learning algorithm needs to focus on low complexity and easily ported. In contrast, the hardware design has to be put more attention on more powerful and integrated AI chips.

References

[1] T. Taleb, "White paper on 6G networking," *6G research visions, No.6*, Jun. 2020.

[2] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 856–868, Jan. 2019.

[3] H. Ye, G. Y. Li, and B.-H. Juang, "Power of deep learning for channel estimation and signal detection in ofdm systems," *IEEE Wireless Commun. Lett.*, vol. 7, no. 1, pp. 114–117, 2017.

[4] J. Xu, L. Chen, and S. Ren, "Online learning for offloading and autoscaling in energy harvesting mobile edge computing," *IEEE Trans. Cognitive Commun. and Netw.*, vol. 3, no. 3, pp. 361–373, Sep. 2017.

[5] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhuang, "Learning-based computation offloading for IoT devices with energy harvesting," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1930–1941, Feb. 2019.

[6] J. Zheng, Y. Cai, Y. Wu, and X. Shen, "Dynamic computation offloading for mobile cloud computing: A stochastic Game-Theoretic approach," *IEEE Trans. Mobile Comput.*, vol. 18, no. 4, pp. 771–786, Apr. 2019.

[7] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Commun. Surveys Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[8] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future generation computer*

systems, vol. 29, no. 7, pp. 1645–1660, 2013.

[9] N. Sornin, M. Luis, T. Eirich, T. Kramp, and O.Hersent, "LoRaWAN specification," *LoRa Alliance*, Jan. 2015. [Online]. Available: https://www.lora-alliance.org/portals/0/specs/LoRaWAN%20Specification%201R0.pdf

[10] "Sigfox", accessed Oct. 2016. [Online]. Available: https://www.sigfox.com/

[11] 3GPP, "Narrowband IoT," 3rd Generation Partnership Project (3GPP), TS 151621, Sep. 2016.

[12] X. Xiong, K. Zheng, R. Xu, W. Xiang, and P. Chatzimisios, "Low power wide area machine-to-machine networks: key techniques and prototype," *IEEE Commun. Mag.*, vol. 53, no. 9, pp. 64–71, Sep. 2015.

[13] M. R. Palattella, M. Dohler, A. Grieco, G. Rizzo, J. Torsner, T. Engel, and L. Ladid, "Internet of Things in the 5G era: Enablers, architecture, and business models," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 3, pp. 510–527, Mar. 2016.

[14] U. Raza, P. Kulkarni, and M. Sooriyabandara, "Low power wide area networks: An overview," *IEEE Commun. Surveys Tutorials*, vol. PP, no. 99, pp. 1–1, 2017.

[15] Z. Qin, F. Y. Li, G. Li, J. A. McCann, and Q. Ni, "Low-power wide-area networks for sustainable IoT," *IEEE Wireless Commun.*, 2018.

[16] Z. Qin and J. A. McCann, "Resource efficiency in low-power wide-area networks for IoT applications," in *in Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2017, pp. 1–7.

[17] S. Kartakis, B. D. Choudhary, A. D. Gluhak, L. Lambrinos, and J. A. McCann, "Demystifying low-power wide-area communications for city IoT applications," in *Proc. of Int. Workshop Wireless Netw. Testbeds, Experimental Evaluation, Characterization, (WiNTECH'16).* New York, NY, USA: ACM, Sep. 2016, pp. 2–8.

[18] M. C. Bor, U. Roedig, T. Voigt, and J. M. Alonso, "Do LoRa low-power wide-area networks scale?" in *Proc. of ACM Intl. Conf. Modeling, Analysis and Simul. Wireless and Mobile Systems, (MSWiM'16)*, Nov. 2016, pp. 59–67.

[19] M. Bor, J. Vidler, and U. Roedig, "LoRa for the Internet of Things," in *Proc.*

of Intl. Conf. Embedded Wireless Syst. Netw. (EWSN'16), Graz, Austria, Feb. 2016, pp. 361–366.

[20] M. Haghighi, Z. Qin, D. Carboni, U. Adeel, F. Shi, and J. A. McCann, "Game theoretic and auction-based algorithms towards opportunistic communications in LPWA LoRa networks," in *IEEE World Forum Internet of Things (WF-IoT'16)*, Dec. 2016, pp. 735–740.

