Performance Improvements for Wireless Mobile Networks via Deep Reinforcement Learning

by

Ying He

A dissertation submitted to the

Faculty of Graduate Studies and Research

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Electrical and Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering (OCIECE)

Department of Systems and Computer Engineering

Carleton University

Ottawa, Ontario, Canada, K1S 5B6

June, 2019

The undersigned hereby recommends to the

Faculty of Graduate Studies and Research

acceptance of the dissertation

Performance Improvements for Wireless Mobile Networks via Deep Reinforcement Learning

submitted by

Ying He, **M.Eng**

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Electrical and Computer Engineering

Professor Cheng Li, External Examiner

Professor Fei Richard Yu, Thesis Supervisor

Professor Peter Liu, Examiner

Professor Amiya Nayak, Examiner

Professor Halim Yanikomeroglu, Examiner

Carleton University

June 2019

Abstract

Deep reinforcement learning (DRL) is evolved from a collection of powerful techniques in artificial intelligence areas, and has been extensively used in different areas. In DRL, an agent learns to take actions that would yield the most reward by interacting with the environment without prior knowledge of an exact mathematical model of the environment. The key idea of DRL is to approximate the value function by leveraging the powerful function approximation property of deep neural networks.

In this work, we investigate the performance improvements for wireless mobile networks via DRL. We firstly present a DRL approach in cache-enabled opportunistic interference alignment wireless networks. Most existing works on cache-enabled interference alignment wireless networks assume that the channel is invariant, which is unrealistic considering the time-varying nature of practical wireless environments. We consider realistic time-varying channels. The complexity of the system is very high when we consider realistic time-varying channels. We use Google TensorFlow to implement DRL in this chapter to obtain the optimal interference alignment user selection policy in cache-enabled opportunistic interference alignment wireless networks. Simulation results are presented to show that the performance in terms of the network's sum rate and energy efficiency can be significantly improved by using the proposed approach.

Secondly, we design a software-defined framework for connected vehicles, which integrates communication, caching and mobile edge computing. A deep reinforcement

learning-based resource allocation scheme is proposed for the connected vehicles. The dynamic change processes of the resources are modeled as Markov chains, respectively. Without any assumptions about the objective funtions or any low-complexity preprocessing, the proposed scheme can directly solve the resource allocation problems with large-scale state space. Simulation results verify that the proposed scheme can converge at a fast speed, improve the network operators total utilities, and possess the ability of resisting perturbation at a certain level.

Thirdly, we study trust-based social networks with mobile edge computing, in-network caching and device-to-device communications. An optimization problem is formulated to maximize the network operator's utility with comprehensive considerations of trust values, computation capabilities, wireless channel qualities, and the cache status of all the available nodes. We apply a DRL approach to automatically make a decision for optimally allocating the network resources. The decision is made purely through observing the network's states, rather than any handcrafted or explicit control rules, which makes it adaptive to variable network conditions. Simulation results with different network parameters are presented to show the effectiveness of the proposed scheme.

Acknowledgments

First of all, I would like to deeply and gratefully thank my supervisor, Prof. Fei Richard Yu, for his invaluable support from all aspects and wonderful supervision. His excellent research insight and on-going encouragement bring me an amount of momentum to explore the research. His comments and suggestions on my works not only increase the quality of the research but also provide a source of thought for my career. This work is as much his contribution as is mine for the fact that he has always wholeheartedly supported every decision I have made during this wonderful study. He has better prepared me for the world and I will always be indebted to him for anything I achieve in my future life.

I would like to thank Dr. Helen Tang, for her valuable comments and suggestions during my research and study. Besides, I want to thank Dr. Shengrong Bu and Dr. Zhexiong Wei for giving me a lot of support and valuable advice in my research.

I would like to thank my colleagues at Carleton University, including Dr. Chengchao Liang, Ms. Mengting Liu, Ms. Chao Qiu, Ms. Qi Song, Mr. Chengmeng Wang, Mr. Dajun Zhang, Ms. Yuchen Zhou, Ms. Yashuang Guo. I also appreciate the service and support from the staff in our department, who have patience and passion to help every student.

I greatly appreciate my parents for their understanding throughout the journey of my PhD study.

Table of Contents

List of Tables

List of Abbreviations

ADMM	Alternating Direction Method of Multiplier
BS	Base Stations
CDNs	Content Delivery Networks
CSI	Channel State Information
DRL	Deep Reinforcement Learning
D2D	Device-to-Device
FSMC	Finite-state Markov Channel
IA	Interference Alignment
ICN	Information-Centric Networking
LSTM	Long Short-Term Memory

MSNs	Mobile Social Networks
MDP	Markov Decision Process
MIMO	Mutiple-Input and Multiple-Output
MEC	Mobile Edge Computing
MCC	Mobile Cloud Computing
NN	Neural Network
NFV	Network Functions Virtualization
OIA	Opportunistic Interference Alignment
P2P	Peer-to-Peer
RSU	Road Side Unit
SDN	Software-Defined Networking
SOM	Self-Organizing Map
SINR	Signal to Interference and Noise Ratio
UE	User Equipment

Chapter 1

Introduction

1.1 Motivations

With the increasing demands for higher data rate, lower transmission delay and more reliable services, the integrated wireless networks with communications, caching and computing have become a developing trend for future networks. In the scenarios of complicated network architectures and multi-dimensional network resources, network control, user scheduling and resource allocation etc. are the main challenging technical issues. Introducing machine learning from the field of artificial intelligence into wireless networks, to facilitate network management and control, has become a hot topic. Recent advances in machine learning can have significant impacts on wireless mobile networks.

Machine learning algorithms can be basically classified into three categories: supervised, unsupervised, and reinforcement learning. Supervised learning is a kind of labelling learning technique. Supervised learning algorithms are given a labeled training dataset (i.e., inputs and known outputs) to build the system model representing the learned relation between the input and output [1, 2]. In contrast to supervised learning, an unsupervised learning algorithm is given a set of inputs without labels

(i.e., there is no output). Basically, an unsupervised learning algorithm aims to find patterns, structures, or knowledge in unlabeled data by clustering sample data into different groups according to the similarity between them. The unsupervised learning techniques are widely used in clustering and data aggregation [2, 3].

Reinforcement learning is an important branch of machine learning, where an agent learns to take actions that would yield the most reward by interacting with the environment. Different from supervised learning, reinforcement learning does not learn from samples provided by an experienced external supervisor. Instead, it has to operate based on its own experience despite that it faces with significant uncertainty about the environment [4]. Model-free reinforcement learning has recently been successfully applied to handle the deep neural network and value functions [5–8]. It can learn policies for tough tasks using the raw state representation directly as the input to the neural networks [9].

This dissertation exploits the current advances of machine learning and deep reinforcement learning algorithm, to tackle the network optimization and resource allocation issues in the integrated wireless networks with communications, caching and computing.

1.2 Contributions and Dissertation Organization

The integrated wireless networks with communication, caching and computing can meet the users's demand for lower network delay, higher bandwidth, higher reliability, stronger security and customized services. At the same time, the complicated network architecture and multi-dimensional network resources makes the network control, resource allocation and resource management more challenging. Therefore, introducing machine learning into the future networks, to facilitate the network management and control has become a hotspot of wireless networks. In this dissertation, targeted at

the three different representative wireless networks, Interference Alignment (IA) wireless networks, vehicular networks and mobile social networks, reinforcement learning method is applied,to tackle with the respective resource allocation problems in each networks. The details of contributions in each chapter is presented as follows.

In Chapter 2, we provide a brief introduction of the background of machine learning, including supervised learning, unsupervised learning, and reinforcement learning, where deep reinforcement learning is majorly presented. Then, the integrated networking, caching and computing in wireless mobile networks is introduced. First, we give out the motivations for the integrated networks, then networking, caching and computing are discussed, respectively.

In Chapter 3, we investigate the performance improvements for wireless mobile networks via deep reinforcement learning (DRL). We firstly present a DRL approach in cache-enabled opportunistic interference alignment (IA) wireless networks. Recently, wireless proactive caching has attracted great attentions from both academia and industry [10, 11]. By effectively reducing the duplicate content transmissions in networks, caching has been recognized as one of the promising techniques for future wireless networks to improve spectral efficiency, shorten latency, and reduce energy consumption [12–14]. Based on the global traffic features, a few popular contents are requested by many users during a short time span, which accounts for most of the traffic load. Therefore, proactively caching the popular contents can remove the heavy burden of the backhaul links. We make the following contributions:

- Cache-enabled opportunistic IA is studied under the condition of time-varying channel coefficients. The channel is formulated as a finite-state Markov channel (FSMC), which has been widely accepted in the literature to characterize the correlation structure of the fading process [15–17]. Considering FSMC models may enable significant performance improvements over the schemes with

memoryless channel models.

- The complexity of the system is very high when we consider realistic FSMC models. Therefore, we propose a novel deep reinforcement learning approach in this chapter. Deep reinforcement learning is an advanced reinforcement learning algorithm that uses deep Q network to approximate the Q value-action function [8], and it has been used in wireless networks to improve the performance [18, 19]. Deep reinforcement learning is used in this chapter to obtain the optimal IA user selection policy in cache-enabled opportunistic IA wireless networks.

- We use Google TensorFlow to implement deep reinforcement learning. The visualization of the deep Q network model is presented. Simulation results with different system parameters are presented to show the effectiveness of the proposed scheme. It is illustrated that the performance of cache-enabled opportunistic IA networks in terms of the network's sum rate and energy efficiency can be significantly improved by using the proposed deep reinforcement learning approach.

In Chapter 4, a deep reinforcement learning-based resource allocation scheme is proposed for the connected vehicles with the integration of communication, caching and computing. The dynamic change processes of the communication, caching, and computing resources are modeled as Markov chains, respectively. Aiming at maximizing the network operators total utility, the joint resource allocation problem is formulated as a reinforcement learning process, and the deep Q learning algorithm is used to pursue the optimal solution. Chapter 4 contains the following contributions:

- Based on the programmable control principle originated from SDN and caching originated from ICN, we propose an integrated framework that can enable dynamic orchestration of networking, caching and computing resources to improve

the performance of next generation vehicular networks.

- In this framework, we formulate the resource allocation strategy as a joint optimization problem, where the gains of not only networking but also caching and computing are taken into consideration in the proposed framework.

- The complexity of the system is very high when we jointly consider three technologies. Therefore, we propose a novel deep reinforcement learning approach. Deep reinforcement learning is an advanced reinforcement learning algorithm that uses deep Q network to approximate the Q value-action function [8], and has been exploited in wireless networks to achieve automatic resource allocation [19, 20]. Deep reinforcement learning is used to obtain the resource allocation policy in vehicular networks with integrated networking, caching, and computing.

- Simulation results with different system parameters are presented to show the effectiveness of the proposed scheme. It is illustrated that the performance of vehicular networks can be significantly improved with integrated networking, caching, and computing.

In Chapter 5, we study trust-based social networks with mobile edge computing, in-network caching and device-to-device communications. Recently, mobile social networks (MSNs) have been developing rapidly, which can provide a variety of social services and applications to mobile users [21]. Millions of mobile users interact with each other in MSNs, and MSNs will become one of the most important networking paradigms in future wireless mobile networks [22]. MSNs have always been trying to be innovative and leverage new technologies. The recently proposed integrated framework of networking, caching and computing can have positive impacts on MSNs. An optimization problem is formulated to maximize the network operators

utility with comprehensive considerations of trust values, computation capabilities, wireless channel qualities, and the cache status of all the available nodes. We apply a DRL approach to automatically make a decision for optimally allocating the network resources. We make the following contributions:

- We consider trust-based social networks specifically with mobile edge computing, in-network caching and device-to-device (D2D) communications under the umbrella of an integrated framework. An optimization problem is formulated to maximize the network operator's utility with comprehensive considerations of trust values, computation capabilities, wireless channel qualities, and the cache status of all the available BSs and D2D nodes.

- To be more realistic, we consider that the network conditions (i.e., trust value, computation capability, wireless channels, and cache status) are varying with time, and the dynamics of the computing capabilities, cache status and wireless channel conditions are modeled as Markov chains. In the meanwhile, the trust values are derived from both direct and indirect observations using Bayesian inference and Dempster-Shafer theory, respectively.

- When we jointly consider the dynamic trust values, computation capabilities, wireless channel conditions and the cache status, it is extremely difficult to solve the formulated optimization problem. In this work, we exploit DRL-based resource allocation strategy to solve the optimization problem without any explicit assumptions or simplifications.

- Simulation results with different system parameters are presented to show the effectiveness of the proposed scheme. It is illustrated that the performance of MSNs can be significantly improved with the proposed approach.

Chapter 6 gives the conclusions of this dissertation.

1.3 List of Publications

The following accepted papers are partially covered in this dissertation. They are in the order of chapters.

Chapter 3: Y. He, Z. Zhang, F. R. Yu, N. Zhao, H. Yin, and V. C. M. Leung, "Deep-reinforcement-learning-based optimization for cache-enabled opportunistic interference alignment wireless networks," *IEEE Trans. Veh. Tech.*, vol. 66, no. 11, pp. 10433-10445, Nov. 2017.

Chapter 3: Y. He, C. Liang, F. R. Yu, N. Zhao,and H. Yin, "Optimization of cache-enabled opportunistic interference alignment wireless networks: A big data deep reinforcement learning approach," in *Proc. IEEE ICC'17*, Paris, France, May 2017.

Chapter 3: Y. He, F. R. Yu, N. Zhao, H. Yin, H. Yao, and R. C. Qiu, "Big data analytics in mobile cellular networks," *IEEE Access*, vol. 4, pp. 1985-1996, 2016.

Chapter 4: Y. He, N. Zhao, and H. Yin. "Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach," *IEEE Trans. Veh. Tech.*, vol. 67, no. 1, pp. 44-55, 2018.

Chapter 4: Y. He, F. R. Yu, N. Zhao, V. C. M. Leung, and H. Yin, "Software-defined networks with mobile edge computing and caching for smart cities: A big data deep reinforcement learning approach," *IEEE Commun. Mag.*, vol. 55, no. 12, pp. 31-37, Dec. 2017.

Chapter 4: Y. He, Z. Wei, G. Du, J. Li, N. Zhao, and H. Yin, "Securing cognitive radio vehicular ad hoc networks with fog computing," *Ad Hoc Sens. Wirel. Netw.*, vol. 40, no. 1, pp. 73-95, 2018.

Chapter 4: Y. He, C. Liang, Z. Zhang, F. R. Yu, N. Zhao, H. Yin, and Y. Zhang,

"Resource allocation in software-defined and information-centric vehicular networks with mobile edge computing," in *Proc. IEEE VTC'17-Fall*, Toronto, Canada, Sept. 2017.

Chapter 5: Y. He, F. R. Yu, N. Zhao, H. Yin. "Secure social networks in 5G systems with mobile edge computing, caching and device-to-device (D2D) communication," *IEEE Wirel. Commun.*, vol. 25, no. 3, pp. 103-109, Feb. 2018.

Chapter 5: Y. He, F. R. Yu, Z. Wei, and V. C. M. Leung, Trust management for secure cognitive radio vehicular ad hoc networks," *Ad Hoc Netw.*, vol. 86, pp. 154-165, 2019.

Chapter 5: Y. He, C. Liang, F. R. Yu, and Z. Han, "Trust-based social networks with computing, caching and communications: A deep reinforcement learning approach," *IEEE Trans. Network Science and Eng.*, DOI: 10.1109/TNSE.2018.2865183, on-line, Aug. 2018.

Chapter 2

Background

In this section, we present background information about this work, including machine learning and integrated networking, caching, and computing.

2.1 Machine Learning

Recently, machine learning has attracted great attentions from both industry and academia. Machine learning has been extensively used in data mining, which allows the system to learn the useful structural patterns and models from training data. Machine learning algorithms can be basically classified into three categories: supervised, unsupervised, and reinforcement learning. In this subsection, widely-used machine learning algorithms are introduced. Each algorithm is briefly explained with some examples.

A machine learning approach usually consists of two main phases: training phase and decision making phase as illustrated in the Fig. 2.1. At the training phase, machine learning methods are applied to learn the system model using the training dataset. At the decision making phase, the system can obtain the estimated output for each new input by using the trained model.

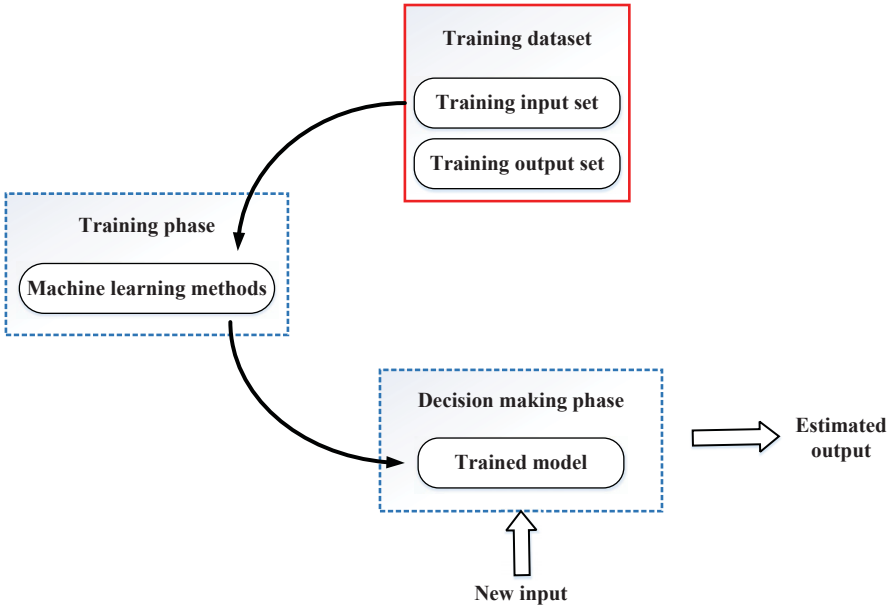

Figure 2.1: The general processing procedure of a machine learning approach.

2.1.1 Supervised Learning

Supervised learning is a kind of labelling learning technique. Supervised learning algorithms are given a labeled training dataset (i.e., inputs and known outputs) to build the system model representing the learned relation between the input and output. After training, when a new input is fed into the system, the trained model can be used to get the expected output [1, 2]. Some widely-used supervised learning algorithms include k-nearest neighbor, decision tree, random forest, neural network, support vector machine, Bayes' theory, and hidden Markov models. As we will use the neural network extensively in this dissertation, we present neural networks in details in the following.

Neural Network (NN) A neural network is a computing system made up of a large number of simple processing units, which operate in parallel to learn experiential knowledge from historical data [23]. The concept of neural networks is inspired by

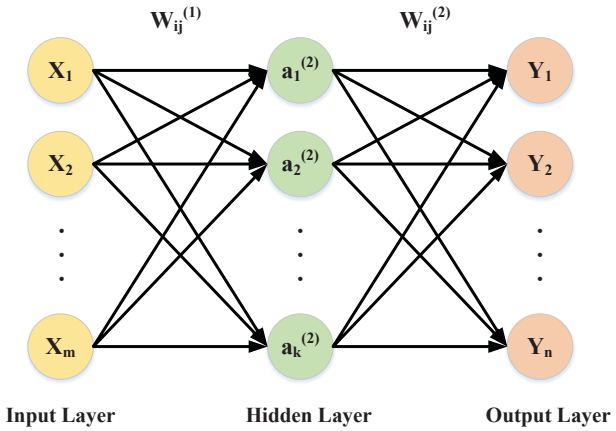

Figure 2.2: A basic neural network with three layers: an input layer, a hidden layer and an output layer. An input has m features (i.e., $X_1, X_2, \ldots, X_m$) and the input can be assigned to n possible classes (i.e., $Y_1, Y_2, \ldots, Y_n$). Also, W_{ij}^l denotes the variable link weight between the ith neuron of layer l and the jth neuron of layer $l+1$, and a_k^l denotes the activation function of the kth neuron in layer l.

the human brain, which uses basic components, known as neurons to perform highly complex, nonlinear and parallel computations. In a NN, its nodes are the equivalent components of the neurons in the human brain. These nodes use activation functions to perform nonlinear computations. The most frequently used activation functions are the sigmoid and the hyperbolic tangent functions [24]. Simulating the way neurons are connected in the human brain, the nodes in a NN are connected to each other by variable link weights.

A NN has many layers. The first layer is the input layer and the last layer is the output layer. Layers between the input layer and the output layer are hidden layers. The output of each layer is the input of the next layer and the output of the last layer is the result. By changing the number of hidden layers and the number of nodes in each layer, complex models can be trained to improve the performance of NNs. NNs are widely used in many applications, such as pattern recognition. The most basic

NN has three layers, including an input layer, a hidden layer and an output layer, which is shown in Fig. 2.2.

There are many types of neural networks, which are often divided into two training types, supervised or unsupervised [25]. In the following, we will give a detailed representation of supervised neural networks which have been applied in the field of SDN. In Subsection 2.1.2, self-organizing map, a representative type of unsupervised neural networks, will be described.

Random NN The random NN can be represented as an interconnected network of neurons which exchange spiking signals. The main difference between random NN and other neural networks is that neurons in random NN exchange excitatory and inhibitory spiking signals probabilistically. In random NN, the internal excitatory state of each neuron is represented by an integer, which is called "potential". The potential value of each neuron rises when it receives an excitatory spiking signal and drops when it receives an inhibitory spiking signal. Neurons whose potential values are strictly positive are allowed to send out excitatory or inhibitory spiking signals to other neurons according to specific neuron-dependent spiking rates. When a neuron sends out a spiking signal, its potential value drops one. The random NN has been used in classification and pattern recognition [26]. For a more insightful discussion on random NN, please refer to [26–28].

Deep NN Neural networks with a single hidden layer are generally referred to as shallow NNs. In contrast, neural networks with multiple hidden layers between the input layer and the output layer are called deep NNs [29–31]. For a long time, shallow NNs are often used. To process high-dimensional data and to learn increasingly complex models, deep NNs with more hidden layers and neurons are needed. However, deep NNs increase the training difficulties and require more computing resources. In

recent years, the development of hardware data processing capabilities (e.g., GPU and TPU) and the evolved activation functions (e.g., ReLU) make it possible to train deep NNs [32]. In deep NNs, each layer's neurons train a feature representation based on the previous layer's output, which is known as feature hierarchy. The feature hierarchy makes deep NNs capable of handling large high-dimensional datasets. Due to the multiple-level feature representation learning, compared to other machine learning techniques, deep NNs generally provide much better performance [32].

Convolutional NN Convolutional NN and recurrent NN are two major types of deep NNs. Convolutional NN [33, 34] is a feed-forward neural network. Local sparse connections among successive layers, weight sharing and pooling are three basic ideas of convolutional NN. Weight sharing means that weight parameters of all neurons in the same convolution kernel are same. Local sparse connections and weight sharing can reduce the number of training parameters. Pooling can be used to reduce the feature size while maintaining the invariance of features. The three basic ideas reduce the training difficulties of convolutional NNs greatly.

Recurrent NN In feed-forward neural networks, the information is transmitted directionally from the input layer to the output layer. However, recurrent NN [35, 36] is a stateful network, which can use internal state (memory) to handle sequential data. A typical recurrent NN and its unrolled form are shown in Fig. 2.3. X_t is the input at time step t. h_t is the hidden state at time step t. h_t captures information about what happened in all the previous time steps, so it is called "memory". Y_t is the output at time step t. U, V and W are parameters in the recurrent NN. Unlike a traditional deep NN, which uses different parameters at each layer, the recurrent NN shares the same parameters (i.e., U, V and W) across all time steps. This means that at each time step, the recurrent NN performs the same task, just with different

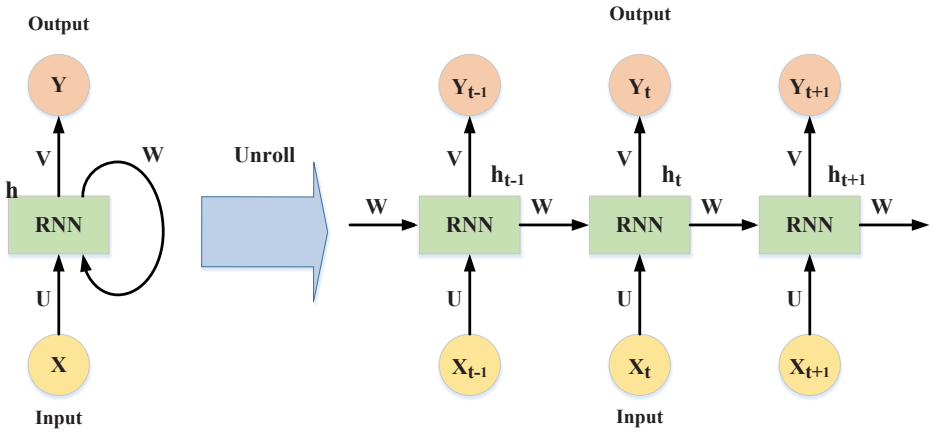

Figure 2.3: A typical recurrent NN and its unrolled form. X_t is the input at time step t. h_t is the hidden state at time step t. Y_t is the output at time step t. U, V and W are parameters in the recurrent NN.

inputs. In this way, the total number of parameters needed to be trained is reduced greatly. Long Short-Term Memory (LSTM) [37, 38] is the most commonly-used type of recurrent NNs, which has a good ability to capture long-term dependencies. LSTM uses three gates (i.e., an input gate, an output gate and a forget gate) to compute the hidden state.

2.1.2 Unsupervised Learning

In contrast to supervised learning, an unsupervised learning algorithm is given a set of inputs without labels (i.e., there is no output). Basically, an unsupervised learning algorithm aims to find patterns, structures, or knowledge in unlabeled data by clustering sample data into different groups according to the similarity between them. The unsupervised learning techniques are widely used in clustering and data aggregation [2, 3]. In the following, we will give a detailed representation of widely-used unsupervised learning algorithms, such as k-means and self-organizing map.

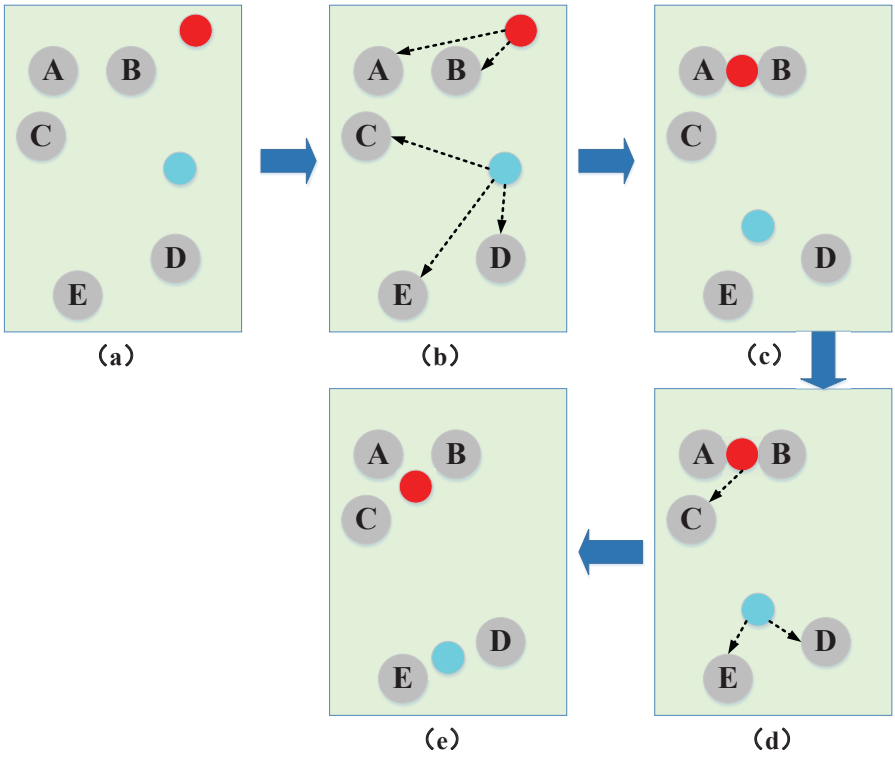

Figure 2.4: Example of k-means algorithm, for $k = 2$. (a) Randomly choosing two data points as two centroids; (b) label each node with the closest centroid, resulting that node A and B are a class, node C, D and E are another class; (c) assign new centroids; (d) label each node with the closest centroid again, resulting that node A, B and C are a class, node D and E are another class; (e) the algorithm is converged.

k-Means The k-means algorithm is a popular unsupervised learning algorithm, which is used to recognize a set of unlabeled data into different clusters. To implement the k-means algorithm, only two parameters (i.e., the initial dataset and the desired number of clusters) are needed. If the desired number of clusters is k, the steps to resolve node clustering problem by using k-means algorithm are: (a) initialize k cluster centroids by randomly choosing k nodes; (b) use a distance function to label each node with the closest centroid; (c) assign new centroids according to the current node memberships and (d) stop the algorithm if the convergence condition is valid, otherwise go back to step (b). An example procedure of k-means algorithm is shown

in Fig. 2.4. For a more insightful discussion on k-means, please refer to [2, 39].

Self-Organizing Map (SOM) SOM, also known as Self-Organizing Feature Map (SOFM), is one of the most popular unsupervised neural network models. SOM is often applied to perform dimensionality reduction and data clustering. In general, SOM has two layers, an input layer and a map layer. When SOM is used to perform data clustering, the number of neurons in the map layer is equal to the desired number of clusters. Each neuron has a weight vector. The steps to resolve data clustering problem by using SOM algorithm are: (a) initialize the weight vector of each neuron in the map layer; (b) choose a data sample from the training dataset; (c) use a distance function to calculate the similarity between the input data sample and all weight vectors. The neuron whose weight vector has the highest similarity is called the Best Matching Unit (BMU). The SOM algorithm is based on competitive learning, which means that there is only one BMU each time. (d) The neighborhood of the BMU is calculated. (e) The weight vectors of the neurons in the BMU's neighborhood (including the BMU itself) are adjusted towards the input data sample. (f) Stop the algorithm if the convergence condition is valid, otherwise go back to step (b). For a more insightful discussion on SOM, please refer to [40, 41].

2.1.3 Reinforcement Learning

Reinforcement learning is an important branch of machine learning, where an agent learns to take actions that would yield the most reward by interacting with the environment. Different from supervised learning, reinforcement learning cannot learn from samples provided by an experienced external supervisor. Instead, it has to operate based on its own experience despite that it faces with significant uncertainty about the environment.

Reinforcement learning is defined not characterizing learning methods, but by

characterizing a learning problem. Any method that is suitable for solving that problem can be considered as a reinforcement learning method [4]. A reinforcement learning problem can be described as an optimal control of a Markov Decision Process (MDP), however, state space, explicit transition probability and reward function are not necessarily required [42]. Therefore, reinforcement learning is powerful in handling tough situations that approach real-world complexity [8].

There are two outstanding features of reinforcement learning: trial-and-error search and delayed reward. Trial-and-error search means making trade-off between exploration and exploitation. The agent prefers to exploit the effective actions that have been tried in the past to produce rewards, but it also has to explore better new actions that may yield higher rewards in the future. The agent must try various actions and progressively favor those that earn the most rewards. The other feature of reinforcement learning is that the agent looks into a big picture, not just considering the immediate reward, but also the cumulative rewards in the long run, which is specified as value function.

Generally, reinforcement learning can be divided into model-free and model-based reinforcement learning, which is based on whether or not the environment elements are already known. Model-free reinforcement learning has recently been successfully applied to handle the deep neural network and value functions [5–8]. It can learn policies for tough tasks using the raw state representation directly as the input to the neural networks [9]. Contrastively, model-based reinforcement learn a model of the system with the help of supervised learning and optimize a policy under this model [9, 43, 44]. Recently, elements of model-based RL have been incorporated into model-free deep reinforcement learning to accelerate the learning rate without losing the strengths of model-free learning [9]. Let us briefly review the reinforcement learning from model-based to model-free case.

Let $X = \{x_1, x_2, ..., x_n\}$ be the state space, and $A = \{a_1, a_2, ...a_m\}$ be the action set. Based on the current state $x(t) \in X$, the agent takes an action $a(t) \in A$ on the environment and then the system transfers to a new state $x(t+1) \in X$ according to the transition probability $P_{x(t)x(t+1)}(a)$. The immediate reward is denoted as $r(x(t), a(t))$.

Taking into the long-term returns, the agent should not only consider the immediate rewards, but also the future rewards. The more into the future, the more discounts the reward may get. Thus, the future rewards are discounted with a discount factor $0 < \epsilon < 1$. The aim of the reinforcement learning agent is to find an optimal policy $a^* = \pi^*(x) \in A$ for each state x, which maximizes the cumulative reward over a long time. The cumulative discounted reward at state x can be expressed by the state value function:

$$V^{\pi}(x) = E\left[\sum_{t=0}^{\infty} \epsilon^t r(x(t), a(t)) | x(0) = x\right], \tag{2.1}$$

where E denotes the expectation, and it is considered over an infinite time horizon.