[21] O. Georgiou and U. Raza, "Low power wide area network analysis: Can LoRa scale?" *IEEE Wireless Commun. Lett.*, vol. PP, no. 99, pp. 1–1, 2017.

[22] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne, "Understanding the limits of LoRaWAN," *IEEE Commun. Mag.*, vol. 55, no. 9, pp. 34–40, Sep. 2017.

[23] Y. He, X. Cheng, W. Peng, and G. L. Stuber, "A survey of energy harvesting communications: Models and offline optimal policies," *IEEE Commun. Mag.*, vol. 53, no. 6, pp. 79–85, Jun. 2015.

[24] D. Mishra, S. De, S. Jana, S. Basagni, K. Chowdhury, and W. Heinzelman, "Smart rf energy harvesting communications: Challenges and opportunities," *IEEE Commun. Mag.*, vol. 53, no. 4, pp. 70–78, Apr. 2015.

[25] N. Harpawi *et al.*, "Design energy harvesting device of UHF TV stations," *in 8th Intl. Conf. Telecom. Systems Services and Applications (TSSA)*, pp. 1–6, Oct. 2014.

[26] S. Keyrouz, H. Visser, and A. Tijhuis, "Ambient RF energy harvesting from DTV stations," *Antennas and Propagation Conference (LAPC), Loughborough*, pp. 1–4, Nov. 2012.

[27] H. Nishimoto, Y. Kawahara, and T. Asami, "Prototype implementation of wireless sensor network using TV broadcast RF energy harvesting," *in Adjunct Proc. of 12th ACM Int. conf. on Ubicomp'10.*, pp. 373–374, Sep. 2010.

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

[29] Ericsson, "Ericsson mobility report," Ericsson, https://www.ericsson.com/4acd7e/assets/local/m

report/documents/2019/emr-november-2019.pdf, Tech. Rep., 2019.

[30] E. Union, "Identification and quantification of key socio-economicdata to support strategic planning for the introduction of 5G in Europe -SMART," 2014/0008, Tech. Rep., 2016.

[31] L. Zhang, Y.-C. Liang, and M. Xiao, "Spectrum sharing for internet of things: A survey," *IEEE Wireless Commun.*, vol. 26, no. 3, pp. 132–139, Jun. 2019.

[32] D. Xu, T. Li, Y. Li, X. Su, S. Tarkoma, and P. Hui, "A survey on edge intelligence," *arXiv preprint arXiv:2003.12172*, 2020.

[33] A. E. Roth, "Deferred acceptance algorithms: History, theory, practice, and open questions," *International Journal of game Theory*, vol. 36, no. 3, pp. 537–569, 2008.

[34] A. Leshem, E. Zehavi, and Y. Yaffe, "Multichannel opportunistic carrier sensing for stable channel access control in cognitive radio systems," *IEEE J. Sel. Areas Commun.*, vol. 30, no. 1, pp. 82–95, 2011.

[35] S. Bayat, R. H. Louie, Z. Han, B. Vucetic, and Y. Li, "Distributed user association and femtocell allocation in heterogeneous wireless networks," *IEEE Trans. Commun.*, vol. 62, no. 8, pp. 3027–3043, 2014.

[36] O. Semiari, W. Saad, S. Valentin, M. Bennis, and H. V. Poor, "Context-aware small cell networks: How social metrics improve wireless resource allocation," *IEEE Trans. Wireless Commun.*, vol. 14, no. 11, pp. 5927–5940, 2015.

[37] E. A. Jorswieck, "Stable matchings for resource allocation in wireless networks," in *2011 17th International Conference on Digital Signal Processing (DSP)*, 2011, pp. 1–8.

[38] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," *The American Mathematical Monthly*, vol. 69, no. 1, pp. 9–15, 1962.

[39] Z. Han and K. Liu, *Resource allocation for wireless networks: basics, techniques, and applications*. Cambridge university press, 2008.

[40] Y. Gu, Y. Zhang, M. Pan, and Z. Han, "Matching and cheating in device to device communications underlying cellular networks," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 10, pp. 2156–2166, 2015.