Due to the Markov property, i.e., the state at the subsequent time instant is only determined by the current state, irrelevant to the former states, the value function can be rewritten as

$$V^{\pi}(x) = R(x,\pi(x)) + \epsilon \sum_{x' \in X} P_{xx'}(\pi(x)) V^{\pi}(x'), \tag{2.2}$$

where $R(x,\pi(x))$ is the mean value of the immediate reward $r(x,\pi(x))$, and $P_{xx'}(\pi(x))$ is the transition probability from x to x', when action $\pi(x)$ is executed. The optimal policy π^* follows Bellman's criterion

$$V^{\pi^*}(x) = \max_{a \in A}\left[R(x,a) + \epsilon \sum_{x' \in X} P_{xx'}(a) V^{\pi^*}(x')\right]. \tag{2.3}$$

Given the reward R and transition probability P, the optimal policy can be obtained.

When R and P are unknown, Q-learning is one of the most widely-used strategies to determine the best policy π^*. Q-Learning belongs to model-free reinforcement learning algorithms. The most important component of Q-learning algorithms is to properly and efficiently estimatethe Q value. The Q function can be implemented simply by a look-up table, or by a function approximator, sometimes a nonlinear approximator, such as a neural network. The agent that uses a neural network to represent Q function is called Q-network, which is denoted as$Q(x, a; \theta)$. The parameter θ stands for the weights of the neural network, and the Q-network is trained by updating θ at each iteration to approximate the real Q values. A state-action function, i.e., Q-function is defined as

$$Q^{\pi}(x, a) = R(x, a) + \epsilon \sum_{x' \in X} P_{xx'}(a) V^{\pi}(x^{'}), \tag{2.4}$$

which represents the discounted cumulative reward when action a is performed at state x and continues optimal policy from that point on.

The maximum Q function will be

$$Q^{\pi^*}(x, a) = R(x, a) + \epsilon \sum_{x' \in X} P_{xx'}(a) V^{\pi^*}(x^{'}), \tag{2.5}$$

then the discounted cumulative state function can be written as

$$V^{\pi^*}(x) = \max_{a \in A} \left[Q^{\pi^*}(x, a) \right]. \tag{2.6}$$

Up to now, the objective can change from finding the best policy to finding the proper Q-function. Usually, Q-function is obtained in a recursive manner using the

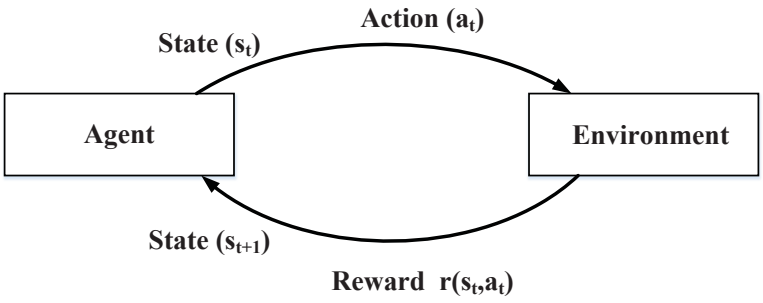

Figure 2.5: A basic diagram of a RL system. The agent takes an action according to the current state and then receives a reward. $r(\mathrm{s}_t, a_t)$ denotes the immediate reward that the agent receives after performing an action a_t at the state s_t.

available information $(x, a, r, x^{'}, a^{'})$, i.e., the state x, the immediate reward r, the action a at the current time instant t, and the state $x^{'}$ and action $a^{'}$ at the next time instant $t+1$. The Q-function is updated as

$$Q_{t+1}(x,a) = Q_t(x,a) + \alpha\left(r + \epsilon[\max_{a'} Q_t(x^{'}, a^{'})] - Q_t(x,a)\right), \tag{2.7}$$

where α is the learning rate. Utilizing proper learning rate, $Q_t(x,a)$ will definitely converges to $Q^*(x,a)$ [45].

2.1.4 Deep Q-learning

While neural networks allow for great flexibility, they do so at the cost of stability when it comes to Q-Learning, which is interpreted in [8]. Deep Q-network that uses deep neural networks instead of approximating the Q-function is proposed recently, and it is proven to be more advantageous with greater performance and more robust learning [8]. To transform an ordinary Q-network into a deep Q-network, three improvements have been implemented.

- Replacing ordinary neural networks with advanced multi-layer deep convolutional networks, which utilize hierarchical layers of tiled convolutional filters to exploit the local spatial correlations, and make it possible to extract high-level features from raw input data [42, 46].

- Utilizing Experience Replay, which stores its interaction experience tuple $e(t) = (x(t), a(t), r(t), x(t+1))$ at time t into a replay memory $D(t) = \{e(1), ..., e(t)\}$, and then randomly samples batches from the experience pool to train the deep convolutional network's parameters rather than directly using the consecutive samples as in Q-learning. This allows the network to learn from more various past experiences, and restrains the network from only focusing on what it is immediately doing.

- Adopting a second network, to generate the target Q values that are used to calculate the loss for each action during the training procedure. One network for both estimations of Q values and target Q values would result in falling into feedback loops between the target and estimated values. Thus, in order to stabilize the training, the target network's weights are fixed and periodically updated.

The deep Q-function is trained towards the target value by minimizing the loss function $L(\theta)$ at each iteration, and the loss function can be written as

$$L_i(\theta_i) = E[(y_i - Q(x, a; \theta_i)^2)], \tag{2.8}$$

where $y_i = r + \varepsilon \max_{a'} Q(x^{'}, a^{'}; \theta_i^{-})$. Here, the weights θ_i^{-} are updated as $\theta_i^{-} = \theta_{i-G}$, i.e., the weights update every G time steps in the deep Q-networks instead of $\theta_i^{-} = \theta_{i-1}$.

2.1.5 Beyond Deep Q-learning

Since deep Q-learning is first proposed, great efforts have been made for even greater performance and higher stability. Here, we briefly introduce two recent improvements: Double DQN [47] and Dueling DQN [5].

Double DQN

In regular DQN, both choosing an action and evaluating the chosen action use the max over the Q values, which would lead to over-optimistic Q value estimation. To relieve the over-estimation problem, the target value in double DQN is designed and updated as

$$y_i^{double} = r + \varepsilon Q(x^{'}, arg\max_{a^{'}} Q(x^{'}, a^{'}; \theta_i); \theta_i^{-}), \tag{2.9}$$

where the action choice is decoupled from the target Q-value generation. This simple trick makes the over-estimation significantly reduced, and the training procedure runs faster and more reliably.

Dueling DQN

The intuition behind Dueling DQN is that it is not always necessary to estimate the value of taking each available action. For some states, the choice of action makes no influence on what happens. Thus, in dueling DQN, the state-action value $Q(x, a)$ is decomposed into two components as follows,

$$Q(x, a) = V(x) + A(a). \tag{2.10}$$

Here, $V(x)$ is the value function, which simply represents how good it is to be in a given state x. $A(a)$ is the advantage function, which measures the relative importance

of a certain action compared with other actions. After $V(x)$ and $A(a)$ are separately computed, their values are combined back into a single Q-function at the final layer. This improvement would lead to better policy evaluation.

Combining both the above two techniques can achieve better performance and faster training speed. In the following section, the deep Q-network with the two improvements is utilized to find the optimal policy for a practical network application.

2.2 Integrated Networking, Caching and Computing in Wireless Mobile Networks

2.2.1 Motivations

New wireless mobile applications, such as natural language processing, augmented reality and face recognition, have emerged rapidly in recent years. However, conventional wireless networks solely focusing on communication are no longer capable of meeting the demand raised by such applications not only on high data rates, but also on high caching and computing capabilities. Although the pivotal role of communication in wireless networks can never be overemphasized, the growth in communication alone is not sustainable any longer. On the other hand, recent advances in communication and information technologies have fueled a plethora of innovations in various areas, including networking, caching and computing, which have the potential to profoundly impact our society via the development of smart homes, smart transportation, smart cities [48], etc.

Therefore, the integration of networking, caching and computing into one system becomes a natural trend [48]. By incorporating caching functionality into the network, the system can provide native support for highly scalable and efficient content retrieval, and meanwhile, duplicate content transmissions within the network

can be reduced significantly. As a consequence, mobility, flexibility and security of the network can be considerably improved. On the other hand, the incorporation of computing functionality endows the network with powerful capabilities of data processing, hence enabling the execution of computationally intensive applications within the network. By offloading mobile devices' computation tasks (entirely or partially) to resource-rich computing infrastructures in the vicinity or in remote clouds, the task execution time can be considerably reduced, and the local resources of mobile devices, especially battery life, can be conserved, thus enriching the computing experience of mobile users. Moreover, networking, caching and computing functionalities can complement and reinforce each other by interactions. For instance, some of the computation results can be cached for future use, thus alleviating the backhaul workload. On the other hand, some cached content can be transcoded into other versions to better suit specific user demands, thus economizing storage spaces and maximizing the utilization of caching.

Despite the potential vision of integrated networking, caching and computing systems, a number of significant research challenges remain to be addressed before widespread application of the integrated systems, including the latency requirement, bandwidth constraints, interfaces, mobility management, resource and architecture tradeoffs, convergence as well as non-technical issues such as governance regulations, etc. Particularly, the resource allocation issue, in which the management and allocation of three types of resources should be jointly considered to effectively and efficiently serve user requirements, is especially challenging. In addition, non-trivial security challenges are induced by a large number of intelligent devices/nodes with self adaptation/context awareness capabilities in integrated systems. These challenges need to be broadly tackled through comprehensive research efforts.

2.2.2 Networking

The communication and networking technologies of the fifth generation (5G) wireless telecommunication networks are being standardized at full speed along the time line of International Mobile Telecommunications, which is proposed by the International Telecommunication Union (ITU). Consequently, unanimous agreement on the prospect of a new, backward-incompatible radio access technology is emerging in the industry.

Significant improvements on data rate and latency performance will be powered by logical network slices [49] in the 5G communication environment. Furthermore, network flexibility will be greatly promoted by dynamic network slices, therefore providing the possibility of the emergence of a variety of new network services and applications. Due to the fact that the key problems of network slices are how to properly slice the network and at what granularity, it is not difficult to predict that a number of already existing technologies, such as software defined networking (SDN) [50] and network functions virtualization (NFV) [51], will be taken into consideration.

SDN can enable the separation between the data plane and the control plane, therefore realizing the independence of the data plane capacity from the control plane resource, which means high data rates can be achieved without incurring overhead upon the control plane. Meanwhile, the separation of the data and control planes endows high programmability, low-complexity management, convenient configuration and easy troubleshooting to the network. The controller in the control plane and the devices in the data plane are bridged by a well-defined application programming interface (API), of which a well known example is OpenFlow. The instructions of a controller to the devices in the data plane are transmitted through flow tables, each of which defines a subset of the traffic and the corresponding action. Due to the advantages described above, SDN is considered promising on providing programmable

network control in both wireless and wired networks.

NFV presents the technology that decouples the services from the network infrastructures that provide them, therefore maximizing the utilization of network infrastructure by offering the possibility that services from different service providers can share the same network infrastructure [51]. Furthermore, easy migration of new technology in conjunction with legacy technologies in the same network can be realized by isolating part of the network infrastructure [50].

2.2.3 Caching

As described above, the spectral efficiency (SE) of wireless communication radio access networks has been increased greatly by ultra-dense small cell deployment. However, this has brought up another issue: the backhaul may become the bottleneck of the wireless communication system due to the tremendous and ever increasing amount of data being exchanged within the network [52]. On the other hand, building enough high speed backhauls linking the core network and the small cells which are growing in number could be exceptionally expensive. Being stimulated by this predicament, research efforts have been dedicated to caching solutions in wireless communication systems [53]. By storing Internet contents at infrastructures in radio access networks, such as base stations (BSs) and mobile edge computing (MEC) servers, caching solutions enable the reuse of cached contents and the alleviation of backhaul usage. Therefore, the problem has been transferred by caching solutions from intensive demands on backhaul connections to caching capability of the network.

However, the original intention of Internet protocols is providing direct connections between clients and servers, while the caching paradigm calls for a usage pattern based on distribution and retrieval of content. This contradiction has led to a degradation of scalability, availability and mobility. To address this issue, Content Delivery Networks

(CDNs) and Peer-to-Peer (P2P) networks have been proposed in application layers, as first attempts to confer content distribution and retrieval capabilities to networks by utilizing the current storage and processing network infrastructures.

CDNs employ clusters of servers among the Internet infrastructures and serve the UE with the replications of content that have been cached in those servers. The content requests generated by UE are transmitted through the Domain Name Servers (DNSs) of the CDN to the nearest CDN servers that hold the requested content, in order to minimize latency. The decision on which server is chosen to store the replication of content is made upon a constant monitoring and load balancing of data traffic in the network.

P2P networks rely on the storage and forwarding of replications of content by UE. Each UE can act as a caching server in P2P networks. In P2P networks, the sources of content are called peers. Instead of caching a complete content, each peer may store only a part of the content, which is called a chunk. Thus, the content request of a UE is resolved and directed to several peers by a directory server. Each peer will provide a part of the requested content upon receipt of the request.

Although CDN and P2P do give a solution for content distribution and retrieval, due to the fact that they solely operate on application layers and the commercial and technological boundaries that they are confined to, the performances of these two techniques are not ideal enough to fulfil the demands on network caching services .

Serving as an alternative to CDN and P2P networking, Information-Centric Networking (ICN) emphasizes information dissemination and content retrieval by operating a common protocol in a network layer, which can utilize current storage and processing network infrastructures to cache content [53]. In general, depending on the location of caches, caching approaches of ICN can be categorized as on-path caching and off-path caching. On-path caching concerns the delivery paths when considering

caching strategies, and hence are usually aggregated with the forwarding mechanisms of ICN. On the other hand, off-path caching solely focuses on the storage and delivery of content, regardless of the delivery path.

2.2.4 Computing

As the prevalence of smartphones is dramatically growing, new mobile applications, such as face recognition, natural language processing, augmented reality, etc. are emerging. This leads to a constantly increasing demand on computational capability. However, due to size and battery life constraints, mobile devices tend to fail in fulfilling this demand. On the other hand, powered by network slicing, SDN and NFV, cloud computing (CC) functionalities are incorporated into mobile communication systems, bringing up the concept of mobile cloud computing (MCC), which leverages the rich computation resources in the cloud for user computation task execution, by enabling computation offloading through wireless cellular networks. After computation offloading, the cloud server will create a clone for each piece of user equipment (UE) individually, then the computation task of the UE will be performed by the clone on behalf of that UE. Along with the migration of computation tasks from UE to a resourceful cloud, we are witnessing a networking paradigm transformation from connection-oriented networking to content-oriented networking, which stresses data processing, storage and retrieval capability, rather than the previous criterion that solely focuses on connectivity.

Nevertheless, due to the fact that the cloud is usually distant from mobile devices, the low-latency requirements of some latency-sensitive (real-time) applications may not be fulfilled by cloud computing. Moreover, migration of a large amount of computation tasks over a long distance is sometimes infeasible and uneconomical.

To tackle this issue, fog computing has been proposed to provide UE with proximity to resourceful computation servers. It is a distributed computing paradigm in which network entities with different computation and storage abilities and various hierarchical levels are placed within a short distance from the cellular wireless access network, connecting user devices to the cloud or Internet. It is worth noting that fog computing is not a replacement but a complement of cloud computing, due to the fact that the gist of fog computing is providing low-latency services to meet the demands of real-time applications, such as smart traffic monitoring, live streaming, etc. However, when the applications requiring a tremendous amount of computation or permanent storage are concerned, the fog computing infrastructures are only acting as gateways or routers for data redirection to the cloud computing framework.

To serve as a complement of cloudlet-based mobile cloud computing, a new MCC paradigm similar to fog computing, named MEC, is proposed [54]. Being placed at the edge of radio access networks, MEC servers can provide sufficient computation resources in physical proximity to UE, which guarantees the fulfilment of the demand of fast interactive response by low-latency connections. Therefore, mobile edge computing is envisioned as a promising technique to offer agile and ubiquitous computation augmenting services for mobile devices, by conferring considerable computational capability to mobile communication networks [54].

Chapter 3

Deep Reinforcement Learning for Cache-Enabled Opportunistic Interference Alignment Wireless Networks

3.1 Introduction

Interference alignment (IA) has been studied extensively as a revolutionary technique to tackle the interference issue in wireless networks [55]. IA exploits the cooperation of transmitters to design the precoding matrices, and thus eliminates the interferences. IA can benefit mobile cellular networks [56]. Due to the large number of users in cellular networks, multiuser diversity has been studied in conjunction with IA, called opportunistic IA, which further improves the network performance [57–60].

Recently, *wireless proactive caching* has attracted great attentions from both academia and industry [10, 11]. By effectively reducing the duplicate content transmissions in networks, caching has been recognized as one of the promising techniques for future wireless networks to improve spectral efficiency, shorten latency, and reduce energy consumption [12–14]. Based on the global traffic features, a few popular contents are requested by many users during a short time span, which accounts for most

of the traffic load. Therefore, proactively caching the popular contents can remove the heavy burden of the backhaul links.

Jointly considering these two important technologies, caching and IA, can be beneficial in IA-based wireless networks [61, 62]. The implementation of IA requires the channel state information (CSI) exchange among transmitters, which usually relies on the backhaul link. The limited capacity of backhaul link has significant impacts on the performance of IA [63]. Caching can relieve the traffic loads of backhaul links, thus the saved capacity can be used for CSI exchange in IA. In [61], the authors investigate the benefits of caching and IA in the context of mutiple-input and multiple-output (MIMO) interference channels, and maximize the average transmission rate by optimizing the number of the active transceiver pairs. In [62], it is shown that by properly placing the content in the transmitters' caches, the IA gain can be increased.

Although some excellent works have been done on caching and IA, most of these previous works assume that the channel is block-fading channel or invariant channel, where the estimated CSI of the current time instant is simply taken as the predicted CSI for the next time instant. Considering the time-varying nature of wireless environments, this kind of memoryless channel assumption is not realistic [64], especially in vehicular environments due to high mobility [65]. In addition, it is difficult to obtain the perfect CSI due to channel estimation errors, communication latency and backhaul link constraints [66, 67].

In this chapter, we consider realistic time-varying channels, and propose a novel *deep reinforcement learning* approach in cache-enabled opportunistic IA wireless networks. Deep reinforcement learning is a kind of machine learning, which is a powerful tool to process wireless data [68]. The distinct features of this chapter are as follows.

- Cache-enabled opportunistic IA is studied under the condition of time-varying

channel coefficients. The channel is formulated as a finite-state Markov channel (FSMC), which has been widely accepted in the literature to characterize the correlation structure of the fading process [15, 16, 69]. Considering FSMC models may enable significant performance improvements over the schemes with memoryless channel models.

- The complexity of the system is very high when we consider realistic FSMC models. Therefore, we propose a novel deep reinforcement learning approach in this chapter. Deep reinforcement learning is an advanced reinforcement learning algorithm that uses deep Q network to approximate the Q value-action function [8], and it has been used in wireless networks to improve the performance [18, 19]. Deep reinforcement learning is used in this chapter to obtain the optimal IA user selection policy in cache-enabled opportunistic IA wireless networks.

- We use Google TensorFlow to implement deep reinforcement learning. The visualization of the deep Q network model is presented. Simulation results with different system parameters are presented to show the effectiveness of the proposed scheme. It is illustrated that the performance of cache-enabled opportunistic IA networks in terms of the network's sum rate and energy efficiency can be significantly improved by using the proposed deep reinforcement learning approach.

The rest of this chapter is organized as follows. In Section 3.2, the system model is presented. In Section 3.3, the cache-enabled opportunistic IA network is formulated as a deep reinforcement learning process. Simulation results are discussed in Section 3.4. Finally, chapter summary is presented in Section 3.5.

3.2 System Model

In this section, the model of IA is described, followed by the time-varying channel. Then, cache-equipped transmitters are described.

3.2.1 Interference Alignment

IA is a revolutionary interference management technique, which theoretically enables the network's sum rate grow linearly with the cooperative transmitter and receiver pairs. That is to say, each user can obtain the capacity $\frac{1}{2}\log(\text{SNR}) + o(\log(\text{SNR}))$, which has nothing to do with the interferences [70]. Actually, SNR plays a crucial role in determining the IA results. Cadambe and Jafar pointed out that IA performs better at very high SNR, and suffers from low quality at moderate SNR levels [70]. Meanwhile higher and higher SNR is required to approach IA network's theoretical maximum sumrate as the number of IA users increases [55]. Thus, there exist competitions among users for accessing to IA network.

Consider a K-user MIMO interference channel. $N_t^{[k]}$ and $N_r^{[k]}$ antennas are equipped at the kth transmitter and receiver, respectively. The degree of freedom (DoF) of the kth user is denoted as $d^{[k]}$. The received signal at the kth receiver can be written as

$$\begin{aligned}\mathbf{y}^{[k]}(t) = &\mathbf{U}^{[k]\dagger}(t)\mathbf{H}^{[kk]}(t)\mathbf{V}^{[k]}(t)\mathbf{x}^{[k]}(t) \\ &+\sum_{j=1,j\neq k}^{K}\mathbf{U}^{[k]\dagger}(t)\mathbf{H}^{[kj]}(t)\mathbf{V}^{[j]}(t)\mathbf{x}^{[j]}(t)+\mathbf{U}^{[k]\dagger}(t)\mathbf{z}^{[k]}(t),\end{aligned} \tag{3.1}$$

where the first term at the right side represents the expected signal, and the other two terms mean the inter-user interference and noise, respectively. $\mathbf{H}^{[kj]}(t)$ is the $N_r^{[k]} \times N_t^{[j]}$ matrix of channel coefficients from the jth transmitter to the kth receiver over the time slot t. Each element of $\mathbf{H}^{[kj]}(t)$ is independent and identically

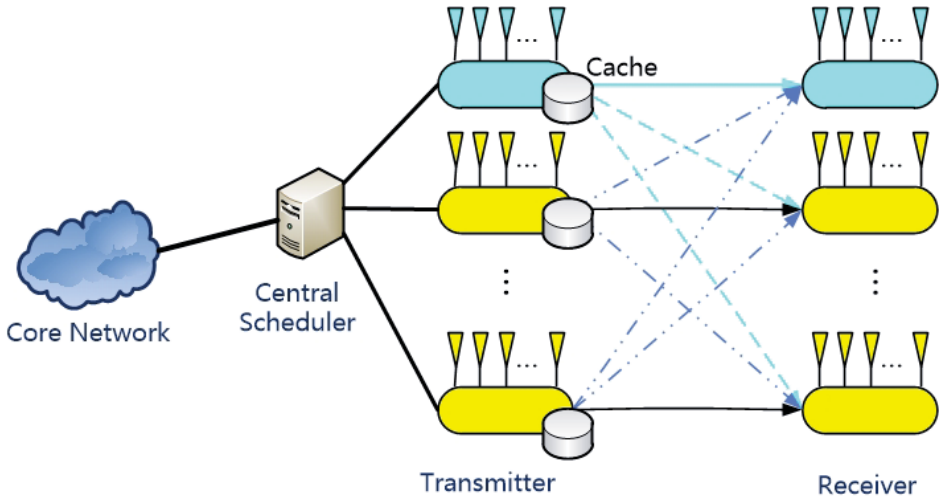

Figure 3.1: System model of a cache-enabled opportunistic IA wireless network.

distributed (i.i.d) complex Gaussian random variable, with zero mean and unit variance. $\mathbf{V}^{[k]}(t)$ and $\mathbf{U}^{[k]}(t)$ are the unitary $N_t^{[k]} \times d^{[k]}$ precoding matrix and $N_r^{[k]} \times d^{[k]}$ interference suppression matrix of the kth user, respectively. $\mathbf{x}^{[k]}(t)$ and $\mathbf{z}^{[k]}(t)$ are the transmitted signal vector of $d^{[k]}$ dimensions and the $N^{[k]} \times 1$ additive white Gaussian noise (AWGN) vector whose elements have zero mean and σ^2 variance at the kth receiver, respectively.

The interference can be perfectly eliminated only when the following conditions can be satisfied

$$\mathbf{U}^{[k]\dagger}(t)\mathbf{H}^{[kj]}(t)\mathbf{V}^{[j]}(t) = 0, \ \forall j \neq k, \tag{3.2}$$

$$\text{rank}\left(\mathbf{U}^{[k]\dagger}(t)\mathbf{H}^{[kk]}(t)\mathbf{V}^{[k]}(t)\right) = d^{[k]}. \tag{3.3}$$

Under this assumption, the received signal at the kth receiver can be rewritten as

$$\mathbf{y}^{[k]}(t) = \mathbf{U}^{[k]\dagger}(t)\mathbf{H}^{[kk]}(t)\mathbf{V}^{[k]}(t)\mathbf{x}^{[k]}(t) + \mathbf{U}^{[k]\dagger}(t)\mathbf{z}^{[k]}(t). \tag{3.4}$$

To meet Condition (3.2), the global CSI is required at each transmitter. Each transmitter can estimate its local CSI (i.e., the direct link), but the CSI of other links can only be obtained by CSI share with other transmitters via the backhaul link [61]. Thus, in IA network, the backhaul link is more than a pipeline for connecting with Internet. The limited capacity should be made optimum use of. The recent advances focus on the benefits of edge caching, which is capable to decrease the data tranfer and leave more capacity for CSI share. The detail is described in the following subsection. In this chapter, we assume the total backhaul link capacity of all the users is C_{total}, and the CSI estimation is perfect with no errors and no time delay.

3.2.2 Cache-equipped Transmitters

In the era of explosive information, the vast amount of content makes it impossible for all of them gain popularity. As a matter of fact, only a small fraction becomes extensively popular. That means certain content may be requested over and over during a short time span, which gives rise to the network congestion and transmission delay. We assume that each transmitter is equipped with a cache unit that has certain amount of storage space. The stored content may follow a certain popularity distribution.

For consistency, the cache of each transmitter stores the same content, usually the web content, and thus alleviating the backhaul burden and shorten delay time. In [71], the authors survey on the existing methods for predicting the popularity of different types of web content. Specifically, they show that different types of content follow different popularity distributions. For example, the popularity growth of online videos complies with power-law or exponential distributions, that of the online news can be represented by power-law or log-normal distributions, etc. Based on the content popularity distribution and cache size, cache hit probability P_{hit} and cache

miss probability P_{miss} can be derived [61]. In this chapter, the specific popularity distribution is not the focus, and we just concentrate on two states, whether the requested content is within the cache or not. We describe the two states as $\Lambda = \{0, 1\}$, where 0 means the requested content is not within the cache, and 1 indicates it is within the cache.

In this chapter, we consider an MIMO interference network with limited backhaul capacity and caches equipped at the transmitter side, as illustrated in Fig. 3.1. There is a central scheduler who is responsible for collecting the channel state and cache status from each user, scheduling the users and allocating the limited resources. All the users are connected to the central scheduler via a backhaul link for CSI exchange and Internet connection, and the total capacity is limited.

When we consider realistic FSMC models (in Section IV), the complexity of the system is very high, which makes it difficult to be solved using traditional approaches. In this chapter, we used a novel reinforcement learning approach to solve the optimization problem in cache-enabled opportunistic IA wireless networks. In the following section, we describe recent advances of deep reinforcement learning.

3.3 Problem Formulation

In this section, we first formulate the realistic time-varying channels as finite-state Markov channels (FSMC), and then we demonstrate how to formulate the cache-enabled IA network optimization problem as a deep Q-learning process, which can determine the optimal policy for IA user grouping.

3.3.1 Time-varying IA-based Channels

We model the realistic time-varying channels as FSMCs, which is an effective method to characterize the fading nature of wireless channels [15]. Specifically, the first-order FSMC is used in this chapter.

Here we consider a multi-user interference network with one DoF for each user, and the received signal to interference and noise ratio (SINR) is an important parameter that can be used to reflect the quality of a wireless channel. In an IA-based wireless network, if the interference is completely eliminated, the received SINR of the kth user at time instant t can be derived as follows,

$$\mathrm{SINR}^{[k]}(t) = \frac{|h^{[kk]}(t)|^2 P^{[k]}}{\sigma^2}, \tag{3.5}$$

where $h^{[kk]}(t) = \mathbf{u}^{[k]\dagger}(t)\mathbf{H}^{[kk]}(t)\mathbf{v}^{[k]}(t)$. On the other hand, if the interference is not perfectly eliminated, the received SINR is

$$\mathrm{SINR}^{[k]}(t) = \frac{|h^{[kk]}(t)|^2 P^{[k]}}{\sum_{j\neq k} |h^{[kj]}|^2 P^{[j]} + \sigma^2}, \tag{3.6}$$

where $h^{[kj]}(t) = \mathbf{u}^{[k]\dagger}(t)\mathbf{H}^{[kj]}(t)\mathbf{v}^{[j]}(t)$.

As we consider opportunistic IA networks with user scheduling, directly modeling the received SINR of a certain user as a Markov random variable is not appropriate. Due to the relationship of the received SINR and the channel coefficient, we model the channel coefficient after using IA, $|h^{[kj]}|^2$, as a Markov random variable. Here, $|h^{[kj]}(t)|^2$ is newly modelled because of its unknown distribution. We partition and quantize the range of $|h^{[kj]}|^2$ into H levels. Each level corresponds to a state of the Markov channel. This channel information is included in the system state, which will be discussed in detail in the following subsection. We consider T time slots over a period of wireless communications. Let us denote $t \in \{0, 1, 2, \ldots, T-1\}$ as the time

instant, and the channel coefficient varies from one state to another state when one time slot elapses.

3.3.2 Formulation of the Network's Optimization Problem

In our system, there are L candidates that want to join in the IA network to communicate wirelessly. We assume that the IA network size is always smaller than the number of candidates, which is in accordance with the fact that a large number of users expect wireless communications anytime and anywhere. As aforementioned, the value of SNR affects the performance of interference alignment, and the candidates who occupy the better channels are more advantageous for accessing to the IA network. Therefore, we make an action at each time slot to decide which candidates are the optimal users for constructing an IA network based on their current states.

Here, a central scheduler is responsible for acquiring each candidate's CSI and cache status, then it assembles the collected information into a system state. Next, the controller sends the system state to the agent, i.e., the deep Q network, and then the deep Q network feeds back the optimal action $\arg\max_{\pi} Q^*(x, a)$ for the current time instant. After obtaining the action, the central scheduler will send a bit to inform the users to be active or not, and the corresponding precoding vector will be sent to each active transmitter. The system will transfer to a new state after an action is performed, and the rewards can be obtained according to the reward function.

Inside the deep Q network, the replay memory stores the agent's experience of each time slot. The Q network parameter θ is updated at every time instant with samples from the replay memory. The target Q network parameter θ^- is copied from the Q network every N time instants. The ε-greedy policy is utilized to balance the exploration and exploitation, i.e., to balance the reward maximization based on the knowledge already known with trying new actions to obtain knowledge unknown.

Algorithm 1 Deep reinforcement learning algorithm in cache-enabled IA networks.

Initialization.

Initialize replay memory.
Initialize the Q network parameters with θ.
Initialize the target Q network parameters with $\theta^{-} = \theta$.

For episode $k = 1, ..., K$ **do**:

Initialize the beginning state x.

For $t = 1, 2, 3..., T$ **do**:

Choose a random probability p.
Choose $a(t)$ as,
if $p \geq \varepsilon$,
$a^{*}(t) = \arg\max_{a} Q(x, a, \theta)$,
otherwise,
randomly choose a solution $a(t) \neq a^{*}(t)$,
Execute $a(t)$ in the system. Observe the reward $r(t)$, and next state $x(t+1)$.
Store the experience $(x(t), a(t), r(t), x(t+1))$ in replay memory.
Get mini-batch of samples $(x(t), a(t), \zeta(t), x(t+1))$ from the replay memory.
Perform a gradient descent step on $(y - Q(x, a, \theta)^{2})$ with respect to θ.
Return the value of parameters θ in the deep Q network.

End for
End for

The training algorithm of the deep Q network is described in Algorithm *1*.

In order to obtain the optimal policy, it is necessary to identify the actions, states and reward functions in our deep Q learning model, which will be described in the next following subsections.

System State

The current system state $x(t)$ is jointly determined by the states of L candidates. The system state at time slot t is defined as,

$$\begin{aligned} x(t) = \{&|h^{[11]}(t)|^2, |h^{[12]}(t)|^2, \ldots, |h^{[kj]}(t)|^2, \ldots, |h^{[LL]}(t)|^2, \\ &c_1(t), c_2(t), \ldots, c_l(t), \ldots, c_L(t)\}. \end{aligned} \tag{3.7}$$

Here, there are two components in the state: the channel coefficient after being processed by IA technique $|h^{[kj]}(t)|^2$ and the cache state $c_i(t)$. The cache state $c_l(t) \in \Lambda = \{0, 1\}$, the index l means the lth candidate, and $l = 1, 2, \ldots, L$.

The number of possible system states can be very large. Due to the curse of dimensionality, it is difficult for traditional approaches to handle our problem. Fortunately, deep Q network is capable of successfully learning directly from high-dimensional inputs [8], thus it is proper to be used in our system.

System Action

In the system, the central scheduler has to decide which candidates to be set active, and the corresponding resources will be allocated to the active users.