[41] R. Mochaourab, B. Holfeld, and T. Wirth, "Distributed channel assignment in cognitive radio networks: Stable matching and walrasian equilibrium," *IEEE Trans. Wireless Commun.*, vol. 14, no. 7, pp. 3924–3936, 2015.

[42] A. Waret, M. Kaneko, A. Guitton, and N. El Rachkidy, "LoRa throughput analysis with imperfect spreading factor orthogonality," *IEEE Wireless Commun. Lett.*, vol. 8, no. 2, pp. 408–411, 2018.

[43] A. Muqattash and M. Krunz, "CDMA-based MAC protocol for wireless ad hoc networks," in *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, 2003, pp. 153–164.

[44] M. Othman, S. A. Madani, S. U. Khan *et al.*, "A survey of mobile cloud computing application models," *IEEE Commun. Surveys Tutorials*, vol. 16, no. 1, pp. 393–413, Jul. 2013.

[45] M. Kamoun, W. Labidi, and M. Sarkiss, "Joint resource allocation and offloading strategies in cloud enabled cellular networks," in *2015 IEEE International Conference on Communications (ICC).* IEEE, Jun. 2015.

[46] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.

[47] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhuang, "Learning-based computation offloading for iot devices with energy harvesting," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1930–1941, Jan. 2019.

[48] S. Bi, Y. Zeng, and R. Zhang, "Wireless powered communication networks: An overview," *IEEE Wireless Commun.*, vol. 23, no. 2, pp. 10–18, Apr. 2016.

[49] H. Ju and R. Zhang, "Throughput maximization in wireless powered communication networks," *IEEE Trans. Wireless Commun.*, vol. 13, no. 1, pp. 418–428, Jan. 2014.

[50] L. Liu, R. Zhang, and K.-C. Chua, "Multi-antenna wireless powered communication with energy beamforming," *IEEE Trans. Commun.*, vol. 62, no. 12, pp. 4349–4361, Dec. 2014.

[51] D. K. P. Asiedu, S. Mahama, and K.-J. Lee, "Joint beamforming and resource

allocation for multi-user full-duplex wireless powered communication networks," in *Proc. IEEE Vehicular Technology Conference (VTC Spring)*, Jun. 2018, pp. 1–5.

[52] R. A. Loodaricheh, S. Mallick, and V. K. Bhargava, "QoS provisioning based resource allocation for energy harvesting systems," *IEEE Trans. Wireless Commun.*, vol. 15, no. 7, pp. 5113–5126, Jul. 2016.

[53] M. R. Zenaidi, Z. Rezki, and M.-S. Alouini, "Performance limits of online energy harvesting communications with noisy channel state information at the transmitter," *IEEE Access*, vol. 5, pp. 1239–1249, Mar. 2017.

[54] S. Mao, M. H. Cheung, and V. W. Wong, "Joint energy allocation for sensing and transmission in rechargeable wireless sensor networks," *IEEE Trans. Veh. Technol.*, vol. 63, no. 6, pp. 2862–2875, Jul. 2014.

[55] P. Sakulkar and B. Krishnamachari, "Online learning schemes for power allocation in energy harvesting communications," *IEEE Trans. Inf. Theory*, vol. 64, no. 6, pp. 4610–4628, Jun. 2018.

[56] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, "All one needs to know about fog computing and related edge computing paradigms: A complete survey," *arXiv preprint arXiv:1808.05283*, Sep. 2018.

[57] L. Bittencourt, R. Immich, R. Sakellariou, N. Fonseca, E. Madeira, M. Curado, L. Villas, L. da Silva, C. Lee, and O. Rana, "The internet of things, fog and cloud continuum: Integration and challenges," *IEEE Internet Things J.*, Oct. 2018.

[58] C.-H. Hong and B. Varghese, "Resource management in fog/edge computing: A survey," *arXiv preprint arXiv:1810.00305*, Sep. 2018.

[59] D. T. Nguyen, L. B. Le, and V. Bhargava, "A market-based framework for multi-resource allocation in fog computing," *arXiv preprint arXiv:1807.09756*, Jul. 2018.