The current composite action $a(t)$ is denoted by

$$a(t) = \{a_1(t), a_2(t), \ldots, a_L(t)\}, \tag{3.8}$$

where $a_l(t)$ represents the control of the lth candidate, i.e., $a_l(t) = 0$ means the candidate l is passive (not selected) at time slot t, and $a_l(t) = 1$ means it is active (selected). Due to the constraint of IA, the condition $\sum_{l=1}^{L} a_l(t) \geq 3$ must be satisfied.

Reward Function

The system reward represents the optimization objective, and we take the objective to maximize the IA network's throughput, and the reward function of the lth candidate is defined as,

$$
r_l(t) = \begin{cases} a_l(t) \log_2 \left(1 + \dfrac{\left|h^{[ll]}(t)\right|^2 P^{[l]} x_l}{\sum\limits_{j=1, j\neq l}^{L} a_j(t) \left|h^{[lj]}(t)\right|^2 P^{[j]} x_j + \sigma^2} \right), \\ \qquad\qquad \text{if } c_l(t) = 1, \\ a_l(t) \min \left\{ \left\lfloor \dfrac{1}{\sum\limits_{i=1}^{L} a_i(t)} (C_{total} - C_c \sum_{i=1}^{L} a_i(t)) \right\rfloor, \right. \\ \qquad \left. \log_2 \left(1 + \dfrac{\left|h^{[ll]}(t)\right|^2 P^{[l]} x_l}{\sum\limits_{j=1, j\neq l}^{L} a_j(t) \left|h^{[lj]}(t)\right|^2 P^{[j]} x_j + \sigma^2} \right) \right\} \\ \qquad\qquad \text{if } c_l(t) = 0, \end{cases} \tag{3.9}
$$

where $\sum\limits_{i=1}^{L} a_i(t) \geq 3$ is a condition for IA network, C_{total} is the total capacity of the backhaul link, and C_c is the reserved capacity allocated to each active user to share CSI with other active users. For the lth candidate, if the requested content is not in

the local cache, it can only acquire the content through the backhaul link, and equal capacity (the total capacity minus the total capacity for CSI exchange) is allocated among the active users. If the requested content is within the cache, the lth candidate can get the maximum rate that an IA user can achieve. We assume the realistic IA situations that the interference cannot be perfectly eliminated, i.e., the interference leakage from other non-direct channels remains. Thus, the reward of the lth candidate is determined by the integrated system's state, including the states of both the direct and non-direct links.

The immediate system reward is the sum of all the candidates' immediate rewards, i.e., $r(t) = \sum_{n=1}^{L} r_n(t)$. The central scheduler gets $r(t)$ in state $x(t)$ when action $a(t)$ is performed in time slot t. The goal of using deep Q network into our system model is to find a selection policy that maximizes the discounted cumulative rewards during the communication period T, and the cumulative reward can be expressed as

$$R = \max_{\pi} E\left[\sum_{t=0}^{T-1} \epsilon^t r(t)\right], \tag{3.10}$$

where ϵ^t approaches to zero when t is large enough. In practice, a threshold for terminating the process can be set.

The Effects of Imperfect CSI In (3.7) and (3.9), we assume perfect CSI. However, in practical networks, the implementation of IA requires the CSI exchange among transmitters on the backhaul link, which will cause the delay of CSI. The limited capacity of backhaul link can have dramatic effects on the performance of IA due to the delayed CSI. Suppose that $\mathbf{H}(t)$ is the matrix consisting of the accurate channel coefficients at time instant t. The delayed channel matrix is denoted as follows.

$\mathbf{H}(t-\tau)$ is the delayed channel matrix when CSI is delayed by τ duration. The

relation between the delayed CSI and the current CSI can be modeled as

$$\mathbf{H}(t) = \rho\mathbf{H}(t-\tau) + \sqrt{1-\rho^2}\boldsymbol{\delta} = \rho\widehat{\mathbf{H}}(t) + \sqrt{1-\rho^2}\boldsymbol{\delta}, \tag{3.11}$$

where $\widehat{\mathbf{H}}(t)$ is the matrix of the delayed channel coefficients at the time instant t. $\boldsymbol{\delta}$ has the same distribution with $\mathbf{H}(t)$ and $\widehat{\mathbf{H}}(t)$. ρ is the normalized autocorrelation function of a fading channel with motion at a constant velocity, and $0 \leq \rho \leq 1$. It can be seen that $\rho = 1$ corresponds to perfect CSI, whereas $\rho = 0$ represents no CSI. ρ is defined as follows.

$$\rho = \frac{\mathbb{E}\left[(\mathbf{H})_{ij}\left(\widehat{\mathbf{H}}\right)^*_{ij}\right]}{\sqrt{\mathbb{E}\left[\left|(\mathbf{H})_{ij}\right|^2\right]\mathbb{E}\left[\left|\left(\widehat{\mathbf{H}}\right)_{ij}\right|^2\right]}}. \tag{3.12}$$

The value of ρ depends on the time scale of channel variation, which can be defined by coherence time [72], and length of the delay τ. When the channel is under Rayleigh fading, ρ can be derived as follows [73]

$$\rho = J_0(2\pi f_d \tau), \tag{3.13}$$

where $J_0(.)$ is the zeroth order Bessel function of the first kind, and f_d is the Doppler frequency, which reflects the velocity of the transceivers.

Derivation of u and v Here we leverage the conventional iterative interference alignment [70] to derive the precoding vector $\mathbf{v}$ and decoding vector $\mathbf{u}$ for each user, which utilizes the reciprocity of wireless channels and aims at minimizing the total interference leakage at receivers.

In the forward direction of iterations, the total interference leakage at the lth

receiver caused by other users can be written as

$$I^{[l]} = \mathrm{Tr}\left[\mathbf{u}^{[l]\dagger}\mathbf{Q}^{[l]}\mathbf{u}^{[l]}\right], \tag{3.14}$$

where Tr[**A**] denotes the trace operator of matrix **A**, and

$$\mathbf{Q}^{[l]} = \sum_{i=1, i\neq l}^{L} P^{[i]}\mathbf{H}^{[li]}\mathbf{v}^{[i]}\mathbf{v}^{[i]\dagger}\mathbf{H}^{[li]\dagger}. \tag{3.15}$$

In the reverse direction of iterations, the total interference leakage at the ith receiver (i.e., the ith transmitter in the original direction) can be denoted as

$$\overleftarrow{I}^{[i]} = \mathrm{Tr}\left[\overleftarrow{\mathbf{u}}^{[i]\dagger}\overleftarrow{\mathbf{Q}}^{[i]}\overleftarrow{\mathbf{u}}^{[i]}\right], \tag{3.16}$$

where

$$\overleftarrow{\mathbf{Q}}^{[i]} = \sum_{l=1, l\neq i}^{L} P^{[l]}\overleftarrow{\mathbf{H}}^{[il]}\overleftarrow{\mathbf{v}}^{[l]}\overleftarrow{\mathbf{v}}^{[l]\dagger}\overleftarrow{\mathbf{H}}^{[il]\dagger}. \tag{3.17}$$

In (3.16) and (3.17), $\overleftarrow{\mathbf{v}}^{[l]} = \mathbf{u}^{[l]}$, $\overleftarrow{\mathbf{u}}^{[l]} = \mathbf{v}^{[l]}$, and $\overleftarrow{\mathbf{H}}^{[li]} = \mathbf{H}^{[il]\dagger}$.

In the original channel, the decoding vector $\mathbf{u}^{[l]}$ of the lth user, which can minimize the total interference leakage, can be updated as

$$\mathbf{u}^{[l]} = \boldsymbol{v}_i\left(\mathbf{Q}^{[l]}\right). \tag{3.18}$$

where $\boldsymbol{v}_i(\mathbf{X})$ means the eigenvector corresponding to the ith smallest eigenvalue of matrix **X**.

In the reverse channel, the precoding vector of the lth user is set as $\overleftarrow{\mathbf{v}}^{[l]} = \mathbf{u}^{[l]}$.

Algorithm 2 The Conventional Iterative Interference Alignment

Initialize the $N_t^{[l]} \times 1$ precoding vector $\mathbf{v}^{[l]}$ arbitrarily.

Execute the iteration

Calculate the interference covariance matrix $\mathbf{Q}^{[l]}$ for the lth receiver based on (3.15), $l = 1, 2, \ldots, L$.

Compute the decoding vector $\mathbf{u}^{[l]}$ for the lth receiver based on (3.18), $l = 1, 2, \ldots, L$.

Reverse the communication direction. Set the reversed precoding vector as the original decoding vector, i.e., $\overleftarrow{\mathbf{v}}^{[l]} = \mathbf{u}^{[l]}$, $l = 1, 2, \ldots, L$.

For the reversed communication direction, calculate matrix $\overleftarrow{\mathbf{Q}}^{[k]}$ for the new lth receiver based on (3.17), $l = 1, 2, \ldots, L$.

Compute the reversed decoding vector $\overleftarrow{\mathbf{u}}^{[l]}$ for the new lth receiver based on (3.19), $l = 1, 2, \ldots, L$.

Reverse the communication direction. Set the original precoding vector as the reversed decoding vector, i.e., $\mathbf{v}^{[l]} = \overleftarrow{\mathbf{u}}^{[l]}$, $l = 1, 2, \ldots, L$.

Continue till Convergence

Obtain the Interference alignment solutions as $\mathbf{v}^{[l]}$ and $\mathbf{u}^{[l]}$, $l = 1, 2, \ldots, L$.

The reverse decoding vector $\overleftarrow{\mathbf{u}}^{[l]}$ can be obtained as

$$\overleftarrow{\mathbf{u}}^{[l]} = \boldsymbol{v}_i \left(\overleftarrow{\mathbf{Q}}^{[l]} \right). \tag{3.19}$$

Then the reverse decoding vector $\overleftarrow{\mathbf{u}}^{[l]}$ is regarded as the precoding vector in the original channel, i.e., $\mathbf{v}^{[l]} = \overleftarrow{\mathbf{u}}^{[l]}$. This process iterates in such a way until convergence, and the process is summarized in Algorithm 2.

3.4 Simulation Results and Discussions

Computer simulations are carried out to demonstrate the performance of the proposed deep reinforcement learning approach to the optimization of cache-enabled opportunistic IA wireless networks. We use TensorFlow [74] in our simulations to implement deep reinforcement learning. In this section, we first introduce TensorFlow, followed by simulation settings. Then, simulation results are discussed.

3.4.1 TensorFlow

TensorFlow is an open source software library for expressing machine learning algorithms, and an implementation for executing such algorithms. TensorFlow has attracted great interests from both academia and industry for a variety of applications, such as speech recognition, Gmail, Google Photos, and search [74]. Originally developed by the Google Brain team for Google's research and production purposes, TensorFlow was later released under the Apache 2.0 open source license in 2015. TensorFlow is Google Brain's second generation machine learning system to replace its predecessor DistBelief for the implementation and deployment of large-scale machine learning models.

TensorFlow provides a Python API, as well as a less documented C++ API. The reference implementation of TensorFlow runs on single devices. Nevertheless, TensorFlow can run on multiple CPUs and GPUs for fast execution. A wide variety of hardware platforms can be used for TensorFlow, which takes computations and maps them onto different hardware platforms, ranging from mobile device platforms (e.g., Android and iOS) to modest-sized systems using single machines containing one or many GPU cards to large-scale systems running hundreds and thousands of GPUs.

TensorFlow is used to transform an ordinary Q-Network into a deep Q-Network (DQN) by making the following improvements:

(1) Extending a simple neural network to a multi-layer convolutional network. We utilize the tf.contrib.layers.convolution2d function to easily create a convolutional layer as follows: $convolution\_layer = tf.contrib.layers.convolution2d(in, num\_out, k\_size, stride, padding)$, where num_out is the number of filters applied to the previous layer, k_size refers to how large a window used to slide over the previous layer, $stride$ is the number of points skipped as we slide the window across the layer, and $padding$ indicates if the window just slides over the bottom layer or adds padding around it to ensure that the convolutional layer has the same dimensions as the previous layer. Please refer to the Tensorflow documentation [75] for more information.

(2) Implementing Experience Replay, which will allow our network to train itself using the stored memories from its experience. By keeping the experiences, the network can learn from a more varied sets of past experiences. We use a tuple of $< state, action, reward, nextstate >$ to store these experiences. In our DQN, a class is used to handle storing and retrieving memories.

(3) Utilizing a second "target" network, which is used to compute the target Q-values during the training procedure. The reason why two networks are used is as follows. The Q-networks values shift at every step of training, and the value estimations can easily spiral out of control, if we adjust our network values by using a constantly shifting set of values. Consequently, there will be feedback loops between the target and estimated Q-values, and the network can become destabilized. To address this issue, we fix the target networks weights, and they are periodically updated to the primary Q-networks values. By doing this, we can have a training process in a more stable manner.

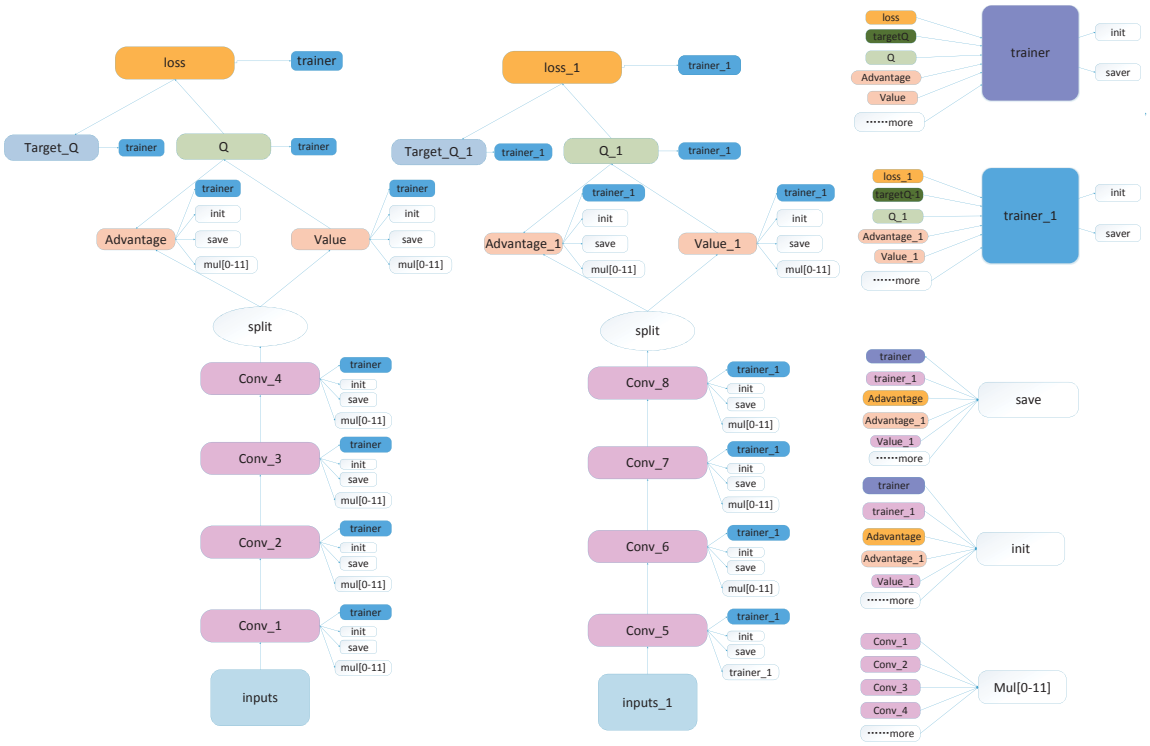

Figure 3.2: Visualization of the proposed deep reinforcement learning algorithm using TensorBoard.

3.4.2 Simulation Settings

In our simulations, we used a GPU-based server, which has 4 Nvidia GPUs with version GTX TITAN. The CPU is Intel Xeno E5-2683 v3 with 128G memory. The software environment we utilized is TensorFlow 0.12.1 with Python 2.7 on Ubuntu 14.04 LTS.

For performance comparison, our proposed OIA scheme is compared with four other schemes as

1. The same proposed scheme without (w.o.) caching.

2. An existing selection scheme without (w.o.) caching [76], in which invariant channels are assumed, i.e., the estimated channel coefficient of the current time

instant is simply taken as the predicted channel coefficient of the next time instant.

3. An existing selection scheme with (w.) caching [61], in which invariant channels are assumed, and it schedules the transceiver pairs to maximize the network throughput.

4. An existing scheme with (w.) caching [77], in which invariant channels are assumed, and power allocation strategy is performed aiming at maximizing the network throughput, however, no user selection is included.

In the simulations, we consider a cache-enabled opportunistic IA network. Due to the feasibility of IA [78], i.e., $N_t + N_r \geq d(L+1)$. Here, we set DoF d to be 1. Other setup parameters are: the bandwidth $B = 10$ MHz per transmitter, the total backhaul capacity $C_{total} = 60$ Mb/s, and the reserved capacity for sharing CSI $C_c = 2$ Mb/s per active user. The data rate for each user via the backhaul links is not smaller than 3Mb/s, that is to say, the maximal number of users simultaneously existing in the IA network is 12. The noise power σ^2 is set to be 0.1mW throughout the simulation. The normalized autocorrelation value ρ is set to be 0.99 in this chapter.

Based on the definition of system states, we quantize and uniformly partition the channel coefficients $|h^{[kj]}|^2$ into 10 levels with 9 boundary values as 10^{-6}, 0.3, 0.6, 0.9, 1.2, 1.5, 1.8, 2.1, 2.4. Uniformly setting the boundary values of FSMC is widely used in the literature (e.g., [79]) for simplicity. Nevertheless, the boundary values for a particular environment under certain criterion should be optimized for better performance [79]. We assume that the channel state transition probability is identical for all the candidates. In one simulation scenario, the transition probability of remaining in the same state is set to be 0.489, and the probability of transition to the adjacent state to be twice that of transition to a nonadjacent state. The channel

$$P_{channel} = \begin{pmatrix} 0.489 & 0.256 & 0.128 & 0.064 & 0.032 & 0.016 & 0.008 & 0.004 & 0.002 & 0.001 \\ 0.001 & 0.489 & 0.256 & 0.128 & 0.064 & 0.032 & 0.016 & 0.008 & 0.004 & 0.002 \\ 0.002 & 0.001 & 0.489 & 0.256 & 0.128 & 0.064 & 0.032 & 0.016 & 0.008 & 0.004 \\ 0.004 & 0.002 & 0.001 & 0.489 & 0.256 & 0.128 & 0.064 & 0.032 & 0.016 & 0.008 \\ 0.008 & 0.004 & 0.002 & 0.001 & 0.489 & 0.256 & 0.128 & 0.064 & 0.032 & 0.016 \\ 0.016 & 0.008 & 0.004 & 0.002 & 0.001 & 0.489 & 0.256 & 0.128 & 0.064 & 0.032 \\ 0.032 & 0.016 & 0.008 & 0.004 & 0.002 & 0.001 & 0.489 & 0.256 & 0.128 & 0.064 \\ 0.064 & 0.032 & 0.016 & 0.008 & 0.004 & 0.002 & 0.001 & 0.489 & 0.256 & 0.128 \\ 0.128 & 0.064 & 0.032 & 0.016 & 0.008 & 0.004 & 0.002 & 0.001 & 0.489 & 0.256 \\ 0.256 & 0.128 & 0.064 & 0.032 & 0.016 & 0.008 & 0.004 & 0.002 & 0.001 & 0.489 \end{pmatrix}.$$

state transition matrix is shown on the top of the next page. We change the channel state transition probability in other simulation scenarios.

The cache at each transmitter includes two states: existence and nonexistence of the requested content. The cache state transition probability matrix is set to be

$$P_{cache} = \begin{pmatrix} 0.6 & 0.4 \\ 0.4 & 0.6 \end{pmatrix}.$$

The detailed parameters in the proposed deep reinforcement learning algorithm are listed in Table 2. The visualization of the deep Q network model is presented in Fig. 3.2 using TensorBoard, which is a build-in module of TensorFlow. From the graph, it can be clearly visualized that double deep Q-networks are utilized, four convolutional layers are adopted in each deep Q-network, and the advantage and value functions are separately computed. The network models are saved and can be loaded to further train or test the models. In the simulations, some parameters are

Table 1: Simulation Parameter Values Used in Chapter 3.

Parameter	Value	Description
Mini-batch size	8	How many experience cases are used for each training step
Update frequency	4	The frequency to perform a training step
Experience replay buffer size	50,000	Training cases are randomly sampled from this number of the most recently experiences
Pre-training steps	10,000	How many steps of random actions are executed before the leaning begins, and the resulting experience are stored to populate the experience replay buffer
Total training steps	500,000	How many steps are used to train the network model
Discount factor	0.99	Discount factor used on the Q-function
Learning rate	0.0001	The learning rate used by AdamOptimizer
Initial exploration	1	The starting chance of random action in the ε-greedy exploration
Final exploration	0.1	The final chance of random action in the ε-greedy exploration
Anneling steps	10,000	How many training steps used to reduce ε from its starting value to the final value
Target network update rate	0.001	The rate to update target Q-network towards primary Q-network

changed to study the effects of these parameters.

3.4.3 Simulation Results and Discussions

Fig. 3.3 shows the convergence performance of the proposed scheme with different learning rates in the deep reinforcement learning algorithm. As we can see from the expression of the system state, the number of the possible states is $2^L \times H^{L^2}$, where 2^L is for the cache status and H^{L^2} is for the status of L^2 equivalent channel coefficients $|h^{[kj]}(t)|^2$ with $k, j = 1, 2, \ldots, L$. In addition to the number of possible states, the complexity of the deep Q learning algorithm depends on many other factors, including the number of actions, the state transition probability, and the rewards. Moreover, since deep Q learning utilizes deep learning to approximate the Q function, the convergence time is affected by many elements, such as the number of convolutional layers, the learning rate, the batch size, and some other parameters in the training process. From Fig. 3.3, we can observe that the average sum rate of the proposed scheme is very low at the beginning of the learning process. With the increase of the number of the episode, the average sum rate increases until it reaches a relatively stable value, which is around 300 Mbps in Fig. 3.3. This shows the convergence performance of the proposed scheme. We can also observe that the learning rate in the AdamOptimizer has effects on the convergence performance in Fig. 3.3. Specifically, the convergence is faster when the learning rate is 0.0001 compared to the case when the learning rate is 0.00001. However, a larger learning rate will result in local optimum instead of global optimum. Therefore, an appropriate learning rate should be chosen for a specific problem. In the rest of the simulations, we choose the learning rate of 0.0001.

Fig. 3.4 shows the effects of the mini-batch size for each gradient update in the deep reinforcement learning algorithm on the convergence performance. The

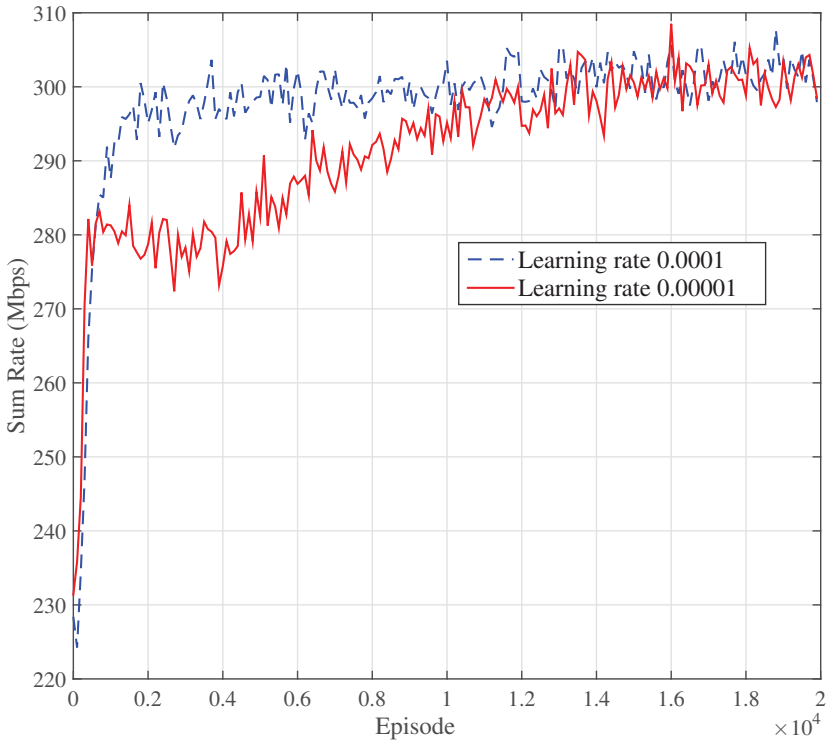

Figure 3.3: Convergence performance with different learning rates.

parameter of mini-batch size determines how many experience cases are used for each training step. We can see from Fig. 3.4 that the convergence is faster when the mini-batch size is 8 compared to the cases when the mini-batch size is 32 and 64. Similar to the learning rate parameter, an appropriate mini-batch size should be chosen for a specific problem. In the rest of the simulations, we choose the mini-batch size of 8.

Fig. 3.5 and Fig. 3.6 present the numbers of users that are existing simultaneously in IA network from the 19950 to 20000 episode, where the deep Q learning algorithm has converged. This represents the optimal numbers of users that access to IA network in different states. Fig. 3.5 compares the numbers of users of the proposed OIA scheme swith and without cache under the condition of SNR=30dB. From the figure, we can see that the proposed scheme tends to select less users to communicate when

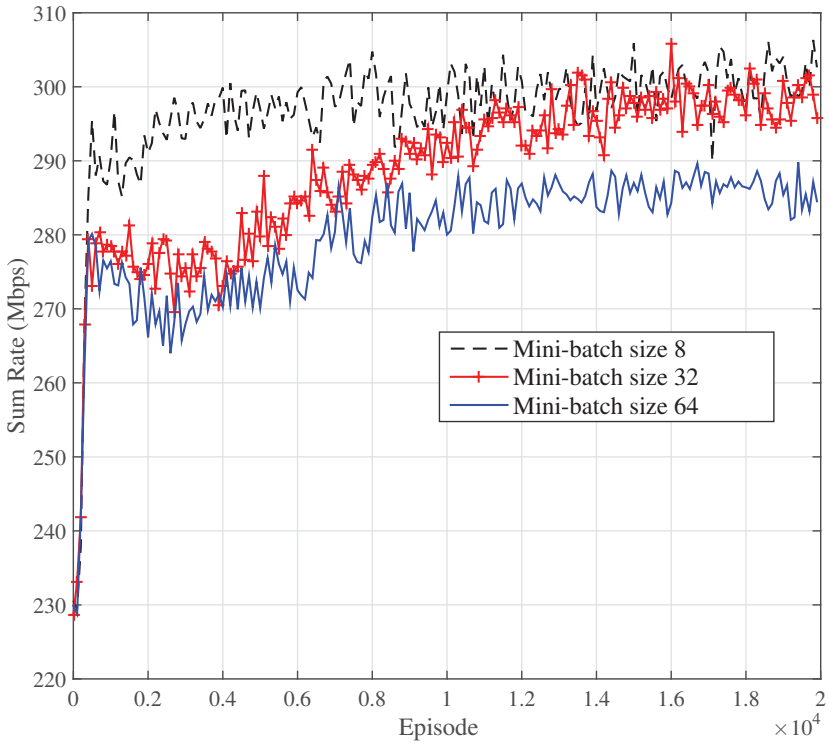

Figure 3.4: Convergence performance with different batch sizes.

the requested content is not cached. This is because at this situation, the sum rate have to rely on the backhaul link, and less users to access the IA network will occupy less backhaul capacity, which leads to larger sum rate when the total backhaul capacity is fixed. However, if the requested content is cached, more users to communicate means higher sum rate. Fig. 3.6 presents the numbers of users in IA network under the conditions of SNR=10dB and SNR=30dB, respectively. From this figure, we can conclude that when SNR is higher, the proposed scheme will select more users, this is because, even though more users means more backhaul capacity for CSI exchange, however, higher wireless data rate can be achieved, which results in much higher sum rate.

Fig. 3.7 shows the network's average sum rate with different state-transition

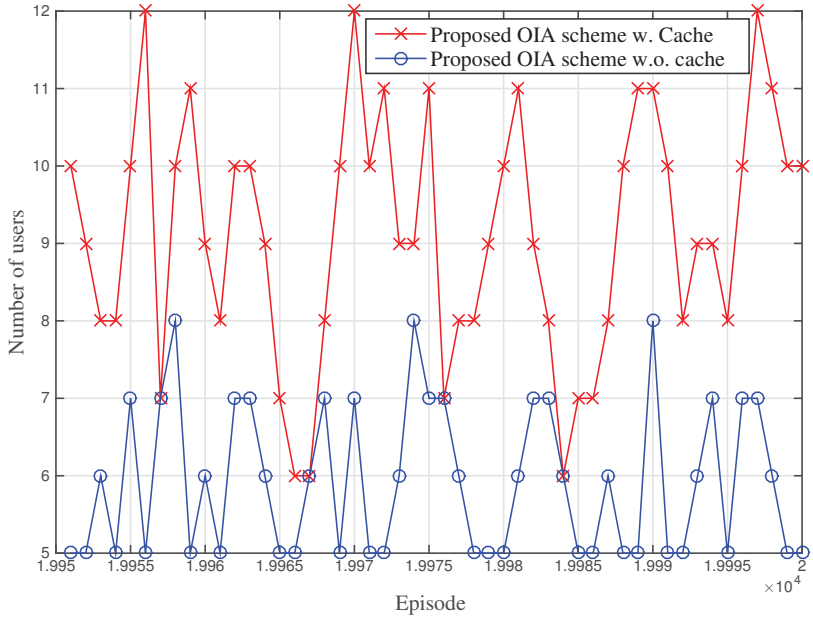

Figure 3.5: The number of users in the IA network after convergence with and without cache.

probabilities of staying in the same state. Our proposed schemes with (w.) and without (w.o.) cache are compared with an existing selection scheme w. cache [61] and an existing selection scheme w.o. cache [76], where invariant channels are assumed for both existing schemes. It can be seen that the proposed OIA w. cache scheme can achieve the highest sum rate compared to the other three schemes. This is because the channel is time-varying, and the proposed scheme can obtain the optimal IA user selection policy in the realistic time-varying channel environment using the deep reinforcement learning algorithm. For both the cases w. cache and w.o. cache, we can observe that the performance of the existing selection method is getting closer to the proposed OIA as the transition probability increases, and this method performs the same when the channel remains absolutely static, i.e., the transition probability

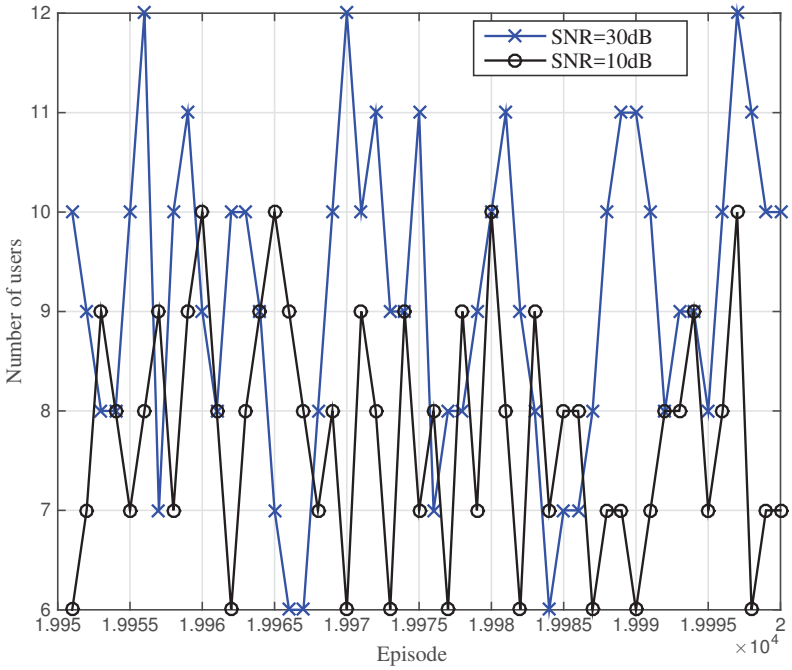

Figure 3.6: The number of users in the IA network after convergence with different SNR values.

that the channel will be in the same state is 1.