[60] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *arXiv preprint arXiv:1702.05309*, Mar. 2017.

[61] Y. Mao, J. Zhang, and K. B. Letaief, "Joint task offloading scheduling and transmit power allocation for mobile-edge computing systems," in *Wireless Commun. and*

Networking Conf. (WCNC 2017). San Francisco, CA: IEEE, Jan. 2017, pp. 1–6.

[62] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Trans. Signal Inf. Process. Over Netw.*, vol. 1, no. 2, pp. 89–103, Jun. 2015.

[63] A. Al-Shuwaili and O. Simeone, "Energy-efficient resource allocation for mobile edge computing-based augmented reality applications," *IEEE Wireless Commun. Lett.*, vol. 6, no. 3, pp. 398–401, Jun. 2017.

[64] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *2016 IEEE International Symposium on Information Theory (ISIT).* IEEE, Jul. 2016.

[65] W. Labidi, M. Sarkiss, and M. Kamoun, "Energy-optimal resource scheduling and computation offloading in small cell networks," in *2015 22nd International Conference on Telecommunications (ICT).* IEEE, Apr. 2015.

[66] S. Cao, X. Tao, Y. Hou, and Q. Cui, "An energy-optimal offloading algorithm of mobile computing based on HetNets," in *2015 International Conference on Connected Vehicles and Expo (ICCVE).* IEEE, Oct. 2015.

[67] S. Bi and Y. J. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, pp. 4177–4190, Apr. 2018.

[68] O. Muñoz, A. Pascual-Iserte, and J. Vidal, "Joint allocation of radio and computational resources in wireless application offloading," in *2013 Future Network & Mobile Summit.* IEEE, Jul. 2013, pp. 1–10.

[69] O. Munoz, A. Pascual-Iserte, and J. Vidal, "Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading," *IEEE Trans. Veh. Technol.*, vol. 64, no. 10, pp. 4738–4755, Oct. 2015.

[70] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Power-delay tradeoff in multi-user mobile-edge computing systems," in *2016 IEEE Global Communications Conference (GLOBECOM).* IEEE, Dec. 2016.

[71] M.-H. Chen, B. Liang, and M. Dong, "A semidefinite relaxation approach to mobile cloud offloading with computing access point," in *2015 IEEE 16th International*

Workshop on Signal Processing Advances in Wireless Communications (SPAWC). IEEE, Jun. 2015.

[72] W. Labidi, M. Sarkiss, and M. Kamoun, "Joint multi-user resource scheduling and computation offloading in small cell networks," in *2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob).* IEEE, Oct. 2015.

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

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

[75] C. You and K. Huang, "Multiuser resource allocation for mobile-edge computation offloading," in *in Proc. Global Communication Conf.(GLOBECOM)*, Washington, DC USA, Dec. 2016, pp. 1–6.

[76] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.

[77] O. Munoz, A. P. Iserte, J. Vidal, and M. Molina, "Energy-latency trade-off for multiuser wireless computation offloading," in *2014 IEEE Wireless Communications and Networking Conference Workshops (WCNCW).* IEEE, Apr. 2014.

[78] G. Zhu, D. Liu, and Y. Du, "Towards an intelligent edge: Wireless communication meets machine learning," *arXiv preprint arXiv:1809.00343*, Sep. 2018.

[79] Y. Du and K. Huang, "Fast analog transmission for high-mobility wireless data acquisition in edge learning," *arXiv preprint arXiv:1807.11250*, Jul. 2018.

[80] J. Park, S. Samarakoon, M. Bennis, and M. Debbah, "Wireless network intelligence at the edge," *Proceedings of the IEEE*, vol. 107, no. 11, pp. 2204–2239, Nov. 2019.

[81] W. Zhan, C. Luo, J. Wang, G. Min, and H. Duan, "Deep reinforcement learning-based computation offloading in vehicular edge computing," in *2019 IEEE Global Communications Conference (GLOBECOM)*, Dec. 2019.

[82] X. Xiong, K. Zheng, L. Lei, and L. Hou, "Resource allocation based on deep reinforcement learning in IoT edge computing," *IEEE J. Sel. Areas Commun.*, pp. 1–1, Apr. 2020.

[83] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4005–4018, Jun. 2019.

[84] ——, "Performance optimization in mobile-edge computing via deep reinforcement learning," *arXiv preprint arXiv:1804.00514*, Mar. 2018.

[85] C. Zhang, Z. Liu, B. Gu, and etc., "A deep reinforcement learning based approach for cost-and energy-aware multi-flow mobile data offloading," *IEICE Trans. on Commun.*, pp. 1625–1634, Jan. 2018.

[86] X. Chen, H. Zhang, and C. Wu, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet Things J.*, Jun. 2018.

[87] J. Chen, S. Chen, Q. Wang, B. Cao, G. Feng, and J. Hu, "iRAF: a deep reinforcement learning approach for collaborative mobile edge computing IoT networks," *IEEE Internet Things J.*, Apr. 2019.

[88] H. Peng and X. S. Shen, "Deep reinforcement learning based resource management for multi-access edge computing in vehicular networks," *IEEE Trans. Netw. Scien. and Engin.*, pp. 1–1, Jun. 2020.

[89] Z. Zhang, F. R. Yu, F. Fu, Q. Yan, and Z. Wang, "Joint offloading and resource allocation in mobile edge computing systems: An actor-critic approach," in *2018 IEEE Global Communications Conference (GLOBECOM).* IEEE, Dec. 2018.

[90] L. Huang, S. Bi, and Y. J. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Trans. Mobile Comput.*, Jul. 2020.

[91] T. Yang, Y. Hu, M. C. Gursoy, A. Schmeink, and R. Mathar, "Deep reinforcement learning based resource allocation in low latency edge computing networks," in *2018 15th International Symposium on Wireless Communication Systems (ISWCS).* IEEE, Aug. 2018.

[92] D. Borthakur, H. Dubey, and N. Constant, "Smart fog: Fog computing framework for unsupervised clustering analytics in wearable internet of things," in *in Proc. Global Conf. Signal Inf. Process. (GlobalSIP)*, Montreal, Canada, Nov. 2017, pp. 472–476.

[93] X. Liu, Z. Qin, and Y. Gao, "Resource allocation for edge computing in IoT networks via reinforcement learning," in *Proc.IEEE Int. Conf. Communication (ICC)*, Shanghai, China, May 2019, pp. 1–6.

[94] X. Liu, J. Yu, J. Wang, and Y. Gao, "Resource allocation with edge computing in IoT networks via machine learning," *IEEE Internet Things J.*, Feb. 2020.

[95] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Wireless Commun.*, vol. 16, no. 9, pp. 5994–6009, Sep. 2017.

[96] S. Bi and Y. J. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, pp. 4177–4190, Jun. 2018.

[97] Z. Ning, P. Dong, X. Kong, and F. Xia, "A cooperative partial computation offloading scheme for mobile edge computing enabled internet of things," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4804–4814, Jun. 2019.

[98] W. Chen, D. Wang, and K. Li, "Multi-user multi-task computation offloading in green mobile edge cloud computing," *IEEE Trans. Serv. Comput.*, vol. 12, no. 5, pp. 726–738, Sep. 2019.

[99] L. Tang and S. He, "Multi-user computation offloading in mobile edge computing: A behavioral perspective," *IEEE Netw.*, vol. 32, no. 1, pp. 48–53, Jan. 2018.

[100] H. Guo, J. Liu, J. Zhang, W. Sun, and N. Kato, "Mobile-edge computation offloading for ultradense IoT networks," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4977–4988, Dec. 2018.

[101] J. Ren, G. Yu, Y. Cai, and Y. He, "Latency optimization for resource allocation in mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 17, no. 8, pp. 5506–5519, Aug. 2018.

[102] Y. Wu, L. P. Qian, K. Ni, C. Zhang, and X. Shen, "Delay-minimization nonorthog-

onal multiple access enabled multi-user mobile edge computation offloading," *IEEE J. Sel. Signal Process.*, vol. 13, no. 3, pp. 392–407, Jun. 2019.

[103] F. Wang, J. Xu, and Z. Ding, "Multi-antenna NOMA for computation offloading in multiuser mobile edge computing systems," *IEEE Trans. Commun.*, vol. 67, no. 3, pp. 2450–2463, Mar. 2019.