Fig. 3.8 shows the network's average sum rate with different average SNR values, where the SNR is defined as $10\log_{10}(P^{[k]}/\sigma^2)$ dB, and the same SNR definition applies to other figures in this chapter. From this figure, we can observe that the average sum rate of the proposed OIA scheme w. cache can outperform the other four schemes with different average SNR values. This is because the existing selection scheme assumes time-invaviant channels, and the estimated CSI of the current time instant is simply taken as the predicted CSI of the next time instant. In addition, the proposed OIA scheme w.o. cache does not take advantages of caching to relieve the traffic loads of backhaul links, thus has worse performance compared to the proposed scheme w. cache. Note that the existing user selection scheme w. cache performs similarly with the existing scheme w.o. user selection due to the fact that they both considers

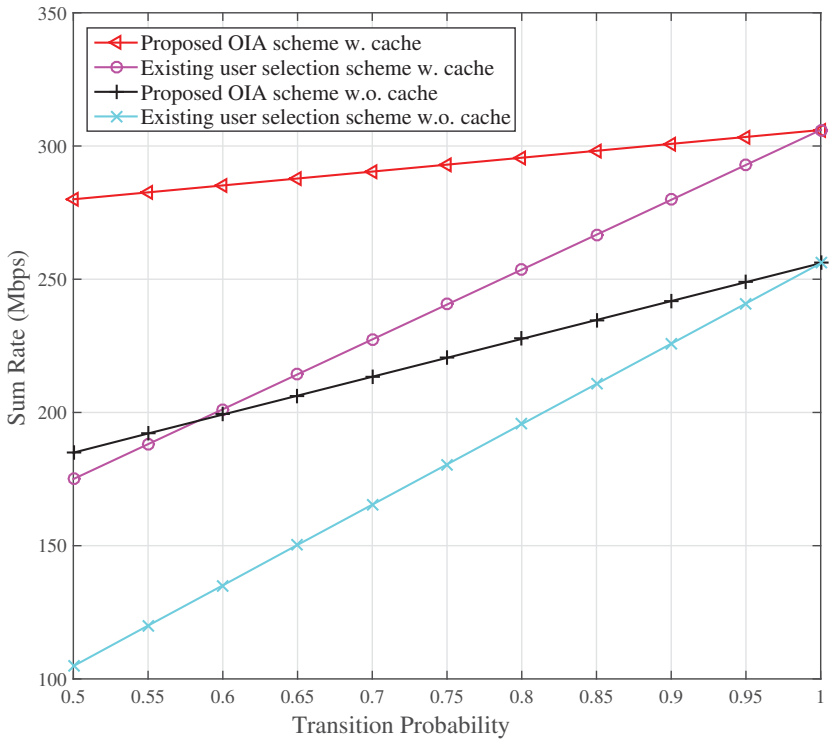

Figure 3.7: Comparison of the average sum rate with different transition probabilities.

caching and aims at maximizing the network sum rate, however, one exploits power allocation strategy, and the other one utilizes user selection strategy.

Fig. 3.9 and Fig. 3.10 present the average sum rate with different total backhaul capacity of the proposed DRL-based OIA scheme with and without cache, respectively. From these two figures, we can see that the network's sum rate increase with the total backhaul capacity. This is because, given large total backhaul capacity, more capacity can be used for CSI exchange, which can result in higher number of users, and consequently achieving larger sum rate. However, the effect of total backhaul capacity on the proposed DRL-based scheme with cache (DRL-wC) is not obvious, which is due to the fact that DRL-wC can utilize the advantages of cache to save

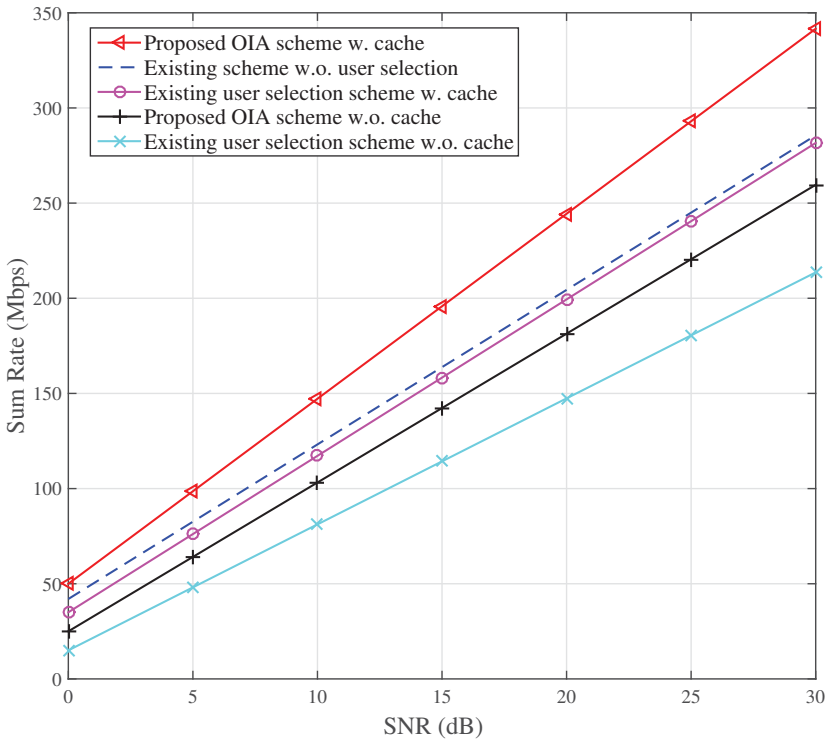

Figure 3.8: Comparison of the average sum rate with different SNR values.

more backhaul to be used for CSI share. Compared with Fig. 3.9, Fig. 3.10 obtains lower average sum rate due to the lower SNR.

Fig. 3.11 shows the cache benefits with total backhaul capacity under the conditions of SNR=10dB and SNR=30dB, respectively. From the figure, we can obviously observe that the cache benefit decreases with the increase of backhaul capacity when the SNR is larger. This is because, when the channel conditions are good, the proposed DRL-based scheme with cache does not rely on the backhaul link, in contrast, the proposed DRL-base scheme without cache is constrained by the limited backhaul capacity, therefore, the advantage of cache is more obvious with the increase of backhaul capacity.

Fig. 3.12 shows the effects of different ρ values. The average SNR is 30 dB. As

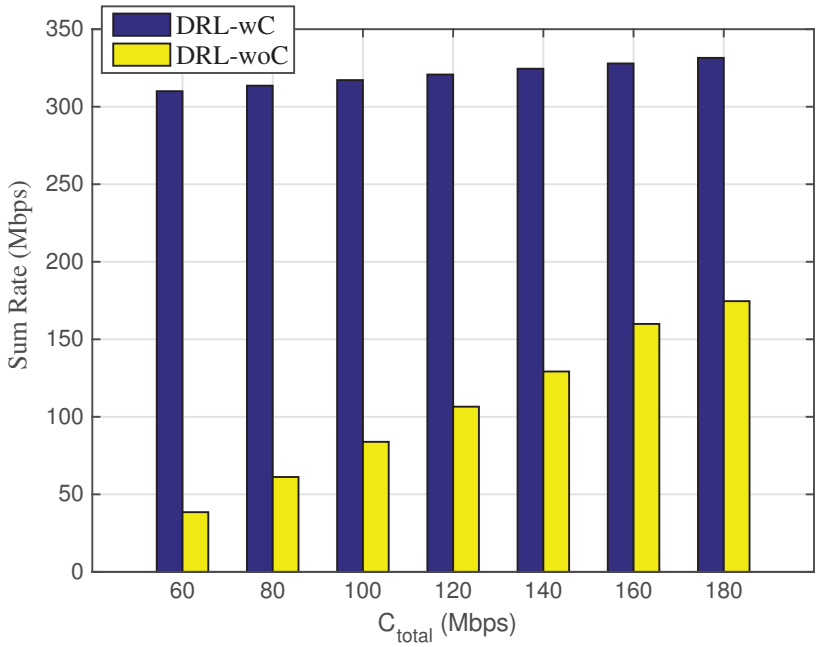

Figure 3.9: Comparison of the average sum rate with different total backhaul capacity (SNR = 30dB).

defined in (3.12), ρ is the normalized autocorrelation function of a fading channel with motion at a constant velocity. From Fig. 3.12, we can see that the network's average sum rate increases with the increase of ρ in different schemes. This is because a higher ρ value means more accurate CSI, which will result in higher average sum rate in the network. In comparison, the proposed OIA with cache scheme has better performance compared to two other schemes, because the proposed scheme considers the time-varying channels and takes advantage of caching. Fig. 3.13 shows the effects of different ρ values when the average SNR is 10 dB. We observe the same trend in Fig. 3.13 with lower average sum rate due to the low SNR value compared to Fig. 3.12.

Fig. 3.14 shows the energy efficiency of proposed scheme and other compared schemes. Except the existing w.o. user selection scheme [77] that exploits power

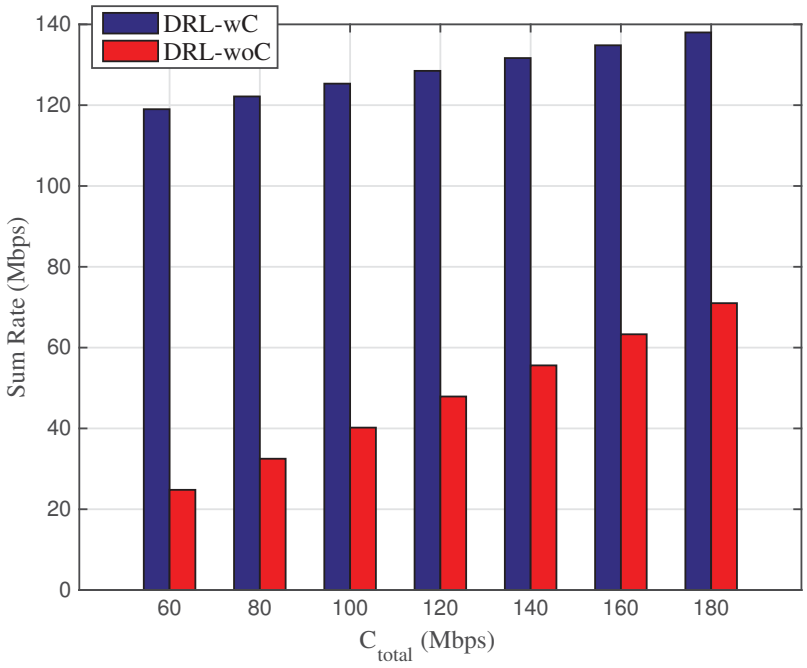

Figure 3.10: Comparison of the average sum rate with different total backhaul capacity (SNR = 10dB).

allocation, all the other four schemes are of equal power allocation. The power consumption of circuit blocks is set to be 210mW. The transmitted power of one user is 100mW, where for the power allocation scheme, the total power of the network is 500 mW for a 5-user IA network. The energy efficiency of the power amplifier is set to 90%. From this figure, we can observe that the existing scheme w.o. user selection that uses power allocation technique performs better than other existing schemes due to effective power allocation. Nevertheless, it still performs worse than the proposed scheme due to the underlying assumption of time-invaviant channels used in the existing scheme w.o. user selection that uses power allocation technique.

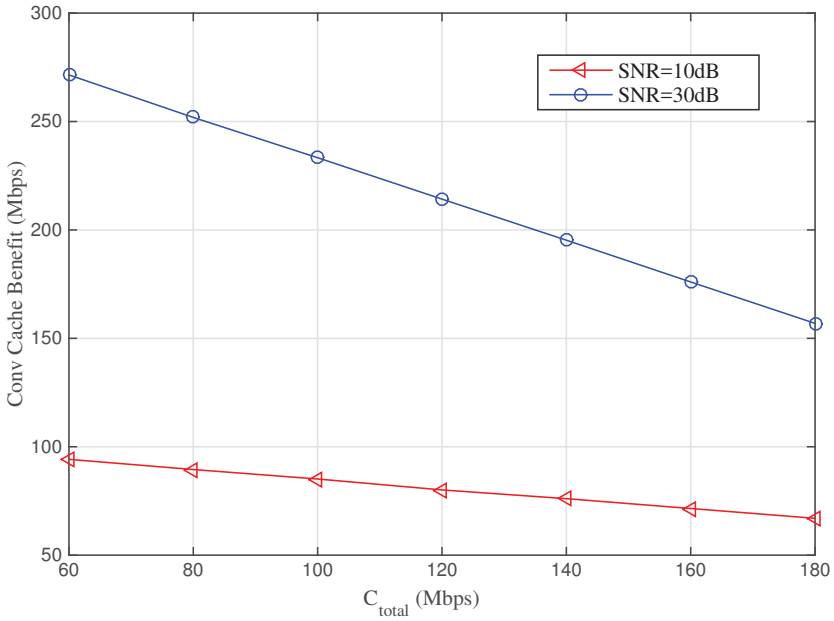

Figure 3.11: Comparison of the average sum rate with different SNR values.

3.5 Chapter Summary

In this chapter, we studied cache-enabled opportunistic IA under the condition of time-varying channel coefficients. The introduction of caching can save the limited backhaul capacity that can be utilized for CSI exchange among the transmitters in interference alignment wireless networks. The system complexity is very high when we model the time-varying channel as a finite-state Markov channel. Thus, we exploited the recent advances, and formulated the optimization of the cache-enabled opportunistic interference alignment network as a deep reinforcement learning problem. A central scheduler is responsible for collecting the CSI from each candidate, and then sends the integral system state to the deep Q network to derive the optimal policy for user selection. Simulation results were presented to show that deep reinforcement learning is an effective approach to solve the optimization problem in cache-enabled

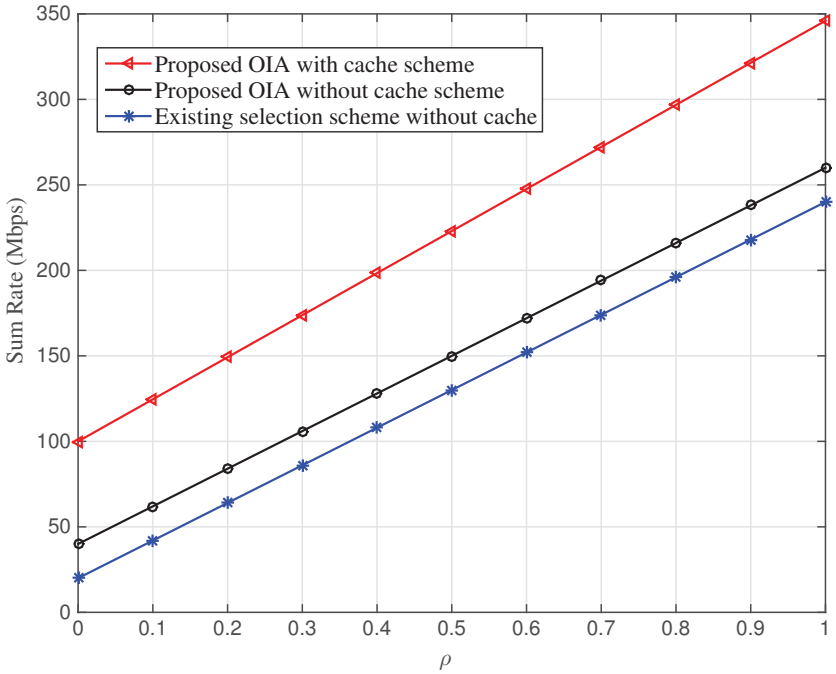

Figure 3.12: Comparison of the average sum rate with different ρ values (average SNR = 30dB).

opportunistic IA wireless networks. It was demonstrated that the performance of cache-enabled opportunistic IA networks can be significantly improved by using the proposed deep reinforcement learning approach. Nevertheless, some parameters, such as learning rate and mini-batch size, should be carefully chosen in the algorithm. Future work is in progress to consider wireless network virtualization in the proposed framework to further improve the network performance.

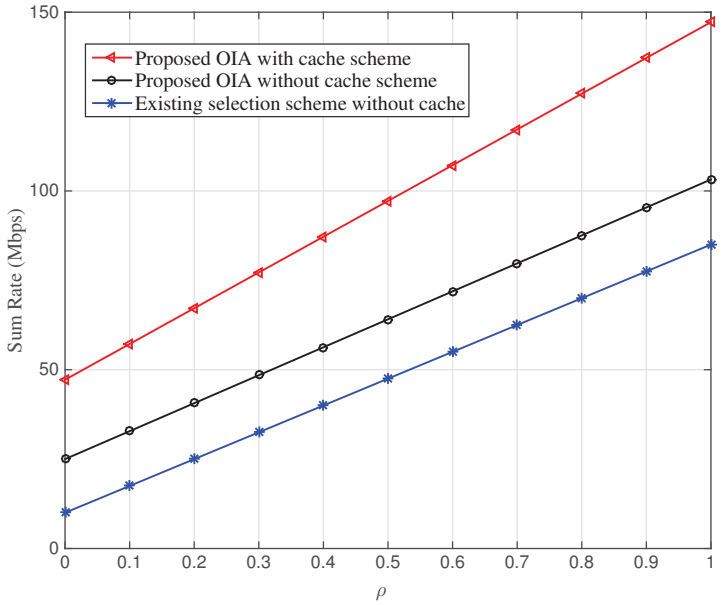

Figure 3.13: Comparison of the average sum rate with different ρ values (average SNR = 10dB).

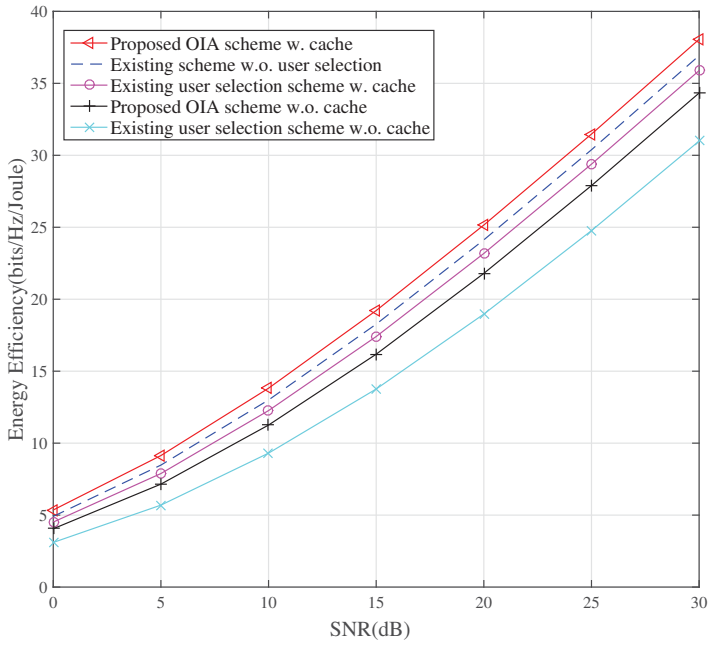

Figure 3.14: Energy efficiency comparision with different SNR values.

Chapter 4

Integrated Networking, Caching and Computing for Connected Vehicles: A Deep Reinforcement Learning Approach

4.1 Introduction

In recent years, there are great interests in connected vehicles, which use advanced technologies to connect vehicles with various infrastructures, devices, users, services, etc. It is envisioned that connected vehicles provide key enabling technologies to improve safety, enhance efficiency, reduce accidents, and decrease traffic congestion in transportation systems.

The developments of connected vehicles are heavily influenced by information and communications technologies, which have fueled a plethora of innovations in various areas, including *networking*, *caching* and *computing*. Recent advances in these areas have been extensively studied in the developments of connected vehicles. In the area of networking, *software-defined networking* (SDN) has attracted great interests in both academia and industry. The basic principle of SDN is to separate the control plane from the data plane, enabling the ability of programming the network via a

centralized software-defined controller with a global view of the network. Another related concept is *network function virtualization* (NFV), which enables abstraction of physical networking resources and flexible sharing of resources by multiple users through isolating each other [51, 80, 81]. It is beneficial to extend SDN and NFV to vehicular networks [82, 83]. *Software defined and virtualized vehicular networks* enable direct programmability of vehicular network controls and abstraction of the underlying infrastructure for a variety of applications of connected vehicles, with improved efficiency and great flexibility in vehicular network management [84].

Another promising technology, *in-network caching*, as one of the key features of *information-centric networking* (ICN), can efficiently reduce the duplicate content transmission in networks [13]. Particularly, caching content (e.g., videos) at network edge nodes (e.g., base stations (BSs) and road side units (RSUs)) has been proposed as one of the key enablers in next generation vehicular networks [11, 85]. The investigation on exploiting caching in vehicular networks has shown that access delays, traffic loads, and network costs can be significantly reduced by caching contents in vehicular networks [86].

Recent advances in computing (e.g., cloud/fog/edge computing) will have profound impacts on vehicular networks as well. *Cloud computing* has become very popular to enable access to a shared pool of computing resources [87, 88]. Nevertheless, as the distance between the cloud and the end device is usually large, cloud computing services may not provide guarantees to low latency applications in vehicular networks. To address these issues, *mobile edge computing* (MEC) [89, 90] have been studied to deploy computing resources closer to end vehicles, which can efficiently improve the quality of service (QoS) for applications that require intensive computations and low latency [91, 92].

While some excellent works have been done on networking, caching and computing

in vehicular networks, these three important enabling technologies have traditionally been studied separately in the existing works on vehicular networks. Jointly considering these new technologies for connected vehicles has been largely ignored in the existing research. In this chapter, we jointly consider networking, caching and computing to enhance the performance of vehicular networks. The distinct features of this chapter are as follows.

- Based on the programmable control principle originated from SDN and caching originated from ICN, we propose an integrated framework that can enable dynamic orchestration of networking, caching and computing resources to improve the performance of next generation vehicular networks.

- In this framework, we formulate the resource allocation strategy as a joint optimization problem, where the gains of not only networking but also caching and computing are taken into consideration in the proposed framework.

- The complexity of the system is very high when we jointly consider three technologies. Therefore, we propose a novel deep reinforcement learning approach in this chapter. Deep reinforcement learning is an advanced reinforcement learning algorithm that uses deep Q network to approximate the Q value-action function [8], and has been exploited in wireless networks to achieve automatic resource allocation [19, 20]. In this chapter, deep reinforcement learning is used to obtain the resource allocation policy in vehicular networks with integrated networking, caching, and computing.

- Simulation results with different system parameters are presented to show the effectiveness of the proposed scheme. It is illustrated that the performance of vehicular networks can be significantly improved with integrated networking, caching, and computing.

The rest of this chapter is organized as follows. In Section 4.2, the system description is presented. The system model is presented in Section 4.3. In Section 4.4, the cache-enabled opportunistic IA network is formulated as a deep reinforcement learning process. Simulation results are discussed in Section 4.5. Finally, conclusions are presented in Section 4.6.

4.2 System Description

In this section, we first introduce vehicular networks, followed by recent advances in software-defined and virtulaized vehicular networks. Then, information-centric networking and mobile edge computing are introduced to vehicular networks. Next, we present the proposed framework of integrated networking, caching and computing for connected vehicles.

4.2.1 Vehicular Networks

Vehicular networks deliver information and entertainment contents to drivers and vehicles by interconnecting various services, content, and application providers through mobile wireless networks. Typical applications over vehicular networks include advance safety warning, managing and playing audio content, utilizing navigation for driving, delivering entertainment such as movies, games, social networking, listening to incoming and sending outgoing SMS text messages, making phone calls, and accessing Internet-enabled or smartphone-enabled content, as well as more complex applications, such as cooperative driving (e.g., lane-merging assistance and platooning) and autonomous driving (e.g., driverless vehicles) [93].

Different vehicular applications have different preconditions. For example, some

of them are exclusively rely on Internet (e.g., web surfing, online gaming, video downloads, and email access), and others are supported by vehicle-to-vehicle communications (e.g., exchanging information and data between vehicles). Most safety applications are based on data dissemination in a neighboring geographical area, while most non-safety applications require a network-layer mobility solution to give vehicles Internet access. In general, vehicular communications are based on two main technologies: dedicated short range communications (DSRC) and cellular networks [94]. By using these two techniques, a vehicle can provide others with information acquired from its own sensors, other neighboring vehicles, road side units (RSUs) or direct access to the Internet.

4.2.2 Software-defined and Virtualized Vehicular Networks

It is beneficial to extend SDN principles (e.g., flexibility, programmability, and centralized control) to manage networking and communication resources in vehicular networks [84], e.g., to optimize channel allocation and network selection and reduce interference in multi-channel, multi-radio environment (e.g., wireless local area networks and cellular networks [95, 96]), to improve packet routing decisions in multi-hop environments, and to effectively handle mobility in high-speed scenarios.

In addition, wireless network virtualization has been considered as an effective and promising approach in the management of network architecture and network resources (e.g., spectrum and infrastructure) [51]. With different QoS, service functions or network requirements, a common physical wireless network can be virtualized into several virtual ones by the hypervisor. Moreover, since virtualization enables the sharing of network infrastructure and resources, the capital expenses (CapEx) as well as operation expenses (OpEx) of wireless access networks can be reduced significantly.

Fig. 4.1 shows a framework of integrated networking, caching and computing

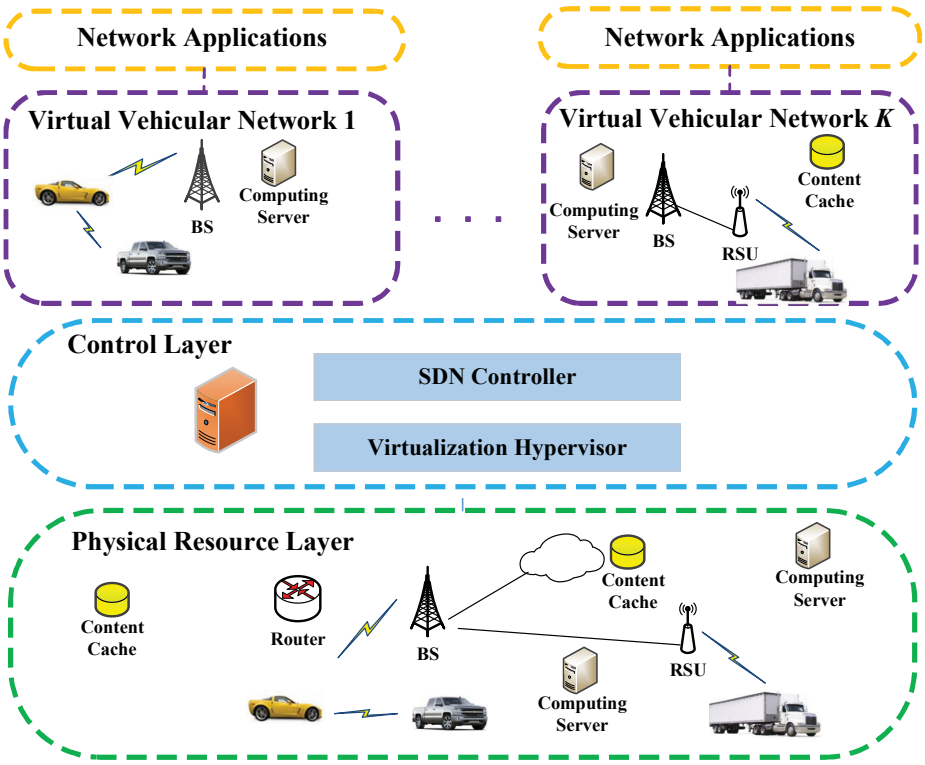

Figure 4.1: A framework of integrated networking, caching and computing for connected vehicles.

with software-defined and virtualized vehicular networks, where the infrastructure (i.e., the substrate network) is abstracted and virtualized into multiple customized virtual vehicular networks with different QoS requirements.

4.2.3 Information-centric Vehicular Networks

In the connected vehicle ecosystem, huge amounts of information-rich and safety-critical data will be exchanged by RSUs and onboard tranceivers in a variety of vehicular applications. Due to poor-quality wireless links and high vehicle mobility, it is challenging to deliver huge amounts of data using the traditional host-centric IP-based approach in vehicular networks [86]. Recent advances in ICN can be extended to

vehicular networks to address this issue. A typical representative example of the ICN architectures is named-data networking (NDN) [97], which uses content name to route and retrieve data. In NDN, the hourglass model of IP is retained, but the waist layer is changed with a hierarchical content naming structure. In addition, NDN integrates security into data itself by cryptographically signing every data packet to ensure integrity and authenticity. Moreover, in-network caching, as one of the key features of ICN, can efficiently reduce the duplicate content transmission in networks [98].

It is beneficial to extend this innovative ICN approach to vehicular networks, which natively privilege the information (e.g., trusted road traffic information in a specific proximity of a hazard and a specific time period) rather than the node identity. In addition, with in-network caching, ICN helps to address the mobility and sporadic connectivity issues in vehicular networks.

4.2.4 Vehicular Networks with Mobile Edge Computing

Recently, cloud computing has been applied in diverse domains, where user data needs to be transmitted to and processed in the data centers. However, since data centers are usually far away from the end users, cloud computing services may not provide guarantees to low latency applications for connected vehicles, and transmitting a large amount of data (e.g., in big data analytics) from the vehicle to the cloud may not be feasible or economical.

To address these issues, MEC has been proposed to deploy computing resources closer to end vehicles, which can efficiently improve the quality of service (QoS) for applications that require intensive computations and low latency [92].

4.2.5 Proposed Framework of Integrated Networking, Caching and Computing for Connected Vehicles

In most existing works on vehicular networks, networking, caching and computing are studied separately. However, from the vehicular application's point of view, network, cache and compute are underlying resources enabling vehicular applications. How to abstract, allocate and optimize these resources can have significant impacts on the performance of vehicular applications. In addition, although this traditional approach provides simplicity of the system design, it could result in suboptimal performance. Therefore, it is desirable to design an integrated framework for connected vehicles that enables network, cache and compute to be abstracted as a common resource pool for different vehicular applications. Moreover, this framework could enable the resources to be dynamically allocated based on the time-varying requirements of different vehicular applications.

Therefore, in this chapter, we propose a novel framework of integrated networking, caching and computing in a systematic for connected vehicles. Fig. 4.1 shows this framework. Based on the programmable control principle originated from SDN, we incorporate the ideas of information centricity originated from ICN. This integrated framework can enable dynamic orchestration of networking, caching and computing resources to meet the requirements of different applications.

One example use case in this framework is as follows. Assume that a vehicle issues a video content request to its associated virtual BS (could be a BS or RSU in the physical wireless network). First of all, according to the description of the video content and the information about the vehicle, the virtual BS will check whether or not its associated cache has the requested content. If yes, the cache will further examine if the version of its cached video content can be played and match with the vehicle. If still yes, the virtual BS will directly send the requested content to the vehicle from

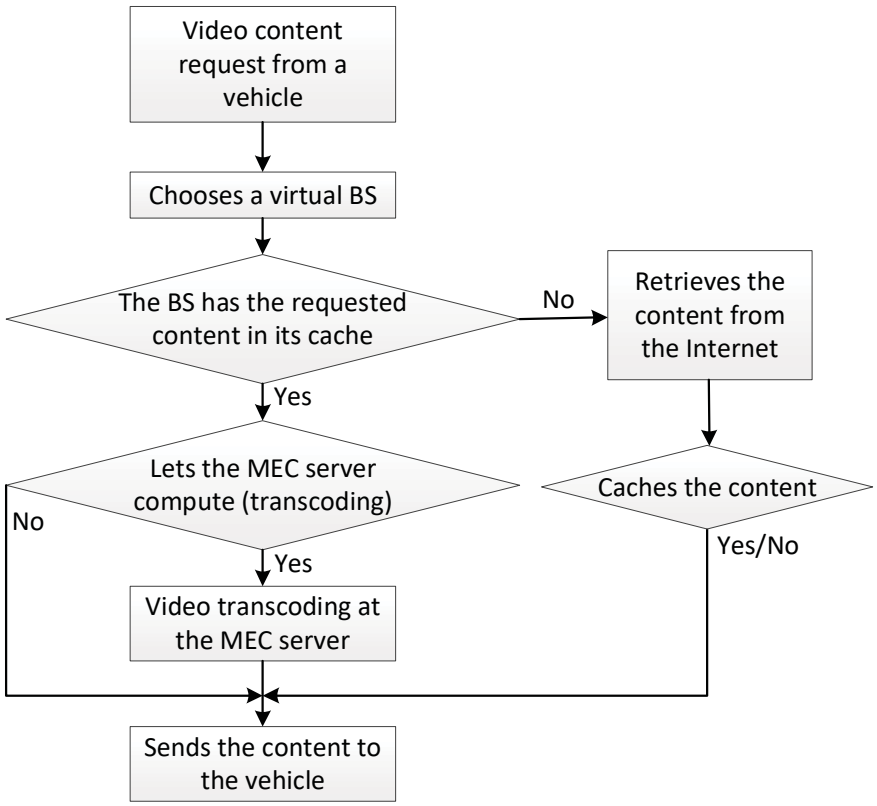

Figure 4.2: The procedure of video services in a vehicular network with MEC and caching.

the cache. If no, the virtual BS will extract the current video content and construct a computation task according to the size of the content involving the input data, codes and parameters, as well as the number of CPU cycles that is needed to accomplish the computing/transcoding task. Then the virtual BS will transmit the task to the MEC server to execute the computation. After the computation is finished, the virtual BS sends the transcoded video content to the vehicle. If the cache cannot find the matched video content, the virtual BS has to retrieve the content from Internet, and this will inevitably occupy some of the backhaul resources. The procedure is shown in Fig. 4.2.

When we consider networking, caching and computing jointly, the complexity of

the system is very high, which makes it difficult solve the resource allocation problem using traditional approaches [20, 68]. In this chapter, we used a novel deep reinforcement learning approach to solve the optimization problem in vehicular networks with integrated networking, caching and computing. We first present the system model considered in this chapter as follows.

4.3 System Model

In this section, we first present the network model, followed by the communication model. Then, the computation model and caching model are presented.

4.3.1 Network Model

We consider a software-defined and virtualized vehicular network with multiple vehicles requesting for video-concerned contents. The physical network infrastructures mainly include multiple base stations (BSs), road side units (RSUs), MEC servers, content caches, vehicles and routers, which are operated by the infrastructure providers (InPs). Let $\mathcal{K}_m$ and $\mathcal{K}_s$ be the sets of BSs and RSUs, respectively. $\mathcal{K} = \mathcal{K}_m \cup \mathcal{K}_s = \{1, ..., K\}$ and $\mathcal{U} = \{1, ..., U\}$ are the sets of all the BSs, RSUs, and vehicles, respectively. The sets of the MEC servers and content caches are denoted as $\mathcal{M} = \{1, ..., M\}$ and $\mathcal{C} = \{1, ..., C\}$, respectively.

Virtual networks are established logically. Each virtual network includes BSs, RSUs, MEC servers, and content caches, which can provide the capabilities of video content caching and high-speed computing for vehicles. The MEC server and the cache associated with BS k are given by m_k and c_k, $m_k \in \mathcal{M}$ and $\forall c_k \in \mathcal{C}$. It is assumed that each virtual network belongs to a different InP, and the licensed spectrum of different InPs is orthogonal so that there exits no interference among

virtual networks. The service providers (SPs) manage the virtual networks, and let $\mathcal{S} = \{1, ..., S\}$ be the set of SPs. For an SP s, the set of vehicles is denoted as $\mathcal{U}_s$, among which one allocated vehicle is represented by u_s. All the vehicles are managed by SPs, and one vehicle only subscribes to one SP at a given time period, i.e., $\mathcal{U} = \cup_s \mathcal{U}_s$ and $\mathcal{U}_s \cap \mathcal{U}_{s'} = \phi$, $\forall s' \neq s$. Let $a_{u_s,k}(t)$ denote the association between vehicle u_s and BS k at time instant t, where $a_{u_s,k}(t) = 1$ means that vehicle u_s connects to BS k to request for the video content; otherwise $a_{u_s,k}(t) = 0$. Practically, each vehicle can only access to one BS or RSU at one time, thus

$$\sum_{k \in \mathcal{K}} a_{u_s,k}(t) = 1, \forall s \in \mathcal{S}, u_s \in \mathcal{U}_s. \tag{4.1}$$

Since the network model is closely related to communication, computation and caching aspects, next the communication model, computation model, and caching model are described separately in detail.

4.3.2 Communication Model

Consider that the wireless channels between the vehicles and their connected BSs/RSUs are realistic time-varying channels, and they are modeled as FSMC. FSMC models have been widely accepted in the literature as an effective approach to characterize the correlation structure of the fading process [99, 100]. Considering FSMC models may enable substantial performance improvement over the schemes with memoryless channel models.

In FSMC models, the received SNR is a proper parameter that can be used to reflect the quality of a channel. The received SNR of the wireless channel linking vehicle u_s and BS k is modelled as a random variable $\gamma_{u_s}^k$. The value range of $\gamma_{u_s}^k$ can be partitioned and quantized into L discrete levels: $\mathcal{L}_0$, if $\gamma_0^* \leqslant \gamma_{u_s}^k < \gamma_1^*$; $\mathcal{L}_1$, if $\gamma_1^* \leqslant \gamma_{u_s}^k < \gamma_2^*$;

$\ldots; \mathcal{L}_{L-1}$, if $\gamma_{u_s}^k \geqslant \gamma_{L-1}^*$. Each level corresponds to a state of a Markov chain, thus forms a L-element state space $\mathcal{D} = \{\mathcal{D}_0, \mathcal{D}_1, \ldots, \mathcal{D}_{L-1}\}$. The channel state realization of $\gamma_{u_s}^k$ at time instant t can be denoted as$\Upsilon_{u_s}^{k}(t)$. We assume there are T time slots during the period of the whole communication, which starts when vehicle u_s issues a request and terminates when the requested content is obtained. Let $t \in \{0, 1, 2, \ldots, T-1\}$ denote the time instant. According to certain transition probabilities, the received SNR$\Upsilon_{u_s}^{k}(t)$ varies from one state to another when one time slot elapses. The transition probability that$\Upsilon_{u_s}^{k}(t)$ jumps from one state g_s to another state h_s can be denoted as $\psi_{g_s h_s}(t)$ at time slot t. The $L \times L$ wireless channel state transition probability matrix for the link between vehicle u_s and BS k is defined as:

$$\Psi_{u_s}^k(t) = \left[\psi_{g_s h_s}(t)\right]_{L \times L}, \tag{4.2}$$

where $\psi_{g_s h_s}(t) = \Pr\left(\Upsilon_{u_s}^k(t+1) = h_s \mid \Upsilon_{u_s}^k(t) = g_s\right)$, and $h_s, g_s \in \mathcal{D}$.

We assume that the whole available spectrum bandwidth is B Hz, and BS k is allocated with B_k Hz. The backhaul capacity of the vehicular network is Z bps, and BS k is assigned to Z_k bps. Let $b_{u_s,k}$ be the spectrum bandwidth orthogonally assigned from BS k to vehicle u_s, thus there is no interference to vehicle u_s from other vehicles that are also connected to BS k. The achievable spectrum efficiency of vehicle u_s associated with BS k can be denoted as $v_{u_s,k}(t)$, and based on Shannon bound it can be easily achieved. The communication rate of vehicle u_s can be expressed as

$$R_{u_s,k}^{\text{comm}}(t) = a_{u_s,k}(t) b_{u_s,k}(t) v_{u_s,k}(t), \forall s \in \mathcal{S}, u_s \in \mathcal{U}_s. \tag{4.3}$$

The sum rate of all the vehicles associated with BS k cannot exceed its backhaul

capacity, thus the following requirement must be met,

$$\sum_{s\in\mathcal{S}}\sum_{u_s\in\mathcal{U}_s} R^{\text{comm}}_{u_s,k}(t) \leq Z_k, \forall k \in \mathcal{K}. \tag{4.4}$$

Similarly, the data rate of all the vehicles in the whole virtual network cannot exceed the total backhaul capacity, thus the following condition should be met,

$$\sum_{k\in\mathcal{K}}\sum_{s\in\mathcal{S}}\sum_{u_s\in\mathcal{U}_s} R^{\text{comm}}_{u_s,k}(t) \leq Z. \tag{4.5}$$

4.3.3 Computation Model

In the computation model, we focus on the constructed computation task $Q_{u_s} = \{o_{u_s}, q_{u_s}\}, \forall s \in \mathcal{S}, u_s \in \mathcal{U}_s$, which is executed on vehicle u_s's associated MEC server m_k. The first parameter o_{u_s} stands for the size of the video content, and the second q_{u_s} denotes the required number of CPU cycles that is needed to complete the task. After the computation, BS k sends the transcoded video content back to vehicle u_s.

We denote the computation capability of BS k allocated to vehicle u_s as $f^k_{u_s}$, which can be measured by CPU cycles per second. In our virtualized vehicular network, different computation-speed MEC servers are assigned to BSs dynamically. In addition, multiple vehicles may access to the same BS and share the same MEC server at a given time. The above two points result in the fact that we don't know exactly the computation capability for vehicle u_s at the next time instant. Thus, the computation capability $f^k_{u_s}$ can be modelled as a random variable, and divided into discrete levels denoted by $\mathcal{E} = \{\mathcal{E}_0, \mathcal{E}_1, \ldots, \mathcal{E}_{N-1}\}$, where N is the number of available computation capability states. Assume the computation capability realization of $f^k_{u_s}$ to be $F^k_{u_s}(t)$ at time slot t. We model the transition of the computation capability level of a BS for one allocated vehicle in this virtualized vehicular network as a Markov chain. The

computation capability state transition probability matrix of BS k for vehicle u_s is defined as:

$$\Theta_{u_s}^{k}(t) = \left[\iota_{x_s y_s}(t)\right]_{N\times N}, \tag{4.6}$$

where $\iota_{x_s y_s}(t) = \Pr\left(F_{u_s}^{k}(t+1) = y_s \mid F_{u_s}^{k}(t) = x_s\right)$, and $x_s, y_s \in \mathcal{E}$.

The computation execution time of the task Q_{u_s} at BS k can be obtained as $T_{u_s,k} = \frac{q_{u_s}}{F_{u_s}^{k}(t)}$. We are more concerned with the metric of computation rate (bits computed per second), which can be given by

$$r_{u_s,k}^{comp}(t) = a_{u_s,k}(t)\frac{o_{u_s}}{T_{u_s,k}} = a_{u_s,k}(t)\frac{f_{u_s}^{k}(t)o_{u_s}}{q_{u_s}}. \tag{4.7}$$

Since the computation capacity of each MEC server is limited, the following condition should be satisfied

$$\sum_{s\in\mathcal{S}}\sum_{u_s\in\mathcal{U}_s} a_{u_s,k}(t)o_{u_s} \le O_k, \forall k \in \mathcal{K}, \tag{4.8}$$

where O_k is the maximum content size simultaneously executed on the MEC server m_k of BS k.

4.3.4 Caching Model

Assume that there are I contents in the server for the time interval we concern. For content i, $i \in 1, 2, \ldots, I$, the average request rate for content i at time t can be denoted as

$$\lambda_i(t) = \frac{\beta}{\rho i^{\alpha}}. \tag{4.9}$$

Here, the index i represents the ith most popular content and it is assumed that the requests for these I contents arrive according to a Poisson process with rate β [101]. The probability of requesting a content is determined by a Zipf-like distribution [102], thus the probability of content i being selected is $1/\rho i^{\alpha}$, where $\rho = \sum_{i=1}^{I} 1/i^{\alpha}$ and α is the Zipf slope such that $0 < \alpha < 1$.

In our system model, the content caches periodically store the contents from the server. Whether or not a vehicle's requested content i is in the cache can be viewed as a random variable ξ_i, and the state of the cache can be modeled using a two-state Markov chain [101]. State 0 corresponds to when content i is not in the cache. State 1 corresponds to when content i is in the cache. The state space can be expressed as $\mathcal{G} = \{0, 1\}$, where $\xi_i \in \mathcal{G}$. The cache state transition probability matrix of content i is defined as:

$$\Gamma_i(t) = [\delta_{\nu_s \omega_s}(t)]_{2\times 2}, \tag{4.10}$$

where $\delta_{\nu_s \omega_s}(t) = \Pr\left(\xi_i(t+1) = \omega_s \mid \xi_i(t) = \nu_s\right)$, and $\nu_s, \omega_s \in \mathcal{G}$.

Based on the cache capacity, two cases can be considered: infinite and finite cache capacity. In the infinite case, the cache can store all the requested contents, and the stored content is only eliminated from the cache when it exceeds the expiry time. The lifetime of content i follows exponential distribution with mean $1/\mu_i$. The cache state transition probability matrix can be derived from the Markov chain flow matrix as [101]

$$\Lambda_i^0 = \begin{bmatrix} -\lambda_i & \lambda_i \\ \mu_i & -\mu_i \end{bmatrix} \tag{4.11}$$

If finite cache capacity is assumed, the cache state transition probability matrix

can be derived according to different cache replacement strategies, among which the least recently used (LRU) cache replacement policy is commonly used. The transition probability matrix can be derived using the following Markov chain flow matrix [101],

$$\Lambda_i^1 = \begin{bmatrix} -\lambda_i & 0 & \cdots & 0 & 0 & \lambda_i \\ \zeta_i + \mu_i & -\beta - \mu_i & \cdots & 0 & 0 & \lambda_i \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mu_i & 0 & \vdots & \zeta_i & -\beta - \mu_i & \lambda_i \\ \mu_i & 0 & \vdots & 0 & \zeta_i & -\zeta_i - \mu_i \end{bmatrix} \tag{4.12}$$

where $\zeta_i = \beta - \lambda_i$.

4.4 Problem Formulation

In this section, we formulate the resource allocation optimization problem as a deep reinforcement learning process.

In our system model, there are K base stations, M MEC servers and C content caches that are virtualized and managed by a mobile virtual network operator (MVNO), and provide services for vehicles. Since we consider the realistic scenarios, where the base stations' downlink channel conditions, the MEC servers' computation abilities and the caches' states are all dynamically changing, the MVNO faces with large amount of system states, and it has to make an decision on which virtualized resources will be assigned to a specific vehicle according to the system's current state. It is barely possible to solve this complicated task by using a traditional method. Deep Q-learning is a recent advance that is capable of receiving complex high-dimensional

data as input, and yielding an optimal action for each input data. By taking the advantages of Deep Q-network, the MVNO can manage the system in an effective and efficient way.

Here, the MVNO is responsible for collecting the status from each base station, MEC server and content cache, and then it assembles the whole information into a system state. Then, the MVNO sends the constructed system state to the agent, (i.e., the deep Q-network) and gets a feedback of the optimal policy a^* for arranging which resources for a certain vehicle. After obtaining the action, the MVNO will send a notice to inform the vehicle to which virtualized network it can access.

Inside the deep Q-network, the replay memory stores the agent's experience of each time slot. The Q network parameters θ is updated at every time instant with samples from the replay memory. The target Q network parameters θ^- is copied from the Q network every N time instants. The ε-greedy policy is utilized to balance the exploration and exploitation, i.e. to balance the reward maximization based on the knowledge already known with trying new actions to obtain knowledge unknown. The training algorithm of the deep Q network is described in Algorithm *1*.

In order to obtain the optimal policy, it is necessary to identify the actions, states and reward functions in our deep Q learning model, which will be described in the next following subsections.

System State

The state of an available base station $k \in \{1, 2, \ldots, K\}$, an available MEC server $m \in \{1, 2, \ldots, M\}$, and an available cache $c \in \{1, 2, \ldots, C\}$ for vehicle u_s in time slot $t \in \{0, 1, \ldots, T-1\}$ is determined by the realization of the states$\Upsilon^{\ k}_{u_s}(t)$ of the random variables $\gamma^k_{u_s}$, the realization of the states $F^m_{u_s}(t)$ of the random variables $f^m_{u_s}$, and the realization of the states$\Xi^{\ c}_{u_s}(t)$ of the random variables ξ_c. Consequently, the

state vector can be described as follows.

$$\begin{aligned}\chi_{u_s}(t) = \big[&\Upsilon^1_{u_s}(t), \Upsilon^2_{u_s}(t), \ldots, \Upsilon^K_{u_s}(t),\\ &F^1_{u_s}(t), F^2_{u_s}(t), \ldots, F^M_{u_s}(t),\\ &\Xi^1_{u_s}(t), \Xi^2_{u_s}(t), \ldots, \Xi^C_{u_s}(t)\ \big].\end{aligned} \tag{4.13}$$

Here,$\Xi\ ^{c}_{u_s}(t) = [\xi_1, \xi_2, \ldots, \xi_I]$, for $\forall i \in \{1, 2, \ldots, I\}$, $\xi_i \in \{0, 1\}$.

System Action

In the system, the agent has to decide which BS is assigned to the vehicle, whether or not the requested content should be cached in the BS, and whether or not the computation task should be offloaded to the MEC server.

The current composite action $a_{u_s}(t)$ is denoted by

$$a_{u_s}(t) = \{a^{\text{comm}}_{u_s}(t), a^{\text{comp}}_{u_s}(t), a^{\text{cache}}_{u_s}(t)\}, \tag{4.14}$$

where $a^{\text{comm}}_{u_s}(t), a^{\text{comp}}_{u_s}(t), a^{\text{cache}}_{u_s}(t)$ are defined and interpreted as follows:

- Define row vector $a^{\text{comm}}_{u_s}(t) = [a^{\text{comm}}_{u_s,1}(t), a^{\text{comm}}_{u_s,2}(t), \ldots, a^{\text{comm}}_{u_s,K}(t)]$, where $a^{\text{comm}}_{u_s,k}(t)$ represents the communication control of the kth base station for vehicle u_s, and each element $a^{\text{comm}}_{u_s,k}(t) \in \{0, 1\}$, where $a^{\text{comm}}_{u_s,k}(t) = 0$ means the base station k is passive (not connected) at time slot t, and $a^{\text{comm}}_{u_s,k}(t) = 1$ means it is active (connected). Note that $\sum_{k\in\mathcal{K}} a^{\text{comm}}_{u_s,k}(t) = 1$.

- Define row vector $a^{\text{comp}}_{u_s}(t) = [a^{\text{comp}}_{u_s,1}(t), a^{\text{comp}}_{u_s,2}(t), \ldots, a^{\text{comp}}_{u_s,M}(t)]$, where $a^{\text{comp}}_{u_s,m}(t)$ represents the offloading control of the mth MEC server for vehicle u_s, and each element $a^{\text{comp}}_{u_s,m}(t) \in \{0, 1\}$, where $a^{\text{comp}}_{u_s,m}(t) = 0$ means the task is not offloaded to the mth MEC server at time slot t, and $a^{\text{comp}}_{u_s,m}(t) = 1$ means it is offloaded. Note that $\sum_{m\in\mathcal{M}} a^{\text{comp}}_{u_s,m}(t) = 1$.

- Define row vector $a_{u_s}^{\text{cache}}(t) = [a_{u_s,1}^{\text{cache}}(t), a_{u_s,2}^{\text{cache}}(t), \ldots, a_{u_s,C}^{\text{cache}}(t)]$, where $a_{u_s,c}^{\text{cache}}(t)$ represents the cache control of the cth content cache for vehicle u_s, and each element $a_{u_s,c}^{\text{cache}}(t) \in \{0, 1\}$, where $a_{u_s,c}^{\text{cache}}(t) = 0$ means the content is not cached at the cth content cache at time slot t, and $a_{u_s,c}^{\text{cache}}(t) = 1$ means it is cached. Note that $\sum_{c \in \mathcal{C}} a_{u_s,c}^{\text{cache}}(t) = 1$.

Reward Function

In this chapter, we set the comprehensive revenue of the MVNO as our system's reward. MVNO rents the wireless spectrum and the backhaul bandwidth from InPs, and assigns them to virtual SPs. MVNO needs to pay for the usage of spectrum to InPs, which is defined as δ_k per Hz for BS k. Furthermore, MVNO also needs to pay the computation fee as long as a computation task is executed on the MEC server. The unit price for the energy consumption at mth MEC server is defined as η_m per Joule. The fee for caching the content is denoted as ζ_c per unit space.

On the other hand, MVNO charge the vehicles for accessing to the virtual networks, which is defined as τ_{u_s} per bps. The vehicles, who have already paid the fee, can access to the virtual network for offloading their computation task. Besides, the fee for vehicle u_s to compute the task at BS k is defined as ϕ_{u_s} per bps. The backhaul cost, paid by MVNO is defined as κ_{u_s} per bps, can be saved when vehicles call for the contents that have already been cached at cache c.

The system reward is the MVNO's revenue, and it is formulated as the function of received SNR of the access wireless link, the computation capability and the cache state. The system action determines whether the reward will be gained. Therefore,

we define the reward for a specific vehicle u_s as:

$$
\begin{aligned}
&R_{u_s}(t) = \sum_{k=1}^{K} R_{u_s,k}^{\text{comm}}(t) + \sum_{m=1}^{M} R_{u_s,m}^{\text{comp}}(t) + \sum_{c=1}^{C} R_{u_s,c}^{\text{cache}}(t) \\
=&\sum_{k=1}^{K} a_{u_s,k}^{\text{comm}}(t)(\tau_{u_s} r_{u_s,k}^{\text{comm}}(1 - w^{\text{comp}} a_{u_s,m}^{\text{comp}}(t)) - \delta_k b_{u_s,k}) \\
&+ \sum_{m=1}^{M} a_{u_s,m}^{\text{comp}}(t)(\phi_{u_s} r_{u_s,m}^{\text{comp}} - \eta_m q_{u_s} e_m) \\
&+ \sum_{c=1}^{C} a_{u_s,c}^{\text{cache}}(t)(\kappa_{u_s} r_{u_s,k}^{\text{cache}} - \zeta_c o_{u_s}) \\
=&\sum_{k=1}^{K} a_{u_s,k}^{\text{comm}}(t)(\tau_{u_s} b_{u_s,k} v_{u_s,k}(1 - w^{\text{comp}} a_{u_s,m}^{\text{comp}}(t)) - \delta_k b_{u_s,k}) \\
&+ \sum_{m=1}^{M} a_{u_s,m}^{\text{comp}}(t)(\phi_{u_s} \frac{F_{u_s}^{m}(t) o_{u_s}}{q_{u_s}} - \eta_m q_{u_s} e_m) \\
&+ \sum_{c=1}^{C} a_{u_s,c}^{\text{cache}}(t)(\kappa_{u_s} b_{u_s,k} v_{u_s,k} \Xi_{u_s}^{c}(t) - \zeta_c o_{u_s}).
\end{aligned}
\tag{4.15}
$$

The immediate system reward is the MVNO's revenue from a certain user u_s at time instant t, i.e., $R_{u_s}(t)$ in equation (25). The agent gets $R_{u_s}(t)$ in state $\chi_{u_s}(t)$ when action $a_{u_s}(t)$ is performed in time slot t. The goal of using deep Q network is to find an optimal policy to maximize the long-term revenue of MVNO, and the cumulative revenue can be written as

$$
R_{u_s}^{long} = \max E\left[\sum_{t=0}^{t=T-1} \epsilon^t R_{u_s}(t)\right], \tag{4.16}
$$

where ϵ^t approaches to zero when t is large enough. In practice, a threshold for terminating the process can be set.

4.5 Simulation Results and Discussions

In this section, we use computer simulations to show the performance of the proposed deep reinforcement learning approach to vehicular networks with integrared networking, caching and computing. We use TensorFlow [74] in our simulations to implement deep reinforcement learning. In this section, we first present simulation settings. Then, simulation results are discussed.

4.5.1 Simulation Settings

In our simulations, we used a GPU-based server, which has 4 Nvidia GPUs with version GTX TITAN. The CPU is Intel Xeno E5-2683 v3 with 128G memory. Software environment we utilized is TensorFlow 0.12.1 with Python 2.7 on Ubuntu 14.04 LTS.

For performance comparison, 4 schemes are compared with the proposed scheme:

1. Existing static scheme with (w.) communication, computing and caching: The system state is assumed to be static, which does not change dynamically. Communication, computing and caching are jointly optimized under this static assumption.

2. Proposed scheme without (w.o.) virtualization: Virtulization is not considered in this scheme, and vehicles can only connect to one SP. MEC offloading and caching are considered in this scheme.

3. Proposed scheme without (w.o.) MEC offloading: MEC offloading is not considered in this scheme, and vehicles can only perform the computation tasks locally. Virtualization and caching are considered in this scheme.

4. Proposed scheme without (w.o.) edge caching: Edge caching is not considered in this scheme, and vehicles can only get the video contents from the remote

server. Virtualization and MEC offloading are considered in this scheme.

All BSs and vehicles are randomly distributed within the covered area of the MBS. In the simulations, we assume that there are five SPs, five BSs, five MEC server, one MVNO, and the bandwidth of each BS is normalized. The wireless channels between the vehicles and BSs follow the Markov model. We assume that the state of the channel can be bad (the spectrum efficiency $v_{u_s,k} = 1$) or good (the spectrum efficiency $v_{u_s,k} = 3$). We set the transition probability of staying in the same state as 0.7, and set the transition probability from one state to another as 0.3. There is one video content, and the cache state follows the Markov model, We set the cache state transition probability of staying in the same state as 0.6, and set the transition probability from one state to another as 0.4. The computation states of MEC servers follow the Markov model. We assume that the computation state of the MEC server can be very low, low, medium, high, and very high. We set the transition probability matrix as follows.

$$\Theta = \begin{bmatrix} 0.5 & 0.25 & 0.125 & 0.0625 & 0.0625 \\ 0.0625 & 0.5 & 0.25 & 0.125 & 0.0625 \\ 0.0625 & 0.0625 & 0.5 & 0.25 & 0.125 \\ 0.125 & 0.0625 & 0.0625 & 0.5 & 0.25 \\ 0.25 & 0.125 & 0.0625 & 0.0625 & 0.5 \end{bmatrix}, \tag{4.17}$$

The values of the rest of parameters are summarized in Table 2.

Table 2: Simulation Parameter Values Used in Chapter 4

Parameter	Value	Description
$b_{u_s,k}$	5MHz	The bandwidth of BS k allocated to vehicle u_s.
τ_{u_s}	10 units/Mbps	The unit charging-price for accessing to the virtualized networks.
δ_k	2 units/MHz	The unit paid-price for the usage of wireless spectrum.
ϕ_{u_s}	1 units/Mbps	The unit charging-price for executing MEC offloading.
q_{u_s}	100 Mcycles	The required number of CPU cycles to complete each task.
o_{u_s}	1 Mbits	The content size for one task.
$v_{u_s,k}$	[1, 3] bps/Hz	The spectrum efficiency.
κ_{u_s}	3 units/Mbps	The charging-price for connecting to the cache servers.
$F^k_{u_s}$	[4, 6, 8, 10,12] GHz	The realization of the computation capability $f^k_{u_s}$.
η_m	100 units/J	The paid-price for unit energy consumption of the MEC servers.
e_m	1W/GHz	The energy consumption for performing one CPU cycle.
w^{comp}	0.5	The effect factor of executing MEC offloading on the virtualization.
ζ_c	3 units/Mbits	The unit paid-price for connecting to the cache servers.

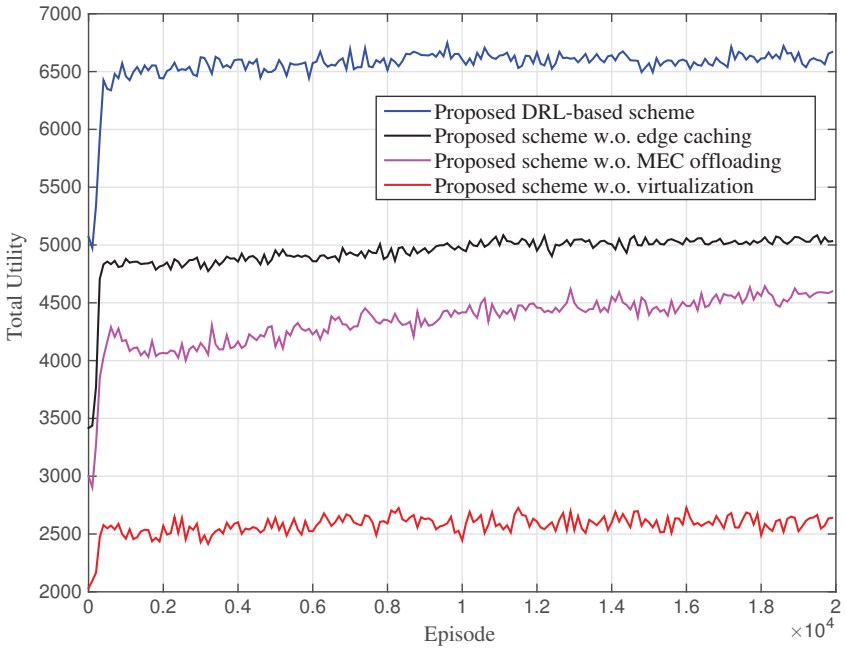

Figure 4.3: Convergence performance of different schemes.

4.5.2 Simulation Results

Fig. 4.3 shows the convergence performance of different scenarios in the proposed scheme using the deep reinforcement learning algorithm. From Fig. 4.3, we can observe that the total utility of different scenarios in the proposed scheme is very low at the beginning of the learning process. With the increase of the number of the episodes, the total utility increases until it reaches a relatively stable value, which is around 6600 in the proposed scheme with integrated networking, caching and computing in Fig. 4.3. This shows the convergence performance of the proposed scheme. We can also observe that the total utility of other scenarios is less when networking, caching and computing are not jointly considered. Particularly, the total utility is the lowest when virtualization is not considered compared to other scenarios.

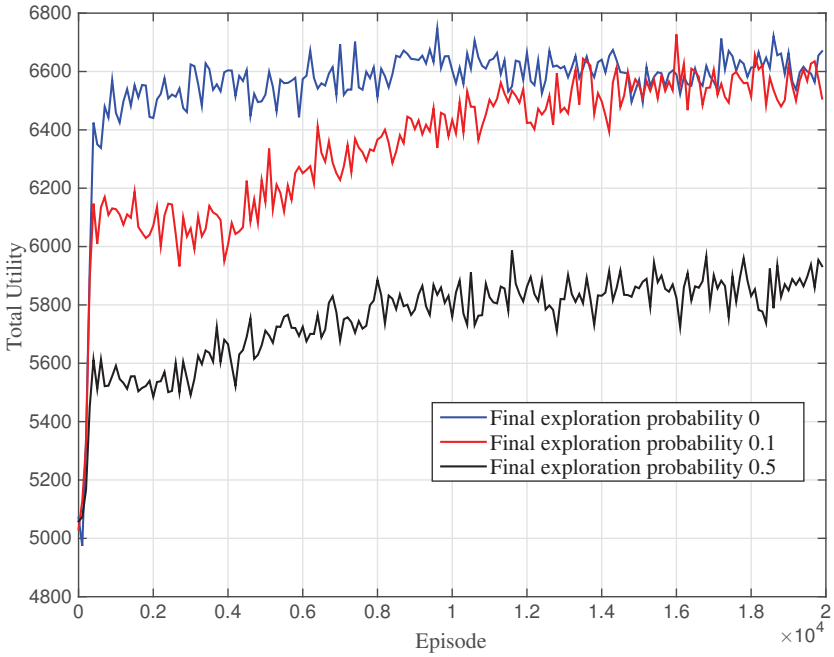

Figure 4.4: Convergence performance with different final exploration probabilities.

Fig. 4.4 shows the convergence performance of the proposed scheme with different final exploration probabilities in the deep reinforcement learning algorithm. Exploration probability is the chance of performing a random action, otherwise it will perform the action produced by the deep Q-network. In order to balance the exploration and exploitation, the initial exploration probability is set large, and the final exploration probability is set much smaller, even zero. From Fig. 4.4, we can observe the that the final exploration probability has some effects on the total utility of the proposed scheme. Specifically, a higher final exploration probability will result in a lower total utility after the algorithm converges. Nevertheless, a lower final exploration probability has the risk of finding local optimum instead of global optimum. In the rest of the simulations, we choose the final exploration probability of 0.

Fig. 4.5 shows the convergence performance with different random disturbance probabilities. In reality, the wireless channel environment is complex, normally there exists random disturbance (noise) when collecting data. In order to verify that the proposed scheme is capable of resisting certain level of random disturbance, we intentionally produce 2% and 20% probability of disturbance. That is to say, in one episode with 50 steps, one and ten states are randomly chosen, and transferred to other states, not based on the given transition probabilities. From the simulation results, we can see that the proposed scheme can still converge quickly with 2% disturbance, however, the total utility is decreased compared to the no disturbance case. This is because, although the proposed scheme can eventually find out the optimal decision, however, it cannot avoid making the wrong the decisions on disturbed states, therefore the total utility is decreased. If there is 20% random disturbance, the proposed scheme is not able to learn from the environment, so the decision is made nearly randomly, and inevitably the total utility is low.

Fig. 4.6 shows the effects of the mini-batch size for each gradient update in the deep reinforcement learning algorithm on the convergence performance. The parameter of mini-batch size determines how many experience cases are used for each training step. We can see from Fig. 4.6 that the convergence is faster when the mini-batch size is 16 compared to the cases when the mini-batch size is 32 and 48. Similar to the learning rate parameter, an appropriate mini-batch size should be chosen for a specific problem. In the rest of the simulations, we choose the mini-batch size of 16.

Fig. 4.7 shows the effects of q_{u_s}, which is the required number of CPU cycles for each task. We can see from Fig. 4.7 that q_{u_s} has no effects on the total utility of the proposed scheme without MEC offloading. This is because the total utility will not be changed with different required number of CPU cycles if MEC offloading is not available in the proposed scheme. In other scenarios of the proposed scheme and

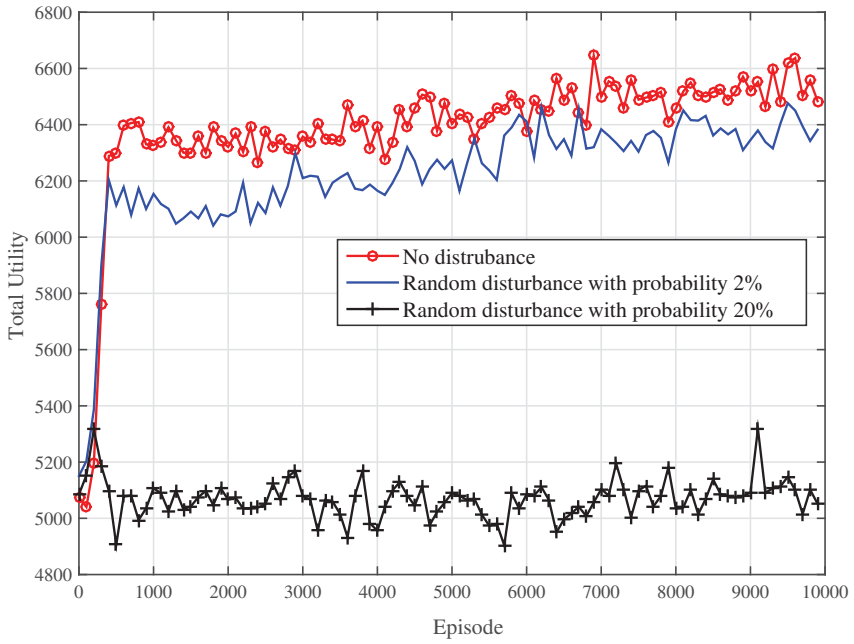

Figure 4.5: Convergence performance with different random disturbance probabilities.

the static scheme, the total utility decreases exponentially with the increase of the required number of CPU cycles for each task. This is because a larger number of required number of CPU cycles will induce a higher consumed computation energy and a lower gain of computation utility, and thus the total utility decreases.