[104] C.-F. Liu, M. Bennis, and H. V. Poor, "Latency and reliability-aware task offloading and resource allocation for mobile edge computing," in *2017 IEEE Globecom Workshops (GC Wkshps)*, Dec. 2017.

[105] S. Ranadheera, S. Maghsudi, and E. Hossain, "Mobile edge computation offloading using game theory and reinforcement learning," *arXiv preprint arXiv:1711.09012*, Nov. 2017.

[106] Y. Zhang, B. Di, Z. Zheng, J. Lin, and L. Song, "Joint data offloading and resource allocation for multi-cloud heterogeneous mobile edge computing using multi-agent reinforcement learning," in *2019 IEEE Global Communications Conference (GLOBECOM)*, Dec. 2019.

[107] X. Chen, Z. Zhao, and H. Zhang, "Stochastic power adaptation with multiagent reinforcement learning for cognitive wireless mesh networks," *IEEE Trans. Mobile Comput.*, vol. 12, no. 11, pp. 2155–2166, Nov. 2012.

[108] J. Cui, Y. Liu, and A. Nallanathan, "Multi-agent reinforcement learning based resource allocation for UAV networks," *IEEE Trans. Wireless Commun.*, Feb. 2020.

[109] A. E. Roth, "The economics of matching: Stability and incentives," *Mathematics of operations research*, vol. 7, no. 4, pp. 617–628, 1982.

[110] F. Echenique and J. Oviedo, "A theory of stability in many-to-many matching markets," 2004.

[111] Y. Gu, W. Saad, M. Bennis, M. Debbah, and Z. Han, "Matching theory for future wireless networks: fundamentals and applications," *IEEE Commun. Mag.*, vol. 53, no. 5, pp. 52–59, May 2015.

[112] K. Zhang, Z. Yang, and T. Başar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," *arXiv preprint arXiv:1911.10635*, 2019.

[113] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Networks and Applications*, vol. 18, no. 1, pp. 129–140, Apr. 2012.

[114] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tutorials*, vol. 19, no. 3, pp. 1628–1656, Feb. 2017.

[115] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "Mobile edge computing: Survey and research outlook," *arXiv preprint arXiv*, vol. 1701, Jan. 2017.

[116] X. Liu and Y. Gao, "Reinforcement learning approaches for iot networks with energy harvesting," in *2019 IEEE/CIC International Conference on Communications in China (ICCC)*. IEEE, 2019, pp. 85–90.

[117] X. Liu, Z. Qin, Y. Gao, and J. A. McCann, "Resource allocation in wireless powered iot networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4935–4945, 2019.

[118] Y. Gu, C. Jiang, L. X. Cai, M. Pan, L. Song, and Z. Han, "Dynamic path to stability in LTE-unlicensed with user mobility: A matching framework," *IEEE Trans. Wireless Commun.*, vol. 16, no. 7, pp. 4547–4561, 2017.

[119] S. Bayat, Y. Li, L. Song, and Z. Han, "Matching theory: Applications in wireless communications," *IEEE Signal Process. Mag.*, vol. 33, no. 6, pp. 103–122, Nov. 2016.

[120] E. Bodine-Baron, C. Lee, A. Chong, B. Hassibi, and A. Wierman, "Peer effects and stability in matching markets," in *International Symposium on Algorithmic Game Theory*. Springer, 2011, pp. 117–129.

[121] B. Su, Z. Qin, and Q. Ni, "Energy efficient uplink transmissions in LoRa networks," *IEEE Transactions on Communications*, vol. 68, no. 8, pp. 4960–4972, 2020.

[122] D. P.Bertsekas, *Dynamic programming and optimal control*, 2nd ed. Athena scientific Belmont, MA, 1995, vol. 1 and 2.

[123] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[124] X. Liu, J. Yu, and Y. Gao, "Multi-agent reinforcement learning for resource allocation in iot networks with edge computing," *arXiv preprint arXiv:2004.02315*, Apr.

2020.

[125] A. Neyman, "From markov chains to stochastic games," in *Stochastic Games and Applications.* Springer, 2003, pp. 9–25.