Fig. 4.8 shows the effects of o_{u_s}, which is the content size. We can see from Fig. 4.8 that the total utility of the proposed scheme without MEC offloading deceases with the increase of content size. This is because a larger content size will increase the fee for caching the content, which induces a lower gain of caching utility, and thus the total utility decreases. By contrast, the total utility of the schemes with MEC offloading increases with the increase of content size due to the fact that a larger content size will increase the charging fee for computation at the MEC server, which

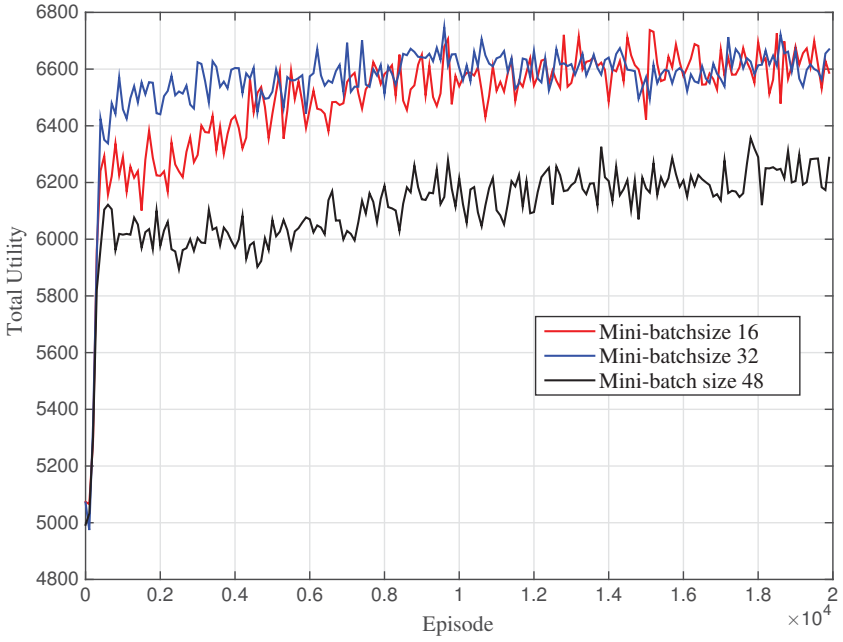

Figure 4.6: Convergence performance with different batch sizes.

induces a higher gain of computation utility, and thus the total utility increases.

Fig. 4.9 shows the effects of τ_{u_s}, which is the unit charging price for accessing to the virtualized networks. From the figure, we can see that the gap between the proposed scheme with and without MEC offloading narrows with the increase of τ_{u_s}. This is due to the fact that MEC offloading will reduce the charging fees from accessing to the virtualized networks. With the increase of τ_{u_s}, the virtualization occupies increasingly higher proportion in the overall incomings, thus MEC offloading will be less activated and phases out in the proposed scheme. By contrast, the scheme without MEC offloading will increase much faster than the scheme with MEC offloading, until they equal.

Fig. 4.10 shows the effects of ϕ_{u_s}, which is the unit charging price for executing MEC offloading. From the figure, it can be observed that the difference between

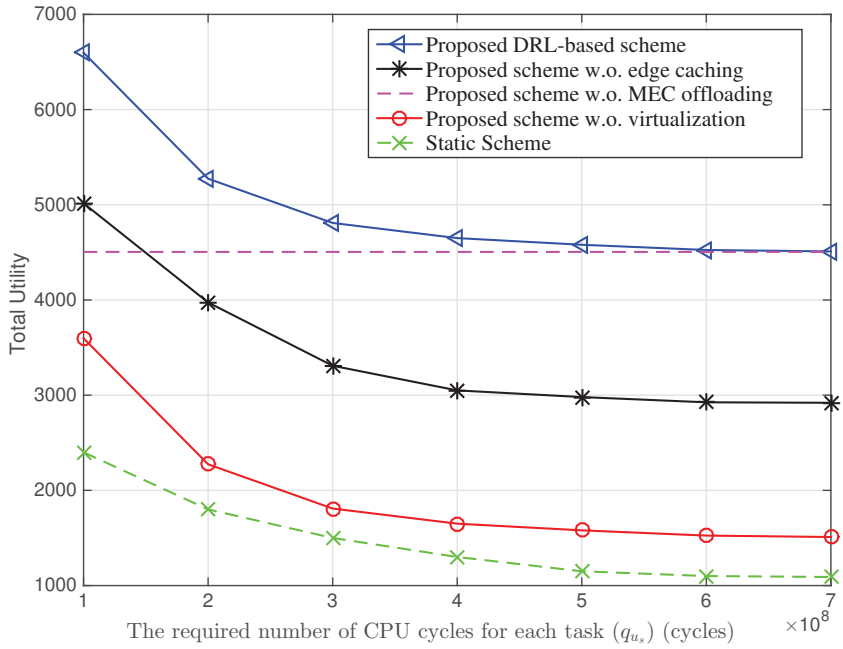

Figure 4.7: Effects of q_{u_s}, which is the required number of CPU cycles for each task.

the proposed scheme with and without virtualization increases with ϕ_{u_s}. This is because that in the proposed scheme with the rise of ϕ_{u_s}, the incomings of executing MEC offloading would increase, thus MEC offloading is of highly probability to be performed , which results in lower earnings from virtualization. This explains the increasing difference between the proposed scheme with and without virtualization.

Fig. 4.11 shows the effects of κ_{u_s}, which is the unit charging price for connecting to cache servers. From the figure, we can see that the total utility of the two schemes without virtualization and without MEC offloading grow parallelly with the proposed scheme, which indicates that the increase of κ_{u_s} has not much effect on these two schemes.

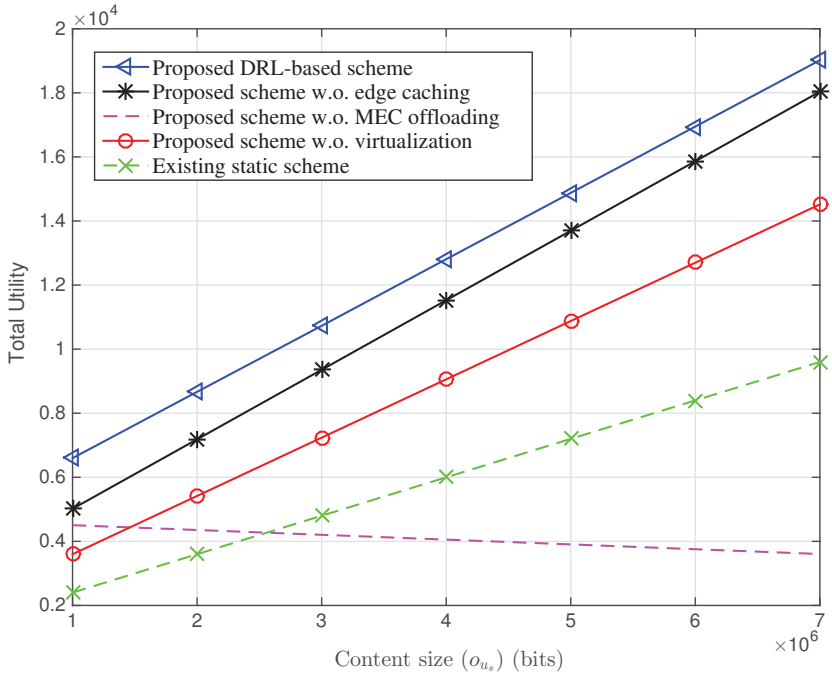

Figure 4.8: Effects of o_{u_s}, which is the content size.

4.6 Chapter Summary

In this chapter, we jointly studied networking, caching and computing to enhance the performance of vehicular networks. Based on the programmable control principle originated from SDN and caching originated from ICN, we proposed an integrated framework that can enable dynamic orchestration of networking, caching and computing resources to improve the performance of next generation vehicular networks. In this framework, we formulated the resource allocation strategy as a joint optimization problem, where the gains of not only networking but also caching and computing are taken into consideration in the proposed framework. Then, we proposed a novel deep reinforcement learning approach in this chapter. We presented the convergence

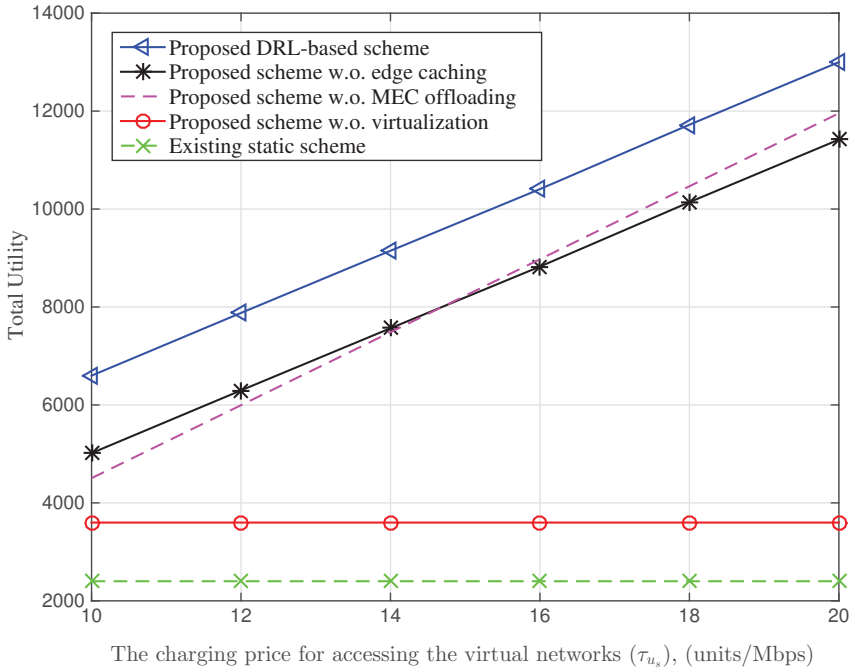

Figure 4.9: Effects of τ_{u_s}, which is the unit charging price for accessing to the virtualized networks.

performance of different scenarios in the proposed scheme using the deep reinforcement learning algorithm. Simulation results with different system parameters were presented to show the effectiveness of the proposed scheme. Future work is in progress to consider energy efficiency issues in the proposed framework.

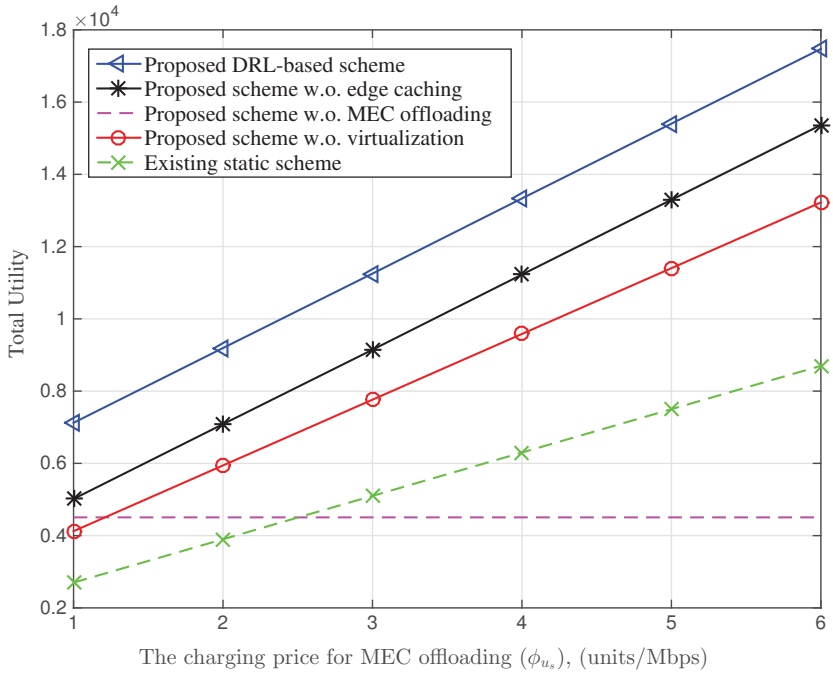

Figure 4.10: Effects of ϕ_{u_s}, which is the unit charging price for executing MEC offloading.

Algorithm *1* Double-Dueling-Deep Q-network algorithm in the virtualized vehicular network.

Initialization.

Initialize the experience replay buffer.
Initialize the main deep-Q network with weights θ.
Initialize the target deep-Q network with weights $\theta^{-} = \theta$.

For episode $k = 1, ..., K$ **do**:

Receive the initial observation s_1, and pre-process s_1 to be the beginning state x_1.

For $t = 1, 2, 3..., T$ **do**:

Choose a random probability p.
Choose a_t as,
if $p \leq \varepsilon$
randomly select an action a_t,
otherwise,
$a_t = \arg\max_a Q(x, a; \theta)$.

Execute action a_t in the system, and obtain the reward r_t, and the next observation s_{t+1}.
Process s_{t+1} to be the next state x_{t+1}.
Store the experience (x_t, a_t, r_t, x_{t+1}) into the experience replay buffer.
Get a batch of U samples (x_i, a_i, r_i, x_{i+1}) from the replay memory.
Calculate the target Q-value y_t^{-} from the target deep-Q network,

$$y_i^{-} = r_i + \varepsilon Q(x_{i+1}, arg\max_{a'} Q(x_{i+1}, a^{'}; \theta); \theta^{-})$$

Update the main deep Q-network by minimizing the loss $L(\theta)$,

$$L(\theta) = \frac{1}{U} \sum_i (y_i^{-} - Q(x_i, a_i; \theta))^2,$$

and perform a gradient descent step on $L(\theta)$ with respect to θ.
Every G steps, update the target deep Q-network parameters with rate σ,

$$\theta^{-} = \sigma\theta + (1 - \sigma)\theta^{-}.$$

End for
End for

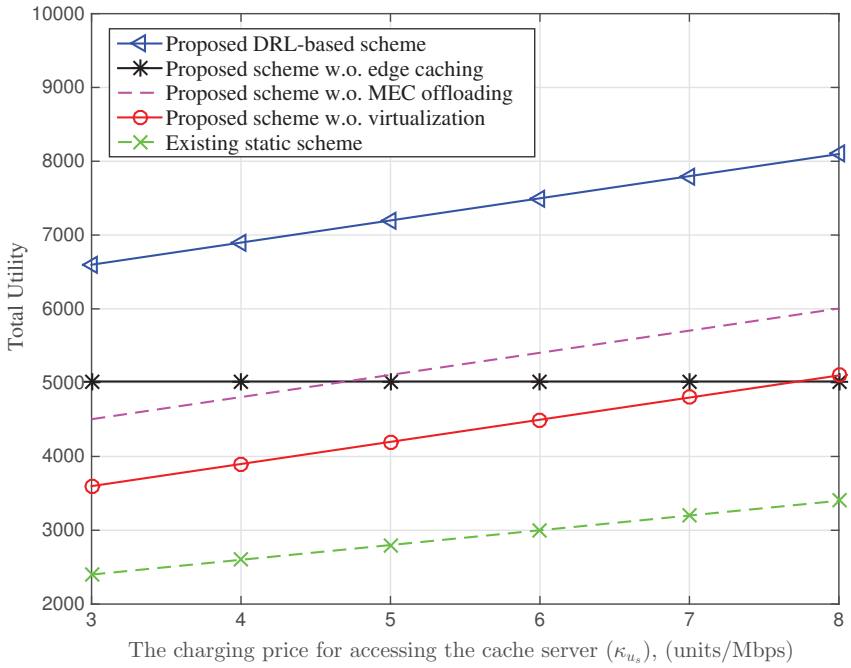

Figure 4.11: Effects of κ_{u_s}, which is the unit charging price for connecting to cache servers.

Chapter 5

Deep Reinforcement Learning for Trust-based Mobile Social Networks with Caching and Edge Computing

5.1 Introduction

Mobile social networks (MSNs) have been developing rapidly, which can provide a variety of social services and applications to mobile users [21]. Millions of mobile users interact with each other in MSNs, and MSNs will become one of the most important networking paradigms in future wireless mobile networks [22]. MSNs have always been trying to be innovative and leverage new technologies. The recently proposed integrated framework of computing, caching and communication (3C) can have positive impacts on MSNs. The integration of the 3C framework and MSNs can help create a new mechanism to share resources among users. The available resources that can be shared extend beyond information, for instance, the storage devices and computing capability contributed by users can also be shared. Additionally, the MSNs have an inherent advantage, i.e., the trust of mobile users is naturally formed through the social relationships and interactions. This makes the resource sharing

between users more reliable. In this chapter, we consider the specific 3C framework with *mobile edge computing* (MEC), *in-network caching*, and *device-to-device* (D2D) communications.

With MEC, computation resources are placed at the edge of wireless mobile networks in physical proximity to mobile users [103]. Compared to traditional mobile cloud computing, MEC can provide faster interactive response by low-latency connections. Therefore, MEC has been envisioned as a promising technique to offer agile and ubiquitous computation augmenting mobile services and applications, including social services and applications [92, 104]. Another promising technology is *in-network caching*, which can effectively reduce the duplicate content transmission in the network. Recent studies of applying the in-network caching technique in MSNs show that traffic loads and latency can be significantly reduced by caching contents in MSNs [22].

In addition, D2D communications can be beneficial to MSNs as well [105]. With D2D communications, users in close proximity can directly communicate with each other via D2D links, instead of accessing base stations (BSs) exclusively. When it comes to the content-centric MSNs, in spite of the smaller-sized storage (compared to that of BSs), the ubiquitous caching capability residing in mobile devices cannot be neglected due to their ubiquitous in-network distribution and ever-increasing storage size [105, 106].

Although some works have been done on applying recent advances of MEC, in-network caching and D2D to improve the performance of MSNs, the *knowledge of social relationships* among users in MSNs is largely ignored in these new paradigms to improve the reliability and efficiency of resource sharing and delivery in MSNs. In fact, social relationships have been investigated to enhance wireless networking. For

example, Zhang *et al.* [107] exploit social ties to achieve a reliable D2D communication, which can improve packet transmission and reduce the workload on the network infrastructures. Pan *et al.* [108] utilize social information as an essential element for designing forwarding algorithms in ad hoc networks. Social relationships also play an important role for routing in wireless environments [109].

Furthermore, the dynamic nature of the available resources has not been paid much attention in the existing literature. To fill in this gap, in this chapter, we study trust-based social networks with recent advances of MEC, caching and D2D. Considering the integrated network, the allocation of resources for subscribed users is complicated, especially when the conditions of the network resources are varying with time. Therefore, we utilized a novel deep reinforcement learning approach to automatically achieve the resource allocation tasks.

In the following, we first review some of the existing excellent works that focus on efficient management and allocation schemes of computing, caching and communication resources. Some of the proposed schemes only consider one of these three types of resources. On the other hand, in some other works, two or three types of resources are jointly considered to effectively and efficiently achieve optimal performance metrics. Then, we discuss our contributions in this chapter.

5.1.1 Related Works

With the increasing popularity of compute intensive applications, such as augmented reality, interative video games, etc., MEC is becoming a very promising paradigm that enables the possibility of offloading the computation tasks from resource-limited mobile devices to more powerful edge servers. MEC can bring various benefits, where minimizing energy consumption and minimizing latency are two major optimization objectives [110]. In the literature, various resource allocation approaches have been

proposed to solve the formulated optimization problems. For example, Zhang *et al.* [111] propose an energy-efficient computation offloading scheme that jointly optimizes radio resource allocation and offloading to minimize the energy consumption of the offloading system with the latency constraint. In their scheme, the mobile devices are first classified into three types and then wireless channels are allocated to mobile devices based on their priority iteratively. By doing this, the optimization problem can be solved in polynomial complexity. Liu *et al.* [112] design an optimal computation task scheduling policy for MEC systems. They first analyze the average delay of each task and average power consumption at the mobile device side under the proposed scheduling policy using Markov chain theory. Then they formulate a delay minimization problem with the power constraint. An efficient one-dimensional search algorithm is used to derive the optimal offloading policy. In [113], the authors jointly optimize the radio resources and computational resources to minimize the total energy expenditure, where an iterative algorithm based on successive convex approximation technique is adopted to solve the formulated non-convex optimization problem. In [114], a distributed game theoretic approach is proposed to solve the NP-hard efficient computation offloading problem. Chen *et al.* [115] choose a model-free RL technique to solve the optimal traffic offloading strategy for heterogeneous cellular networks with dynamic traffic.

When caching technology is integrated into mobile networks, where to cache, what to cache, and how to cache are the major factors that make a significant influence on the system performance [110]. In [116], the authors investigate the proactive storage allocation problem over BSs in cellular networks, which is proved to be NP-hard. To get a low-complexity solution, a heuristic method is utilized and the convergence is proved by strict theoretic deduction. A caching-enabled D2D communication scheme is designed in [117], where the caching strategy is optimized by jointly considering

users' social relationships and common interests with the constraint of hit ratio, delay and caching capacity. In fact, contents with the most popularity should be cached within limited caching capacity. Since the content popularity is varying with time, learning based caching policies are proposed in [118]. The authors formulate the distributed caching over small BSs (SBSs) as a reinforcement learning problem, and with the help of coded caching, the complicated caching problem is solved by being reduced to a convex optimization problem. In [119], Qiao *et al.* consider the mobility pattern of mobile users, and formulate the mobility-aware caching problem as an optimization problem to maximize the caching utility. The problem is solved by using a polynominal-time heuristic solution. He *et al.* exploit deep reinforcement learning to address the resource allocation problems in cache-enabled interference alignment networks [120, 121].

The comprehensive resource allocation schemes for efficient integration of computing, caching and communication resources to achieve the optimal performance have been developed, even though not yet been widely investigated. In [122], Zhou *et al.* design a novel information-centric heterogeneous network framework, and with the virtualization technology, communication, computing and caching resources are shared among users. They formulate the virtual resource allocation strategy as a joint optimization problem, which is a non-convex NP hard problem. An alternating direction method of multiplier (ADMM) approach is used to solve the problem by simplifying and relaxing the non-convex problem into a convex one. ADMM method is also used in [123] to achieve the optimal resource allocation strategy in wireless networks with MEC and in-network caching being jointly considered. He *et al.* propose an integrated framework that can enable dynamic orchestration of networking, caching and computing resources, where the dynamics are modeled as finite-state Markov chains [124, 125]. The authors use the deep Q-leaning approach to solve the

complex problems with large numbers of system states and actions. Research on this topic of resource allocation strategies for integrated systems of computing, caching and communication will continue with the evolving of each of these three technologies.

5.1.2 Contributions

The main contributions of this chapter are summarized as follows.

- In this chapter, we consider trust-based social networks specifically with MEC, in-network caching and D2D communications under the umbrella of a 3C framework. An optimization problem is formulated to maximize the network operator's utility with comprehensive considerations of trust values, computation capabilities, wireless channel qualities, and the cache status of all the available BSs and D2D nodes.

- To be more realistic, we consider that the network conditions (i.e., trust value, computation capability, wireless channels, and cache status) are varying with time, and the dynamics of the computing capabilities, cache status and wireless channel conditions are modeled as Markov chains. In the meanwhile, the trust values are derived from both direct and indirect observations using Bayesian inference and Dempster-Shafer theory, respectively.

- The complexity of the integrated network is very high when we jointly consider the dynamic trust values, computation capabilities, wireless channel conditions and the cache status, and it is extremely difficult to solve the formulated optimization problem. In this chapter, we exploit deep Q-learning based resource allocation strategy to solve the optimization problem without any explicit assumptions or simplifications.

- Google TensorFlow is used to implement the proposed deep Q-learning approach. Simulation results with different system parameters are presented to show the effectiveness of the proposed scheme. It is illustrated that the performance of MSNs can be significantly improved with the proposed approach.

The remainder of this chapter is outlined as follows. We describe the system model in Section 5.2, which includes system description, network model, communication model, caching model and computing model. Next, the social trust scheme with uncertain reasoning is presented in Section 5.3. Section 5.4 formulates this system as a deep reinforcement learning problem. Simulation results are presented and discussed in Section 5.5. Finally, we give the chapter summary in Section 5.6.

5.2 System Model

In this section, we first present the system description. Next, the network model, communication model, cache model and computing model are presented, respectively.

5.2.1 System Description

MSNs have been developed rapidly to provide a variety of social services and applications to mobile users, focusing not only on the behaviors but also on the social needs of the users [21]. Compared to conventional mobile wireless networks, mobile users in MSNs do not always contact remote servers to request contents. Instead, mobile users in MSNs can directly obtain contents from each other within a community based on their social ties [107, 126].

In MSNs, huge amounts of information-rich data will be exchanged by mobile users. Although cloud computing is powerful, it is difficult for data centers to provide

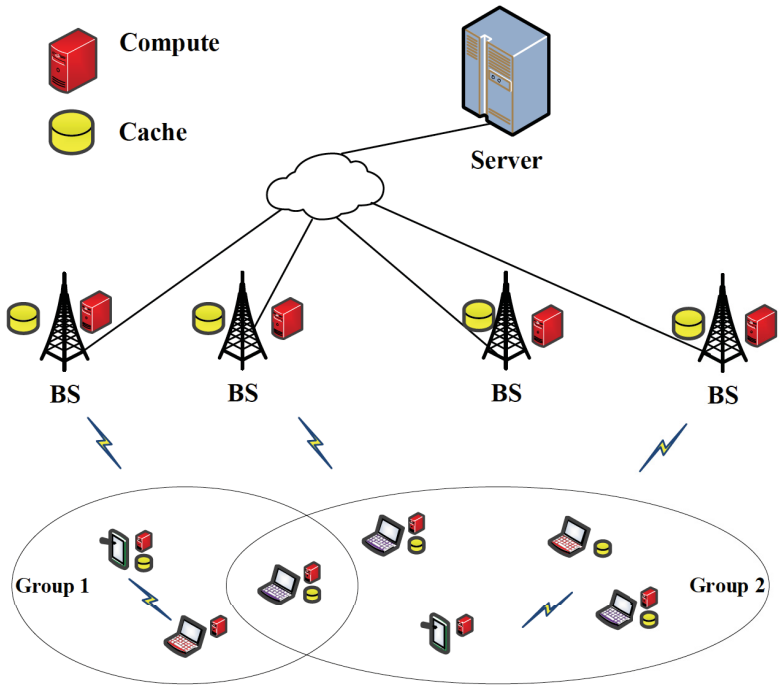

Figure 5.1: A mobile social network with edge computing, caching and device-to-device (D2D) communications.

mobile users with low-latency services. To address these issues, MEC has been proposed to deploy computing resources closer to end users, which can efficiently improve the quality of service (QoS) for applications that require intensive computations and low latency [91, 92]. MEC applies the concept of cloud computing in network edge nodes, to facilitate services and applications, including mobile social services and applications.

In addition, due to user mobility and poor-quality wireless radio links, it is challenging to deliver huge amounts of data using the traditional client-server approach in MSNs [22]. Recent advances of in-network caching can be extended to MSNs to address this issue, which can efficiently reduce the duplicate content transmission in networks. This innovative content-centric approach has been studied in MSNs, which natively privilege the information (e.g., trusted information in a specific proximity

of an event and a specific time period) rather than the node identity. In addition, with in-network caching, mobility and sporadic connectivity issues can be effectively addressed in MSNs.

Moreover, with D2D communications, users in close proximity can directly communicate with each other via D2D links, instead of accessing BSs exclusively. As a promising approach to offload traffic from BSs, D2D communications can enable the sharing of radio connectivity and direct information delivery between two close users [127, 128].

Security is always an important aspect in wireless applications and services [129, 130]. Trust-based security schemes are important detection-based approaches in MSNs. The definition of trust in MSNs is similar to that in sociology, where trust is interpreted as degrees of the belief that an entity will carry out tasks as expected [131]. In this chapter, we present a framework for trust-based social networks specifically with MEC, in-network caching and D2D communications, which is depicted in Fig. 5.1. Under the framework, we illustrate a video content request task as an example, which will be elaborated in the following network model.

5.2.2 Network Model

In this chapter, we consider the scenario with K BSs, and M mobile users that are considered as transmitters in the potential D2D communications. The network is operated by a central controller. Assume that there are L subscribed mobile users issuing video requests to the network operator, which is responsible for assigning a proper content provider to each requester, i.e., associating with one BS or setting up a one-to-one D2D communication. The sets are denoted as $\mathcal{C}_K = \{1, 2, ..., K\}$, $\mathcal{C}_M = \{1, 2, ..., M\}$ and $\mathcal{L} = \{1, 2, ..., L\}$, respectively. All the K BSs, the M D2D transmitters and the L mobile users are equipped with both caching and computing

capabilities. The unicast D2D communication is available only when the mobile user's requested video is stored in the transmitter's cache. In addition, the transmitters can also provision mobile edge computing if they have spare computing capacity left at the considered moment.

In our system model, we assume that the lth mobile user, denoted as s_l, issues a video request. First, the network controller checks all the video providers, i.e., all the BSs and D2D transmitters. We denote $\mathcal{C} = \mathcal{C}_K \bigcup \mathcal{C}_M$ be the set of video providers, and let $c_i (1 \leq i \leq K)$ be the ith BS and $c_j (K+1 \leq j \leq K+M)$ be the jth transmitter. Each video provider's cache is labeled by a content indicator which indicates whether or not the requested video is being stored.

If the requested video is being stored at any of the video provider's caches, and the versions match up as well, the network controller will built up a communication between mobile user s_l and the optimal video provider. However, if all the stored videos' versions mismatch with the requested one, video transcoding should be performed at either mobile user s_l itself or at the video provider's side. On the other hand, If the requested video is not being stored at any of the caches, it has to associate with a proper BS.

Note that in this work, we use video version to represent video specification (e.g., H.263, H.264, MPEG2, or MPEG4). With the rapid growth of mobile services, increasing volumes of videos are played by mobile devices. Consequently, service providers often need to transcode the video contents into different specifications (e.g., bit rate, resolution, quality, etc.) with different QoS (e.g., delay) for heterogeneous mobile devices, networks, and user preferences.

5.2.3 Communication Model

Denote $h_{s_l}^{c_p}$ as the channel gain between mobile user s_l and content provider $c_p, \forall l \in \mathcal{L}, \forall p \in \mathcal{C}$. According to the Shannon theorem, the achievable rate of mobile user s_l when associating with provider c_p can be expressed as

$$r_{s_l}^{c_p} = B \log_2 \left(1 + \frac{p_T h_{s_l}^{c_p}}{N_o B}\right), \tag{5.1}$$

where B denotes the bandwidth allocated to each mobile user, p_T is the equally transmitted power, and N_o is the noise spectral density.

Here, we consider the realistic wireless channels, and actually $h_{s_l}^{c_p}$ is a continuous random variable. Such assumption is intractable for analysis [132], and therefore we model the wireless channels as FSMC, which may achieve performance improvement compared with the traditional assumption of static channels. In our model, the channel gain $h_{s_l}^{c_p}$ is discretized and quantized into H levels: $\mathcal{H}_0$, if $h_0^* \leqslant h_{s_l}^{c_p} < h_1^*$; $\mathcal{H}_1$, if $h_1^* \leqslant h_{s_l}^{c_p} < h_2^*$; $\ldots$; $\mathcal{H}_{H-1}$, if $h_{s_l}^{c_p} \geqslant h_{H-1}^*$. The boundary values for a particular environment under certain criterion should be optimized for better performance [79]. However, for simplicity, uniformly setting the boundary values of FSMC is widely used in the literature (e.g., [79]). In this chapter, all the boundary values from h_1^* to h_{H-1}^* are increasing in the same distance with each other. Each level corresponds to a state of the Markov chain, and therefore a H-element state space is formed. Due to the relationship between $h_{s_l}^{c_p}$ and $r_{s_l}^{c_p}$, for convenience we use $r_{s_l}^{c_p}$ to describe a state of the wireless channel. We consider the dynamic process, and the channel state realization of $r_{s_l}^{c_p}$ at time instant t is denoted as$\Upsilon_{s_l}^{c_p}(t)$. Based on a certain transitional probability,$\Upsilon_{s_l}^{c_p}(t)$ varies from one state to another as one time slot is gone. The transitional probability of$\Upsilon_{s_l}^{c_p}(t)$ from one state j_s to another state k_s is denoted as $\psi_{j_s k_s}(t)$. The $H \times H$ channel state transitional probability matrix between

mobile user s_l and provider c_p is shown as:

$$\Psi_{s_l}^{c_p}(t) = [\psi_{j_s k_s}(t)]_{H \times H}\,, \tag{5.2}$$

where $\psi_{j_s k_s}(t) = \Pr\left(\Upsilon_{s_l}^{c_p}(t+1) = k_s \mid \Upsilon_{s_l}^{c_p}(t) = j_s\right)$.

5.2.4 Cache Model

Assume that there are totally I video contents in the network that mobile users can request for. The set of the contents is denoted as $\mathcal{I} = \{1, 2, ..., I\}$, which is ranked by popularity. Here, we consider finite cache capacity, i.e., each provider caches some of the I video contents, and refreshes the contents periodically. We consider that mobile user s_l has a video request $v_i, i \in \mathcal{I}$. After receiving the request message, the network controller checks every provider's content indicator for video v_i, i.e., $x_{c_p}^{v_i}, \forall p \in \mathcal{C}$. The content distribution indicator $x_{c_p}^{v_i} = 1$ means that video v_i is being stored in the cache of provider p; otherwise $x_{c_p}^{v_i} = 0$. Here, $x_{c_p}^{v_i}$ is viewed as a random variable, representing the state of the cache, and is modeled using a two-state (i.e., state 0 and 1) Markov chain [101]. The transitional probability matrix of the state $x_{c_p}^{v_i}$ is defined as:

$$\Gamma_{c_p}^{v_i}(t) = [\delta_{a_s b_s}(t)]_{2 \times 2}, \tag{5.3}$$

where $\delta_{a_s b_s}(t) = \Pr\left(x_{c_p}^{v_i}(t+1) = b_s \mid x_{c_p}^{v_i}(t) = a_s\right)$, and $a_s, b_s \in \{0, 1\}$.

When the cache capacity is assumed to be finite, the transitional probability matrix of the cache state can be derived based on different cache refreshment strategies. An important one is the least recently used (LRU) cache refreshment policy. The transitional probability matrix can be obtained using the following Markov chain

flow matrix [101],

$$\Lambda_i = \begin{bmatrix} -\gamma_i & 0 & \cdots & 0 & 0 & \gamma_i \\ \zeta_i + \mu_i & -\beta - \mu_i & \cdots & 0 & 0 & \gamma_i \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mu_i & 0 & \vdots & \zeta_i & -\beta - \mu_i & \gamma_i \\ \mu_i & 0 & \vdots & 0 & \zeta_i & -\zeta_i - \mu_i \end{bmatrix}. \tag{5.4}$$

Here $\zeta_i = \beta - \gamma_i$, and γ_i represents the average request rate for the ith popular video v_i that can be denoted as

$$\gamma_i(t) = \frac{\beta}{\rho i^{\alpha}}, \tag{5.5}$$

where the request for v_i is assumed to arrive as a Poisson process with rate β [101], and the probability follows a Zipf-like distribution. Therefore, the probability of requesting content v_i is $1/\rho i^{\alpha}$, where $\rho = \sum_{i=1}^{I} 1/i^{\alpha}$ and α is the Zipf slope with $0 < \alpha < 1$. The video content is refreshed periodically, and the lifetime of any video content v_i follows an exponential distribution with mean $1/\mu_i$.

5.2.5 Computing Model

On the condition that the versions of the requested video at all the providers' caches do not match with the requested version, the associated provider has to extract the current video content and construct a computation task based on the input data information, as well as the number of CPU cycles. In the computation model, we construct the computation task as $Q_{sl}^{v_i} = \{o_{sl}^{v_i}, q_{sl}^{v_i}\}$. The first parameter $o_{sl}^{v_i}$ represents

the size of the requested-version video, and the second parameter $q_{s_l}^{v_i}$ indicates the required number of CPU cycles that is needed to finish the computation task.

The computation capability of provider c_p assigned to mobile user s_l is denoted as $f_{s_l}^{c_p}$, which can be measured by the number of CPU cycles per second [114, 122]. In our network, multiple mobile users may associate with the same provider and share the computing device simultaneously, which would result in the fact that the computation capability for provider c_p is not exactly known at the next time instant. Therefore, the computation capability $f_{s_l}^{c_p}$ is modelled as a random variable, and equally divided into N discrete levels denoted by $\mathcal{E} = \{\mathcal{E}_0, \mathcal{E}_1, \ldots, \mathcal{E}_{N-1}\}$. The realization of the random variable $f_{s_l}^{c_p}$ is denoted as $F_{s_l}^{c_p}$ at time slot t. We model the transition of the computation capability level of a provider as a Markov process. The transitional probability matrix of $F_{s_l}^{c_p}$ can be expressed as:

$$\Theta_{s_l}^{c_p}(t) = \left[\iota_{x_s y_s}(t)\right]_{N \times N}, \tag{5.6}$$

where $\iota_{x_s y_s}(t) = \Pr\left(F_{s_l}^{c_p}(t+1) = y_s \mid F_{s_l}^{c_p}(t) = x_s\right)$, and $x_s, y_s \in \mathcal{E}$.

The consumed time for executing the task $Q_{s_l}^{v_i}$ at provider c_p can be obtained as $t_{s_l,c_p} = \frac{q_{s_l}^{v_i}}{F_{s_l}^{c_p}(t)}$. Furthermore, the rate of computation, i.e., the number of bits computed per second, can be given

$$r_{s_l,c_p}^{\text{comp}}(t) = a_{s_l,c_p}^{\text{comp}}(t)\frac{o_{s_l}^{v_i}}{t_{s_l,c_p}} = a_{s_l,c_p}^{\text{comp}}(t)\frac{f_{s_l}^{c_p}(t)o_{s_l}^{v_i}}{q_{s_l}^{c_p}}, \tag{5.7}$$

where $a_{s_l,c_p}^{\text{comp}}(t) \in \{0, 1\}$ represents whether or not the computation task is decided to be performed at provider c_p.

5.3 Social Trust Scheme with Uncertain Reasoning

In this section, we will derive how to obtain the trust values of mobile users. We evaluate the trustworthiness of a mobile user by a real number Tr ranging from 0 to 1. In our model, the trust value Tr is jointly determined based on direct observations and indirect observations. The direct observation trust of a mobile user is defined as the estimated degree of trustworthiness from its directly-connected mobile users based on their past experiences. However, the subjective evaluation from direct connections may be prejudiced, and therefore in order to be more objective and impartial, we also consider the rating of trust from other indirectly-connected mobile users. Here, we denote the trust value from direct observations as Tr^D and the trust value from indirect observations as Tr^{InD}. By combining these two trust values, we can obtain a more accurately estimated trust value of a mobile user as

$$Tr = \omega Tr^D + (1 - \omega) Tr^{InD} \tag{5.8}$$

where ω is the weight coefficient to adjust the weightiness between direct and indirect observations, and $0 \leq \omega \leq 1$.

The trust evaluation procedure in our model can be visually explained as Fig. 5.2. In the following, we will discuss how to obtain the trust evaluation from direct observations and indirect observations by using the Bayesian inference approach and Dempster-Shafer theory [133], respectively.

5.3.1 Trust Evaluation from Direct Observations

In the direct observations, consider that an observing mobile user can overhear the data forwarded by the observed mobile user, and identify the observed mobile user's

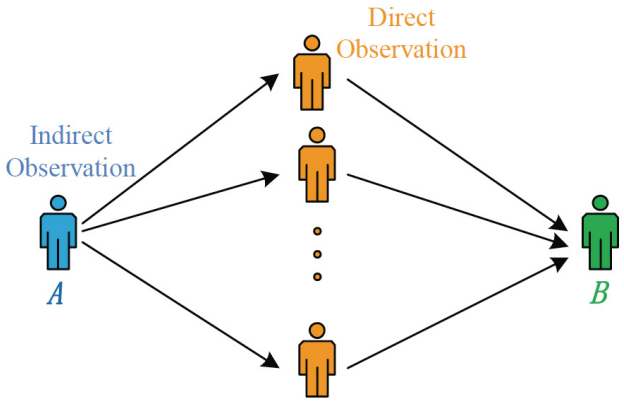

Figure 5.2: Social trust evaluation with both direction observation and indirect observation.

malicious behaviors, such as discarding or modifying some of the original data.

Through multiple observations of the observed mobile user's behavior, the observing mobile user can evaluate the trust value by exploiting Bayesian inference [134], which is a method of statistical inference using the Bayes' theorem to update the probability for a hypothesis with more evidence becomes available.

Under the Bayesian framework, we model the trust of a mobile user as a continuous random variable, denoted as Φ, where ϕ takes values from 0 to 1. We assume thatΦ follows a beta distribution [135], i.e.,$\Phi \sim Beta(a, b)$, which is defined as follows with parameters a and b

$$Beta(a, b) = \frac{\phi^{a-1}(1-\phi)^{b-1}}{\int_0^1 \phi^{a-1}(1-\phi)^{b-1}\,\mathrm{d}\phi}, \tag{5.9}$$

for $0 \leq \phi \leq 1$. Here, since Φ is assumed to obey a beta distribution, the trust value can be represented by the two parameters a and b.

We summarize our belief of the trust Φ in a probability distribution iteratively as more observations are available. Assume that the prior probability density function

(pdf) at the $(t-1)$th observation is known. Then, according to the Bayes theorem, the posterior distribution at the tth observation can be obtained with the pdf as

$$f_t(\phi) = \frac{f_t(x_t|\phi, y_t) f_{t-1}(\phi)}{\int_0^1 f_t(x_t|\phi, y_t) f_{t-1}(\phi)\, \mathrm{d}\phi}, \tag{5.10}$$

where x_t and y_t are the number of data packets that has been forwarded correctly and the number of packets received by the observed mobile user at the tth observation, respectively; $f_t(x_t|\phi, y_t)$ is the likelihood function, which follows a binomial distribution as

$$f_t(x_t|\phi, y_t) = \binom{y_t}{x_t} \phi^{x_t} (1-\phi)^{y_t - x_t}. \tag{5.11}$$

In Bayesian inference, the beta distribution is the conjugate prior probability distribution for the binomial distribution [134, 135]. Since the likelihood fuction $f_t(x_t|\phi, y_t)$ follows a binomial distribution, the priori distribution $f_{t-1}(\phi)$ is definitely assumed to follow a beta distribution, which reflects what is already known about the distribution of Φ at the $(t-1)$th observation. Given that the priori distribution $f_{t-1}(\phi)$ follows a beta distribution, the posterior distribution $f_t(\phi)$ also follows a beta distribution. Particularly, if $f_{t-1}(\phi) \sim Beta(a_{t-1}, b_{t-1})$ and x_t, y_t from the tth observation are also given, then

$$f_t(\phi) \sim Beta(a_{t-1} + x_t, b_{t-1} + y_t - x_t), t \geq 1. \tag{5.12}$$

At the very beginning of the observation, there is no evidence about the distribution ofΦ , thus we assume that it follows a uniform distribution, i.e., $f_0(\phi) \sim Beta(1, 1)$. Summarily, we can say that $f_t(\phi)$ follows $Beta(a_t, b_t)$ with parameters

$$a_t = a_{t-1} + x_t, \qquad b_t = b_{t-1} + y_t - x_t,$$

$$a_0 = 1, \qquad b_0 = 1. \tag{5.13}$$

The trust value can be represented with the mathematical expectation of beta distribution as

$$\mathbb{E}_t[\Phi] = \frac{a_t}{a_t + b_t}. \tag{5.14}$$

Intuitively, the trust value of a mobile user is 0.5 at the beginning, and updated continuously by using the follow-up observations.

From the above discussion, we can see that past experiences play an important role in the Bayesian inference. In fact, recent behaviors of a mobile user should weight more in the trust evaluation. Here, we introduce a punishment factor for reputation fading, which gives more weights on misbehaviors in the Bayesian inference. The trust evaluation formula in (5.14) is revised as follows:

$$\mathbb{E}_t[\Phi] = \frac{a_t}{a_t + \tau b_t}, \tag{5.15}$$

where τ is the punishment factor, and $\tau \geq 1$. With the increment of τ, the trust value declines quickly.

The punishment factor makes the trust evaluation more realistic and reliable. First, if a mobile user once behaves maliciously, compared with those who have no bad records, its trust value will be lowered much more. Second, the trust value will not recover quickly even if he behaves well recently because of the constraint of the punishment factor. This helps distinguish the malicious mobile users quickly and prevent them disrupting others' trust evaluation. Based on the above deduction, the trust value from direct observation, Tr^D, is defined as:

$$Tr^D = \mathbb{E}_t[\Phi]. \tag{5.16}$$

5.3.2 Trust Evaluation from Indirect Observations

Apart from the direct observations, indirect observations from other mobile users are also very important in assessing the trustworthiness of an observed mobile user. Considering the indirect observations helps mitigate the situation that an observed mobile user is loyal to one mobile user but cheating on others. Assume that an observing mobile user collects observations from several other mobile users (also called subsidiary observing mobile user), and combines the collected evidence into a decision about the observed mobile user's trust value. However, these subsidiary observing mobile users may be untrustworthy or the evidence offered by them is unreliable.

The Dempster-Shafer theory can be used as an effective way to handle the uncertainty issue and combine the evidence from multiple subsidiary observers [136]. The core of this theory is based on two ideas: the degrees of belief about a proposition can be obtained from multiple subjective probabilities of a related theme, and these degrees of belief can be combined together under the condition that they are from independent evidence [133]. In the indirect observation, we assume that there are more than one subsidiary observing mobile users and the evidence provided by them is mutually independent.

Belief Function As an introduction to the belief function, suppose that each subsidiary observing mobile user has two states, i.e., trustworthy and untrustworthy with probabilities p_1 and $1 - p_1$ respectively. Assume that mobile user 1 claims that the observed mobile user B is trustworthy. If mobile user 1 is trustworthy, its statement is regarded as true; however, if mobile user 1 is untrustworthy, its statement is not necessarily untrue. We think that mobile user 1 provides p_1 degree of belief in the observed mobile user's trustworthiness (hypothesis H), 0 degree of belief in

the untrustworthiness (hypothesis $\overline{H}$), and $1 - p_1$ degree in the uncertainty, i.e., either trustworthy or untrustworthy (hypothesis U). Each hypothesis is assigned with a basic probability value $m(H)$, $m(\overline{H})$, $m(U)$ taking values from the interval $[0, 1]$, respectively. In our scheme, we assign the basic probability value with the direct observation trust value. For example, if mobile user 1 believes that mobile user B is trustworthy, then the basic probability value for each possible hypothesis is:

$$\begin{aligned} m_1(H) &= T_{A1}^D, \\ m_1(\overline{H}) &= 0, \\ m_1(U) &= 1 - T_{A1}^D, \end{aligned} \tag{5.17}$$

where T_{A1}^D represents the trust value of mobile user 1 from the direct observation of mobile user B.

Oppositely, if mobile user 1 thinks that user B is untrustworthy, the basic probability value will be:

$$\begin{aligned} m_1(H) &= 0, \\ m_1(\overline{H}) &= T_{A1}^D, \\ m_1(U) &= 1 - T_{A1}^D. \end{aligned} \tag{5.18}$$

Concerning the formal definition of the belief function, let Ω be the universe of discourse, i.e., the set representing all possible states of the considered system. $\Omega = \{trustworthy, untrustworthy\}$. The power set 2^Ω refers to the set of all subsets ofΩ , here $2^\Omega = \{\varnothing, \{trustworthy\}, \{untrustworthy\}, \Omega\}$. Any hypothesis A_i refers to a subset of 2^Ω, and is mapped into a basic probability $m(A_i)$ that reflects the proportion of total belief assigned to hypothesis A_i. These two conditions should be satisfied: $m(\varnothing) = 0$ and $\sum_{A_i \subseteq \Omega} m(A_i) = 1$. For any hypothesis B, the belief function

is defined as

$$beli(B) = \sum_{A_i \subseteq B} m(A_i), \tag{5.19}$$

which represents the strength of the evidence that supports hypothesis B's provability.

Dempster's Rule of Combining Belief Functions Based on the definition of belief function, the Dempster-Shafer theory combines multiple users' belief. Assuming that $beli_1(B)$ and $beli_2(B)$ are two belief functions over the same universe of discourse, Ω. the orthogonal sum of $beli_1(B)$ and $beli_2(B)$ can be defined as

$$\begin{aligned} beli(B) =& beli_1(B) \oplus beli_2(B) \\ =& \frac{\sum_{i,j,A_i \bigcap A_j = B} m_1(A_i) m_2(A_j)}{\sum_{i,j,A_i \bigcap A_j \neq \varnothing} m_1(A_i) m_2(A_j)}, \end{aligned} \tag{5.20}$$

where $A_i, A_j \subseteq \Omega$.

In our scenario, the combined degree of belief from mobile user 1 and mobile user 2 can be calculated as follows [133]:

$$\begin{aligned} m_1(H) \oplus m_2(H) =& \frac{1}{K}[m_1(H)m_2(H) + m_1(H)m_2(U) \\ & + m_1(U)m_2(H)], \\ m_1(\overline{H}) \oplus m_2(\overline{H}) =& \frac{1}{K}[m_1(\overline{H})m_2(\overline{H}) + m_1(\overline{H})m_2(U) \\ & + m_1(U)m_2(\overline{H})], \\ m_1(U) \oplus m_2(U) =& \frac{1}{K} m_1(U)m_2(U), \end{aligned} \tag{5.21}$$

where

$$\begin{aligned} K =& m_1(H)m_2(H) + m_1(H)m_2(U) + m_1(U)m_2(U) \\ &+ m_1(U)m_2(H) + m_1(U)m_2(\overline{H}) + m_1(\overline{H})m_2(\overline{H}) \\ &+ m_1(\overline{H})m_2(U). \end{aligned} \tag{5.22}$$

Following the rule of combination of belief, we can combine more degree of belief from other mobile users. Based on the Dempster-Shafer theory, T_{AB}^{N} is defined as:

$$Tr_{AB}^{InD} = m_1(H) \oplus m_2(H) \ldots \oplus m_n(H), \tag{5.23}$$

where there are totally between mobile user A and mobile user B.

5.4 Problem Formulation

In this section, we formulate an optimization problem of the integrated trust-based social network with the 3C framework with MEC, caching, and D2D communications. We assume that a mobile user requests for a video content to the integrated network. For the network operator, it should decide which BS or a D2D transmitter is assigned to serve the requesting user, whether or not the video transcoding (i.e., MEC) should be performed, and whether or not newly emerged contents should be cached. The network operator need a comprehensive consideration of many factors, including: the wireless channel conditions, whether or not the requested content is stored at the local cache, whether or not the content version is matched up, the computational capacity, the trustworthiness of a D2D transmitter. Here, we consider dynamic scenarios, i.e., the available network conditions are varying with time. We exploit deep Q-learning algorithm to solve the formulated optimization problem.

In order to obtain the optimal policy, identifying the system's states, actions, and

reward functions is necessarily required, which will be described in more details below.

5.4.1 System State

The system states for a subscriber s_l requesting video v_i at time slot $t \in \{0, 1, \ldots, T-1\}$ mainly includes five components: the realization$\Upsilon_{s_l}^{c_p}(t)$ of the random variables $\gamma_{s_l}^{c_p}$ (channel state), the realization $F_{s_l}^{c_p}(t)$ of the random variables $f_{s_l}^{c_p}$ (computation capability), the realization $X_{v_i}^{c_p}(t)$ of the random variables $x_{v_i}^{c_p}$ (content indicator), the version indicator $y_{v_i}^{c_p}$, and the trust value index $Tr_{s_l}^{c_p}$ for all the video providers, i.e., $\forall p \in \mathcal{C}$. Consequently, the system state vector can be described as follows.

$$\begin{aligned}\chi_{s_l}(t) = \big[&\Upsilon_{s_l}^{c_1}(t), \Upsilon_{s_l}^{c_2}(t), \ldots, \Upsilon_{s_l}^{c_{K+M}}(t),\\ &F_{s_l}^{c_1}(t), F_{s_l}^{c_2}(t), \ldots, F_{s_l}^{c_{K+M}}(t),\\ &X_{v_i}^{c_1}(t), X_{v_i}^{c_2}(t), \ldots, X_{v_i}^{c_{K+M}}(t),\\ &y_{v_i}(t),\\ &Tr_{s_l}^{c_1}(t), Tr_{s_l}^{c_2}(t), \ldots, Tr_{s_l}^{c_{K+M}}(t)\big].\end{aligned} \tag{5.24}$$

Here, the trust index for all the BSs should be set to 1, i.e., $Tr_{s_l}^{c_p}(t) = 1, \forall c_p \in \mathcal{C}_K$; meanwhile, this index for all the D2D transmitters should be less than 1, i.e., $Tr_{s_l}^{c_p}(t) < 1, \forall c_p \in \mathcal{C}_M$. $y_{v_i}(t)$ denotes the indicator for whether or not the cached video version matches with the requested version. If yes $y_{v_i}(t) = 1$, if no $y_{v_i}(t) = 0$.

5.4.2 System Action

In our system, the network controller decides which video provider is assigned to the subscribed user, whether or not the computation offloading (video transcoding) should be performed, and whether or not the video provider should cache the new

video. The network controller's composite action for subscriber s_l is given by

$$\mathbf{a}_{s_l}(t) = \{\mathbf{a}_{s_l}^{\text{comm}}(t), \mathbf{a}_{s_l}^{\text{comp}}(t), \mathbf{a}_{s_l}^{\text{cache}}(t)\}, \tag{5.25}$$

where the row vectors $\mathbf{a}_{s_l}^{\text{comm}}(t), \mathbf{a}_{s_l}^{\text{comp}}(t), \mathbf{a}_{s_l}^{\text{cache}}(t)$ are interpreted in more details as follows:

- The first row vector $\mathbf{a}_{s_l}^{\text{comm}}(t)$ is defined as

$$\mathbf{a}_{s_l}^{\text{comm}}(t) = [a_{s_l,c_1}^{\text{comm}}(t), a_{s_l,c_2}^{\text{comm}}(t), \ldots, a_{s_l,c_{K+M}}^{\text{comm}}(t)], \tag{5.26}$$

  where $a_{s_l,c_p}^{\text{comm}}(t)$ is the association indicator, and $a_{s_l,c_p}^{\text{comm}}(t) \in \{0,1\}$. If the subscribed user is associated with video provider c_p, $a_{s_l,c_p}^{\text{comm}}(t) = 1$, otherwise $a_{s_l,c_p}^{\text{comm}}(t) = 0$.

- The second row vector $\mathbf{a}_{s_l}^{\text{comp}}(t)$ is defined as

$$\mathbf{a}_{s_l}^{\text{comp}}(t) = [a_{s_l,c_1}^{\text{comp}}(t), a_{s_l,c_2}^{\text{comp}}(t), \ldots, a_{s_l,c_{K+M}}^{\text{comp}}(t)], \tag{5.27}$$

  where $a_{s_l,c_p}^{\text{comp}}(t)$ indicates the computation offloading decision, and $a_{s_l,c_p}^{\text{comp}}(t) \in \{0,1\}$. If the network controller has decided that the computation task should be performed at the provider c_p's device, $a_{s_l,c_p}^{\text{comp}}(t) = 1$; on the other hand, if the computation is decided to be executed on subscriber's own mobile device, $a_{s_l,c_p}^{\text{comp}}(t) = 0$.

- The third row vector $\mathbf{a}_{s_l}^{\text{cache}}(t)$ is defined as

$$\mathbf{a}_{s_l}^{\text{cache}}(t) = [a_{s_l,c_1}^{\text{cache}}(t), a_{s_l,c_2}^{\text{cache}}(t), \ldots, a_{s_l,c_{K+M}}^{\text{cache}}(t)], \tag{5.28}$$

  where $a_{s_l,c_p}^{\text{cache}}(t)$ means the whether or not the video provider c_p should cache the

newly-emerged video or the new version, and $a^{\text{cache}}_{s_l,c_p}(t) \in \{0,1\}$. If the network controlled decides that c_p caches the video, $a^{\text{cache}}_{s_l,c_p}(t) = 1$, otherwise $a^{\text{cache}}_{s_l,c_p}(t) = 0$.

5.4.3 Reward Function

In this chapter, we set the system reward to be the total revenue (i.e., utility) of the network operator. The network operator charges the subscribed user s_l for associating with the video provider, which is denoted as λ_{s_l} unit price per bps. On the condition that video transcoding (computation offloading) is decided to be executed on video provider's side, the operator can charge for its computing service, which is defined as ν_{s_l} unit price per bps. In addition, if the network controller decides to let the BSs or D2D transmitters cache the new video or new version, the operator gets a potential revenue on estimated backhaul save, which is denoted as κ_{s_l} unit price per Hz.

On the other hand, the operator has to pay for the rented spectrum and backhaul bandwidth. The unit price for the usage of spectrum is defined as δ_{c_p} per Hz for provider c_p. Moreover, if video transcoding is performed, a certain amount of energy will be consumed for the computing, the network operator is obliged to pay η_{c_p} unit price for per consumed Joule. The cost of caching the video content in the memory of provider c_p is denoted as ς_{c_p} unit price per unit space.

The system reward for serving subscribed user s_l requesting video v_i is defined as the network operator's total revenue, and it is formulated as a function of the system states, and actions. The system actions determine whether or not the reward can be

optimized. Here, we define the reward function for a specific subscribed user s_l as:

$$
\begin{aligned}
R_{s_l}^{v_i}(t) &= \sum_{c_p \in \mathcal{C}} \left[R_{s_l,c_p}^{\text{comm}}(t) + R_{s_l,c_p}^{\text{comp}}(t) + R_{s_l,c_p}^{\text{cache}}(t) \right] \\
&= \sum_{c_p \in \mathcal{C}} a_{s_l,c_p}^{\text{comm}}(t) \left(\lambda_{s_l} Tr_{s_l}^{c_p}(t) \Upsilon_{s_l}^{c_p}(t)(1 - w^{\text{comp}} a_{s_l,c_p}^{\text{comp}}(t)) - \delta_{c_p} B \right. \\
&\quad \left. - (1 - X_{v_i}^{c_p}(t)) \sigma^{c_p} \Upsilon_{s_l}^{c_p}(t) \right) \\
&\quad + \sum_{c_p \in \mathcal{C}} (1 - y_{v_i}^{c_p}) a_{s_l,c_p}^{\text{comp}}(t) (\nu_{s_l} Tr_{s_l}^{c_p}(t) \frac{F_{s_l}^{c_p}(t) o_{s_l}^{v_i}}{q_{s_l}^{v_i}} - \eta_{c_p} q_{s_l}^{c_p} e_{c_p}) \\
&\quad + \sum_{c_p \in \mathcal{C}} a_{s_l,c_p}^{\text{cache}}(t) (\kappa_{s_l} Tr_{s_l}^{c_p}(t) \Upsilon_{s_l}^{c_p}(t) - \varsigma_{c_p} o_{s_l}^{v_i}).
\end{aligned}
\tag{5.29}
$$

The above reward function is composed of three terms, i.e., the earnings from communications, computing and caching, respectively. Here, the trust index $Tr_{s_l}^{c_p}(t)$ is added to each incoming revenue term. For the first communication earnings, $\lambda_{s_l} Tr_{s_l}^{c_p}(t) \Upsilon_{s_l}^{c_p}(t)(1 - w^{\text{comp}} a_{s_l,c_p}^{\text{comp}}(t))$ denotes the available earnings for providing the video transmitting service to subscriber s_l. Note that if video transcoding is decided to be performed, the video will be compressed and the corresponding income will be reduced. $\delta_{c_p} B$ is the cost for consuming the radio spectrum bandwidth. $(1 - X_{v_i}^{c_p}(t)) \sigma^{c_p} \Upsilon_{s_l}^{c_p}(t)$ is the cost for possible consumed backhaul bandwidth: if $X_{v_i}^{c_p}(t) = 1$, i.e., the requested video exists in the cache of provider c_p, no backhual cost is needed; otherwise $X_{v_i}^{c_p}(t) = 0$, the video has to be fetched from the Internet and backhaul cost is unavoidable. Note that σ^{c_p} is a very important parameter. It is not realistic to build up a D2D communication when the requested video does not exist in its cache. Thus, σ^{c_p} should be set to be an extremely large number for $c_p, \forall c_p \in \mathcal{C}_M$. For the second computing earnings, $(1 - y_{v_i}^{c_p})$ is used to indicate whether or not the version is matching. $\nu_{s_l} Tr_{s_l}(t)^{c_p} \frac{F_{s_l}^{c_p}(t) o_{s_l}^{v_i}}{q_{s_l}^{v_i}}$ represents the gaining for implementing video transcoding, and $\eta_{c_p} q_{s_l}^{c_p} e_{c_p}$ is the expenditure for the energy consumption, where e_{c_p} denotes consumed energy for running one CPU cycle. For the third caching earnings,

$\kappa_{s_l} Tr_{s_l}(t)^{c_p} \Upsilon_{s_l}^{c_p}(t)$ is the estimated future income of the saved backhaul bandwidth if the new video or new version is decided to be cached. Here the saved backhaul bandwidth is equal to the wireless communication rate. At last, $\varsigma_{c_p} o_{s_l}^{v_i}$ denotes the cost for caching the content.

$R_{s_l}^{v_i}(t)$ is the system's immediate reward, i.e., the network operator gets $R_{s_l}^{v_i}(t)$ at state $\chi_{s_l}(t)$ when action $\mathbf{a}_{s_l}(t)$ is performed in time slot t. However, a maximum immediate value is not a guarantee for the maximum long-term future rewards. Thus, we should also consider a big picture. A future reward with a discount factor φ is reasonable, which can be denoted as

$$R_{s_l,v_i}^{f} = \max_{\pi} \mathbb{E}\left[\sum_{t=0}^{t=T-1} \varphi^t R_{s_l}^{v_i}(t)\right], \tag{5.30}$$

where π represents a Q-learning policy, i.e., a series of actions where a specific action will be executed given a specific system state, and φ^t approximates to be zero when T is large enough. In fact, a condition for terminating the training process should be set.

The objective of adopting deep Q-learning into our network model is to help find an optimization policy that can maximize the cumulated future rewards for the network operator.

5.5 Simulation Results and Discussions

In this section, we evaluate the performance of proposed scheme using computer simulations. We use TensorFlow [74] in our simulations to implement deep Q-learning. For performance comparison, the following four algorithms are presented: 1). Proposed scheme, which considers MEC, caching, D2D, as well as both direct observation and indirect observation; 2). Proposed scheme w.o. indirect observation, which does

not consider indirect observation; 3). An existing scheme w.o. mobile edge computing [137]; and 4). An existing scheme w.o. D2D communications [22].

5.5.1 Simulation Settings

In the simulations, we consider an MSN consisting of 5 BSs, and 15 D2D transmitters. The radius of the cell is set to 500m. Because D2D communications typically perform within short ranges, we use a clustered-based distribution model [138], where multiple users are located within one cluster with a radius of 50m. In the network, there are 20 subchannels, each of which has a bandwidth of 180KHz. The transmit powers of D2D transmitter and BS are 24dBm and 46dBm, respectively. The noise spectral density is -174dBm/Hz. A loss model $35.3 + 37.6 \log(d(m))$ [139] is used. In addition, block fading with a block size of 100 is assumed for channel fading. Moreover, we assume that there are totally 10 types of contents distributed in the network, and each content cache state follows the Markov model. We set the cache state transition probability of staying in the same state in the BS as 0.6 (0.3 in the D2D transmitter), and set the transition probability from one state to another as 0.4 (0.7 in the D2D transmitter). We further assume a Zipf popularity distribution in the simulations, with $\theta = 1.5$ [140]. The computation states of MEC servers follow the Markov model. We assume that the computation state of the MEC server can be very low, low, medium, high, and very high.

We assume that there are two types of D2D transmitters in the network: normal nodes, which share the content and perform computing normally, and compromised nodes, which modify contents maliciously. The BSs are assumed to be not compromised due to the physical security of BSs. We also assume that the number of compromised nodes is much less than the total number of nodes in the network. The attackers are independent. Hence, there is no collusion attack in the network.

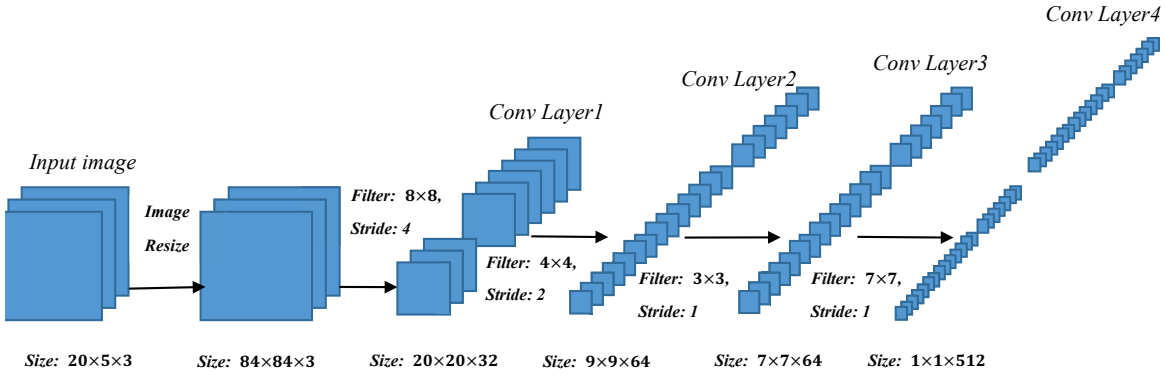

Figure 5.3: The training process of the deep convolution networks.

In our simulations, we used a GPU-based server, which has 4 Nvidia GPUs with version GTX TITAN. The CPU is Intel Xeno E5-2683 v3 with 128G memory. The software environment we utilized is TensorFlow 0.12.1 with Python 2.7 on Ubuntu 14.04 LTS.

The training process of the deep convolutional networks are shown in Fig. 5.3. We formulate the 5 BSs and 15 D2D transmitters as the rows, and the 5 levels of computational capacity as the columns in a grid image. Whether or not the required content is in the local cache is characterized by different colors in each small grid. The initial input image of size $20 \times 5 \times 3$ is firstly resized into $84 \times 84 \times 3$. Through 4 layers of convolutional operations, the output is 512 nodes, and these nodes will be fully connected to be trained in the deep reinforcement learning.

5.5.2 Simulation Results and Discussions

The effects of average size per content on the total reduced backhaul usage is shown in Fig. 5.4. There two malicious D2D transmitters, and there are 4 types of contents in the system. We can see from Fig. 5.4 that the total reduced backhaul usage

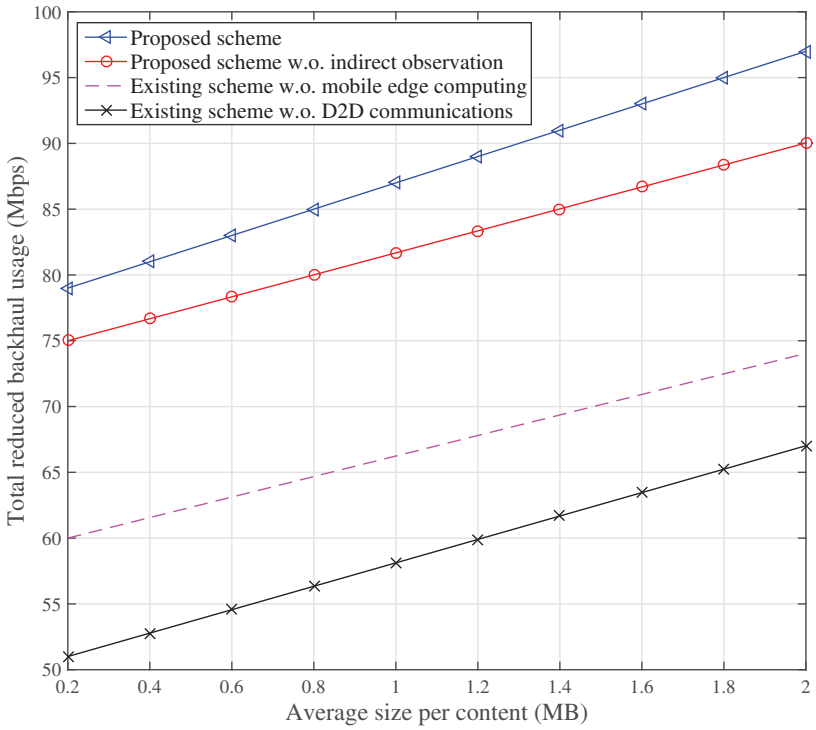

Figure 5.4: The total reduced backhaul usage v.s. average size per content.

increases with the increase of the average size per content. This is because a larger content size will increase the fee for caching the content, which induces a lower gain of caching utility, and thus the system is reluctant to cache the contents in the network. Compared to the existing schemes without mobile edge computing and D2D communications, the proposed scheme has larger total reduced backhaul usage due to the benefits of mobile edge computing and D2D communications. In addition, without indirect observation, the accuracy of trust evaluation is lower in the proposed scheme, which induces a lower gain of mobile edge computing and caching, and lower total reduced backhaul usage.

Fig. 5.5 and Fig. 5.6 show the effects of the number of content types on the total

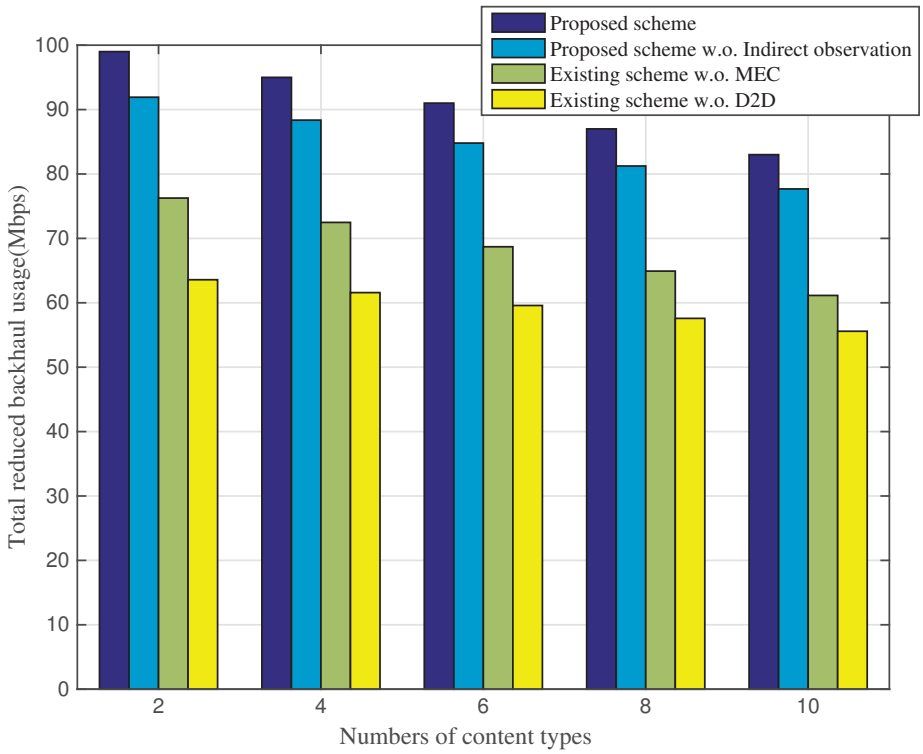

Figure 5.5: The total reduced backhaul usage v.s. the number of content types.

reduced backhaul usage and the total utility, respectively. In these simulations, the average size per content is 1.8 MB, and there are two malicious D2D transmitters. We can see from both figures that the reduced backhaul usage and the total utility decrease as the number of content types increases. This is because more content types in the system will lead to the scenarios that a requesting mobile user is unlikely to find the specific content from the BSs and D2D transmitters due to the limited storage. In addition, more content types will result in reduced the popularity of all contents according to the Zipf popularity distribution, thus decreasing caching gain.

Fig. 5.7 shows the effects of the required numbers of CPU cycles for video transcoding. From Fig. 5.7, we can see that the required number of CPU cycles

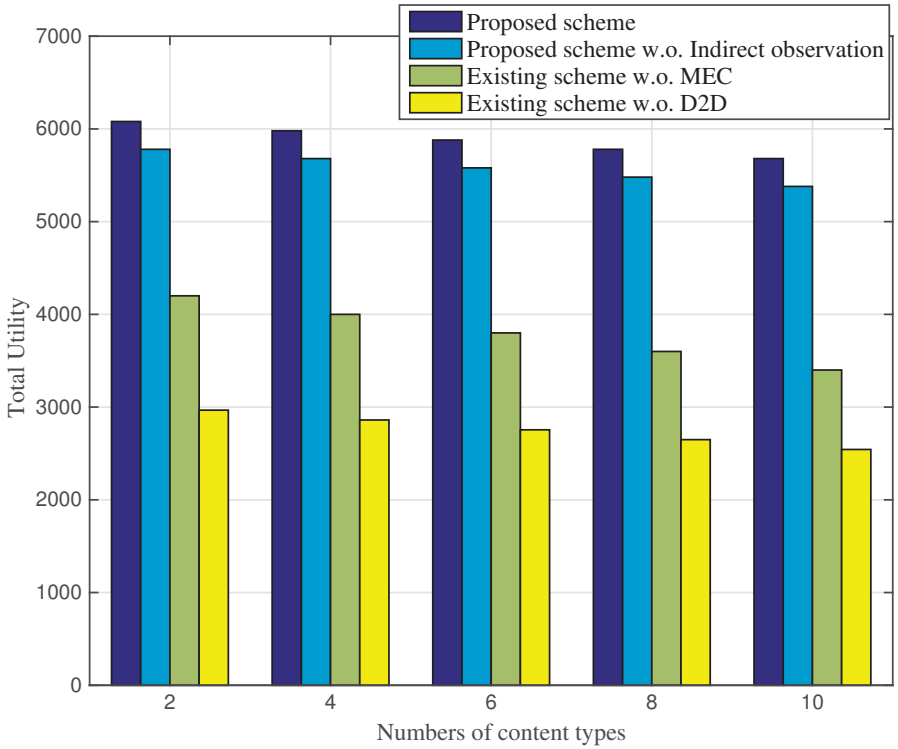

Figure 5.6: The total utility v.s. the number of content types.

has no effects on the total utility of the existing scheme without mobile edge computing. This is because the total utility will not be changed with different required numbers of CPU cycles if mobile edge computing is not available in the system. In the proposed scheme and the existing scheme without D2D communications, the total utility decreases exponentially with the increase of the required number of CPU cycles for video transcoding. This is because a larger number of required number of CPU cycles will result in a higher consumed computation energy, and consequently a lower gain of computation utility. Therefore, the total utility decreases with the required number of CPU cycles for video transcoding.

Next, we study the performance of the proposed social trust scheme. Assume that, at episode 14K, D2D transmitter 1 changes its maliciousness to 0.6. Our goal

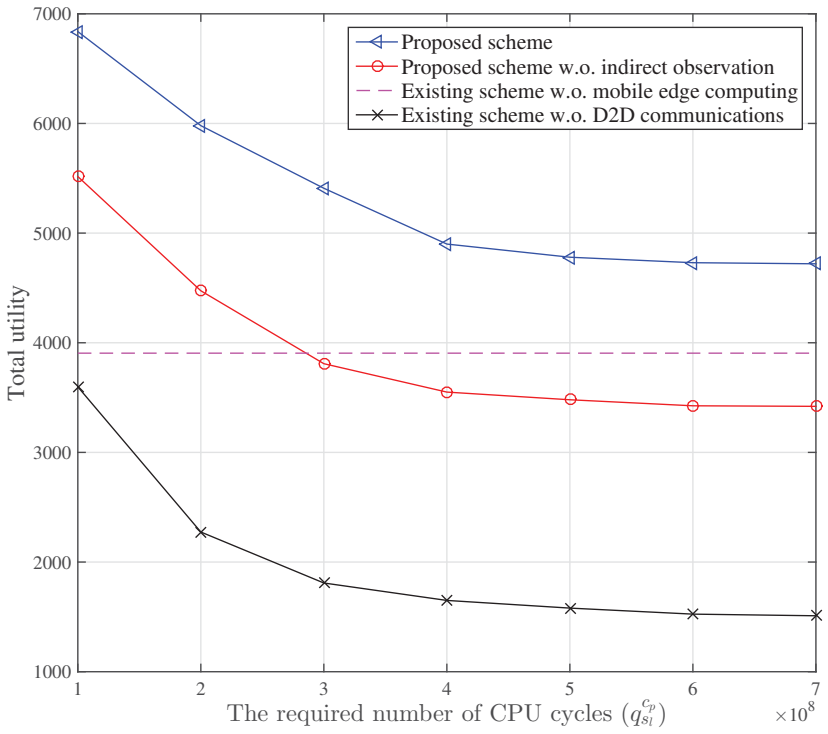

Figure 5.7: The total utility v.s. the required number of CPU cycles for video transcoding.

is to observe the accuracy of trust tracking in this scenario where the malicious behavior of a D2D transmitter changes rapidly. Fig. 5.8 shows the trust tracking of the system using direct observation with Bayesian inference and the proposed scheme using both direct observation with Bayesian inference and indirect observation with the Dempster-Shafer theory. We can observe from Fig. 5.8 that only direct observation can result in inaccurate trust values. By contrast, the proposed scheme can track the trust value accurately with both direct observation and indirect observation.

The number of malicious D2D transmitters in the network also has a significant impact on the performance the network. Here, We investigate the system utility

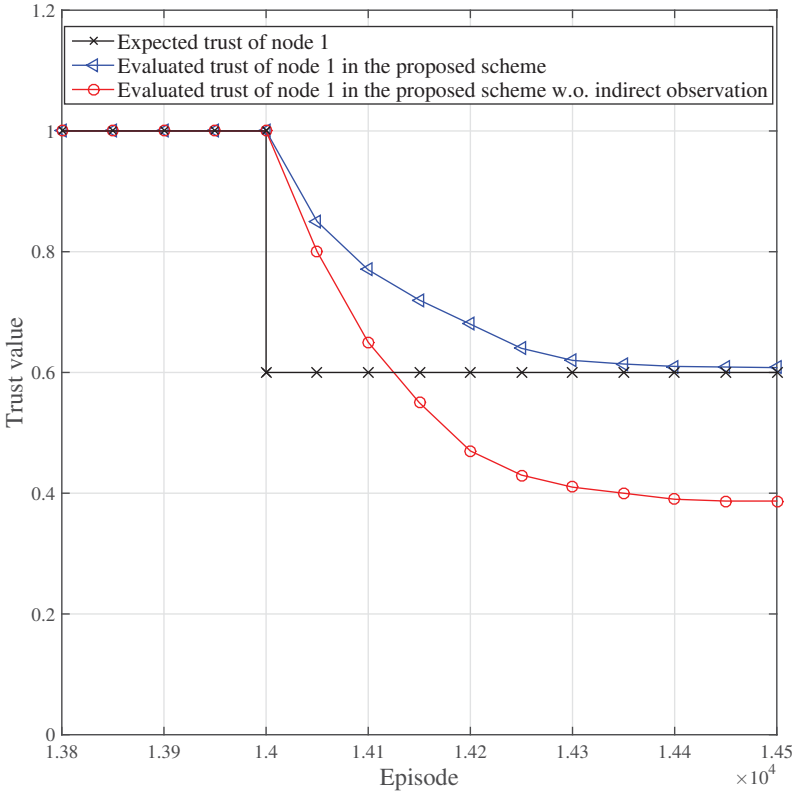

Figure 5.8: The trust value v.s. the episodes.

with the number malicious D2D transmitters, from 1 to 5, in a 15 D2D transmitter environment. The basic parameter is the same as above. Fig. 5.9 shows that, as the number of malicious D2D transmitters increases, the utility drops dramatically. When the number of malicious D2D transmitters reaches to one third of the total number of D2D transmitters in the network, the utility decreases to about half of the utility in the network with 1 malicious nodes. From this figure, we can see that the network is deeply affected by the number of malicious D2D transmitters.

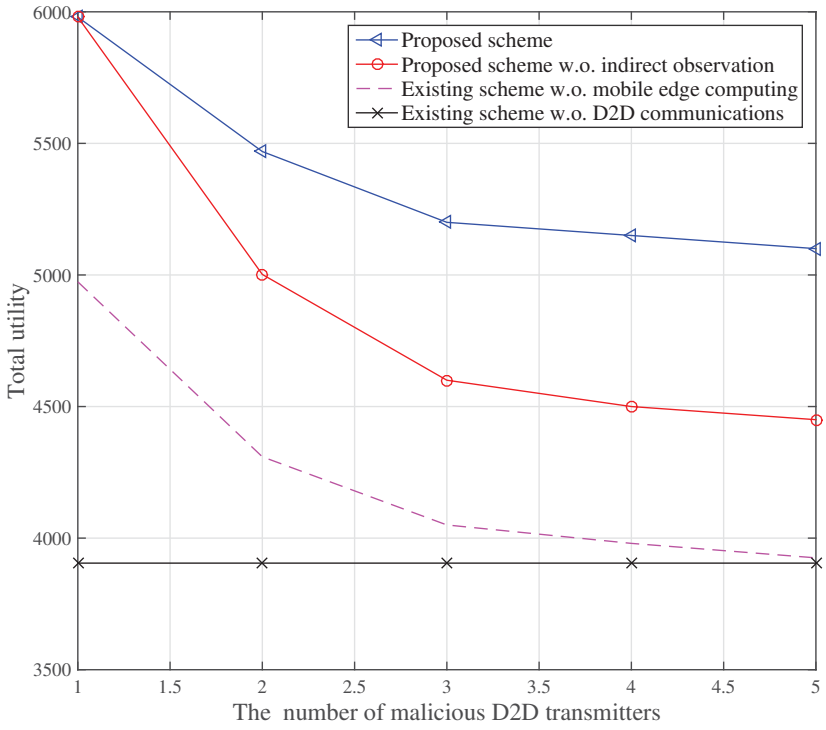

Figure 5.9: The total utility v.s. the number of malicious D2D transmitters.

5.6 Chapter Summary

MSNs have become one of the most important networking paradigms in future wireless mobile networks. Recent advances of wireless mobile networks can have significant impacts on the performance of MSNs. In this chapter, we studied recent advances in mobile edge computing, in-network caching and D2D communications in MSNs. In addition, we considered the knowledge of social relationships in these new paradigms to improve the security and efficiency of MSNs. Specifically, we presented a social trust scheme with both direct observation using Bayesian inference and indirect observation using the Dempster-Shafer theory. We further proposed a deep Q-learning

approach to study this complicated system. Extensive simulation results were presented to show the effectiveness of the proposed scheme.

Chapter 6

Conclusions and Future Work

In this dissertation, we have thoroughly studied the integrated networking, caching and computing wireless mobile networks via a deep reinforcement learning approach. In this chapter, we conclude the accomplished works and present some possible research directions in the future.

6.1 Summary

This dissertation exploits the current advances of machine learning and deep reinforcement learning algorithm, to tackle the network optimization and resource allocation issues in the integrated wireless networks with communications, caching and computing. The main contributions are as follows.

- More realistic time-varying wireless channels are considered, where the time-varying wireless channels based on interference alignment are modelled as finite-state Markov channels. A deep-reinforcement-learning-based resource allocation scheme is proposed for cache-enabled opportunistic interference alignment networks. Within the proposed scheme, the formula of network capacity is given with the considerations of both perfect CSI and imperfect CSI. Aiming at maximizing the network capacity, the resource allocation problem is formulated as a

reinforcement learning process, and the deep Q learning algorithm is exploited to obtain the optimal strategy. The performances of the proposed scheme with different SNR, different transition probabilities and imperfect CSI are presented via simulations. The simulation results show that the proposed scheme is applicable to complicated time-varying wireless communication scenarios. It can effectly reduce the backhaul link usage and allow more users to access interference alignment networks, and thus the network sum rate and energy efficiency are improved.

- A software-defined virtulized framework for connected vehicles is designed, which integrates communication, caching and mobile edge computing. A deep reinforcement learning-based resource allocation scheme is proposed for the connected vehicles with the integration of communication, caching and computing. The dynamic change processes of the communication, caching, and computing resources are modeled as Markov chains, respectively. Aiming at maximizing the network operators total utility, the joint resource allocation problem is formulated as a reinforcement learning process, and the deep Q learning algorithm is used to pursue the optimal solution. Without any assumptions about the objective funtions or any low-complexity preprocessing, the proposed scheme can directly solve the resource allocation problems with large-scale state space. Simulation results verify that the proposed scheme can converge at a fast speed, improve the network operators total utilities, and possess the ability of resisting perturbation at a certain level.

- A resource allocation scheme is proposed for future social networks based on the social trust model and a deep reinforcement learning approach. The social trust model utilizes uncertain reasoning to derive the trust values of D2D users, including the trust from direct observations based on Bayesian inference and

the trust from indirect observations based on Dempster-Shafer theory. The effects of cache sizes and content types on backhaul link usage are analyzed, and the effectiveness of the proposed social trust scheme is verified via simulations that it can successfully track the variations of D2D users trust values. The simulation results also show that the proposed scheme can reduce the effects of the malicious users on the network total utility.

6.2 Future Work

On the basis of the accomplished research works, a number of new interesting research problems rise.

- In the future work, if real network data can be obtained, Markov chain monte carlo method can be exploited to achieve the real Markov chain and its transitional probability matrix. In addition, for some specific scenarios, the parameters in the Markov decision process may be changed. Therefore, in the next step, we consider adaptive reinforcement learning, which can adaptively restart to interact with the environment, and learn to make decisions based on the new environment.

- In this dissertation, deep Q learning algorithm is utilized, where falls into the category of value-based reinforcement learning. However, the most recent research shows that policy-based reinforcement learning can achieve better performance of convergence. Therefore, in the future study, we will consider to update the deep reinforcement learning algorithms, for example, to exploit the policy-based A3C (Asynchronous Advantage Actor-Critic) algorithm to solve the more complicated network problems with larger state space.

- In this dissertation, the communication resource allocation mainly refers to how

to allocate the basic network architecture. In our following work, we will further investigate other physical layer resources, such as wireless spectrum, power, etc. Additionally, in Chapter 3 and Chapter 4, the total utility is optimized with given charging prices. In the future, we will jointly consider more factors, such as charging prices, spectrum, power, network architecture, caching, computing status, and etc., which will definitely improve the quality of networks further.

List of References

[1] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," *Emerging Artificial Intelligence Applications in Computer Engineering*, vol. 160, pp. 3–24, 2007.

[2] J. Friedman, T. Hastie, and R. Tibshirani, *The Elements of Statistical Learning*, vol. 1. Springer Series in Statistics New York, 2001.

[3] E. Alpaydin, *Introduction to Machine Learning*. MIT Press, 2014.

[4] R. S. Sutton, "Introduction: The challenge of reinforcement learning," in *Reinforcement Learning*, pp. 1–3, Springer, 1992.

[5] Z. Wang, N. de Freitas, and M. Lanctot, "Dueling network architectures for deep reinforcement learning," *arXiv preprint arXiv:1511.06581*, 2015.

[6] M. Hausknecht and P. Stone, "Deep reinforcement learning in parameterized action space," *arXiv preprint arXiv:1511.04143*, 2015.

[7] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[8] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[9] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep Q-learning with model-based acceleration," *arXiv preprint arXiv:1603.00748*, 2016.

[10] G. Paschos, E. Bastug, I. Land, G. Caire, and M. Debbah, "Wireless caching: technical misconceptions and business barriers," *IEEE Commun. Mag.*, vol. 54, pp. 16–22, Aug. 2016.

[11] C. Liang, F. R. Yu, and X. Zhang, "Information-centric network function virtualization over 5G mobile wireless networks," *IEEE Network*, vol. 29, pp. 68–74, May 2015.

[12] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. C. M. Leung, "Cache in the air: exploiting content caching and delivery techniques for 5G systems," *IEEE Commun. Mag.*, vol. 52, pp. 131–139, Feb. 2014.

[13] C. Fang, F. R. Yu, T. Huang, J. Liu, and Y. Liu, "A survey of green information-centric networking: Research issues and challenges," *IEEE Commun. Surv. Tut.*, vol. 17, pp. 1455–1472, Thirdquarter 2015.

[14] D. Liu, B. Chen, C. Yang, and A. F. Molisch, "Caching at the wireless edge: design aspects, challenges, and future directions," *IEEE Commun. Mag.*, vol. 54, no. 9, pp. 22–28, 2016.

[15] Y. Wei, F. R. Yu, and M. Song, "Distributed optimal relay selection in wireless cooperative networks with finite-state Markov channels," *IEEE Trans. Veh. Tech.*, vol. 59, no. 5, pp. 2149–2158, 2010.

[16] H. S. Wang and P.-C. Chang, "On verifying the first-order Markovian assumption for a Rayleigh fading channel model," *IEEE Trans. Veh. Tech.*, vol. 45, no. 2, pp. 353–357, 1996.

[17] C. Luo, F. R. Yu, H. Ji, and V. C. M. Leung, "Cross-layer design for TCP performance improvement in cognitive radio networks," *IEEE Trans. Veh. Tech.*, vol. 59, pp. 2485–2495, June 2010.

[18] Y. He, C. Liang, F. R. Yu, N. Zhao, and H. Yin, "Optimization of cache-enabled opportunistic interference alignment wireless networks: A big data deep reinforcement learning approach," in *Proc. IEEE ICC'17*, (Paris, France), June 2017.

[19] Y. He, F. R. Yu, N. Zhao, V. C. M. Leung, and H. Yin, "Software-defined networks with mobile edge computing and caching for smart cities: A big data deep reinforcement learning approach," *IEEE Commun. Mag.*, vol. 55, Dec. 2017.

[20] Y. He, Z. Zhang, F. R. Yu, N. Zhao, H. Yin, V. C. M. Leung, and Y. Zhang, "Deep reinforcement learning-based optimization for cache-enabled opportunistic interference alignment wireless networks," *IEEE Trans. Veh. Tech.*, 2017. online.

[21] N. Vastardis and K. Yang, "Mobile social networks: Architectures, social properties, and key research challenges," *IEEE Commun. Surv. Tut.*, vol. 15, pp. 1355–1371, Third Quarter 2013.

[22] Z. Su and Q. Xu, "Content distribution over content centric mobile social networks in 5G," *IEEE Commun. Mag.*, vol. 53, pp. 66–72, Jun. 2015.

[23] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, 1994.

[24] S. Haykin and N. Network, "A comprehensive foundation," *Neural Networks*, vol. 2, no. 2004, p. 41, 2004.

[25] K. Lee, D. Booth, and P. Alam, "A comparison of supervised and unsupervised neural networks in predicting bankruptcy of Korean firms," *Expert Syst. Appl.*, vol. 29, no. 1, pp. 1–16, 2005.

[26] S. Timotheou, "The random neural network: A survey," *The Computer J.*, vol. 53, pp. 251–267, Mar. 2010.

[27] S. Basterrech and G. Rubino, "A tutorial about random neural networks in supervised learning," *arXiv preprint arXiv:1609.04846*, 2016.

[28] H. Bakirciouglu and T. Koccak, "Survey of random neural network applications," *Eur. J. Oper. Res.*, vol. 126, no. 2, pp. 319–330, 2000.

[29] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[30] J. Baker, "Artificial neural networks and deep learning," Feb. 2015.

[31] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[32] G. Pandey and A. Dukkipati, "Learning by stretching deep networks," in *Proc. ICML'14*, pp. 1719–1727, 2014.

[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. NIPS'12*, pp. 1097–1105, 2012.

[34] C. Li, Y. Wu, X. Yuan, Z. Sun, W. Wang, X. Li, and L. Gong, "Detection and defense of DDoS attack-based on deep learning in OpenFlow-based SDN," *Int. J. Commun. Syst.*, 2018.

[35] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Interspeech'10*, 2010.

[36] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Interspeech'14*, 2014.

[37] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[38] X. Li and X. Wu, "Constructing long short-term memory based deep recurrent neural networks for large vocabulary speech recognition," in *Proc. IEEE ICASSP'15, Brisbane, QLD, Australia*, pp. 4520–4524, April 2015.

[39] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *IEEE Trans. Pattern Anal.*, vol. 24, no. 7, pp. 881–892, Jul. 2002.

[40] T. Kohonen, "The self-organizing map," *Neurocomputing*, vol. 21, no. 1-3, pp. 1–6, 1998.

[41] M. M. Van Hulle, "Self-organizing maps," in *Handbook of Natural Computing*, pp. 585–622, Springer, 2012.

[42] H. Y. Ong, K. Chavez, and A. Hong, "Distributed deep Q-learning," *arXiv preprint arXiv:1508.04186*, 2015.

[43] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *J. Mach. Learn. Res.*, vol. 17, no. 39, pp. 1–40, 2016.

[44] M. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *Proc. ICML'11*, pp. 465–472, 2011.

[45] J. Nie and S. Haykin, "A Q-learning-based dynamic channel assignment technique for mobile communication systems," *IEEE Trans. Veh. Tech.*, vol. 48, no. 5, pp. 1676–1687, 1999.

[46] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[47] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," *CoRR, abs/1509.06461*, 2015.

[48] C. Wang, Y. He, F. R. Yu, Q. Chen, and L. Tang, "Integration of networking, caching, and computing in wireless systems: A survey, some research issues, and challenges," *IEEE Commun. Surv. Tut.*, vol. 20, pp. 7–38, Firstquarter 2018.

[49] M. Richart, J. Baliosian, J. Serrat, and J. Gorricho, "Resource slicing in virtual wireless networks: A survey," *IEEE Trans. Netw. Serv.*, vol. 13, pp. 462–476, Sep. 2016.

[50] N. Bizanis and F. A. Kuipers, "SDN and virtualization solutions for the Internet of things: A survey," *IEEE Access*, vol. 4, pp. 5591–5606, 2016.

[51] C. Liang and F. R. Yu, "Wireless network virtualization: A survey, some research issues and challenges," *IEEE Commun. Surv. Tut.*, vol. 17, pp. 358–380, Firstquarter 2015.

[52] Y. Zhou and W. Yu, "Optimized backhaul compression for uplink cloud radio access network," *IEEE J. Sel. Area. Commun.*, vol. 32, pp. 1295–1307, June 2014.

[53] C. Fang, H. Yao, Z. Wang, W. Wu, X. Jin, and F. R. Yu, "A survey of mobile information-centric networking: Research issues and challenges," *IEEE Commun. Surv. Tut.*, vol. 20, pp. 2353–2371, thirdquarter 2018.

[54] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surv. Tut.*, vol. 19, pp. 2322–2358, Fourthquarter 2017.

[55] V. R. Cadambe and S. A. Jafar, "Interference alignment and degrees of freedom of the K-user interference channel," *IEEE Trans. Inform. Theory*, vol. 54, no. 8, pp. 3425–3441, Aug. 2008.

[56] C. Suh and D. Tse, "Interference alignment for cellular networks," in *Proc. 46th Annual Allerton Conf. on Commun., Control, and Computing,*, pp. 1037–1044, Monticello, IL, Sep. 2008.

[57] S. M. Perlaza, N. Fawaz, S. Lasaulce, and M. Debbah, "From spectrum pooling to space pooling: opportunistic interference alignment in MIMO cognitive networks," *IEEE Trans. Signal Proc.*, vol. 58, no. 7, pp. 3728–3741, 2010.

[58] X. Li, N. Zhao, Y. Sun, and F. R. Yu, "Interference alignment based on antenna selection with imperfect channel state information in cognitive radio networks," *IEEE Trans. Veh. Tech.*, vol. 65, no. 7, pp. 5497–5511, July 2016.

[59] B. C. Jung and W.-Y. Shin, "Opportunistic interference alignment for interference-limited cellular TDD uplink," *IEEE Commun. Lett.*, vol. 15, no. 2, pp. 148–150, 2011.

[60] Y. He, H. Yin, and N. Zhao, "Multiuser-diversity-based interference alignment in cognitive radio networks," *AEU-Int. J. Electron. C*, vol. 70, no. 5, pp. 617–628, 2016.

[61] M. Deghel, E. Baştuğ, M. Assaad, and M. Debbah, "On the benefits of edge caching for MIMO interference alignment," in *Proc. IEEE SPAWC'15*, pp. 655–659, 2015.

[62] M. A. Maddah-Ali and U. Niesen, "Cache-aided interference channels," in *Proc. IEEE ISIT'15*, pp. 809–813, 2015.

[63] O. E. Ayach, S. W. Peters, and R. W. Heath, "The practical challenges of interference alignment," *IEEE Wirel. Commun.*, vol. 20, no. 1, pp. 35–42, Feb. 2013.

[64] J. Yang, A. K. Khandani, and N. Tin, "Statistical decision making in adaptive modulation and coding for 3G wireless systems," *IEEE Trans. Veh. Tech.*, vol. 54, no. 6, pp. 2066–2073, 2005.

[65] A. Z. Ghanavati, U. Pareek, S. Muhaidat, and D. Lee, "On the performance of imperfect channel estimation for vehicular ad-hoc networks," in *Proc. IEEE VTC'10-Fall*, pp. 1–5, Sept. 2010.

[66] G. Taricco and E. Biglieri, "Space-time decoding with imperfect channel estimation," *IEEE Trans. Wireless Commun.*, vol. 4, no. 4, pp. 1874–1888, Jul. 2005.

[67] S. Wen and F. Richard Yu, "Predictive control for energy efficiency in wireless cellular networks," in *Proc. IEEE VTC'12S*, Yokohama, Japan, May 2012.

[68] Y. He, F. R. Yu, N. Zhao, H. Yin, H. Yao, and R. C. Qiu, "Big data analytics in mobile cellular networks," *IEEE Access*, vol. 4, pp. 1985–1996, Mar. 2016.

[69] C. Pimentel, T. H. Falk, and L. Lisbôa, "Finite-state Markov modeling of correlated Rician-fading channels," *IEEE Trans. Veh. Tech.*, vol. 53, no. 5, pp. 1491–1501, 2004.

[70] K. Gomadam, V. R. Cadambe, and S. A. Jafar, "A distributed numerical approach to interference alignment and applications to wireless interference networks," *IEEE Trans. Inform. Theory*, vol. 57, no. 6, pp. 3309–3322, 2011.

[71] A. Tatar, M. D. de Amorim, S. Fdida, and P. Antoniadis, "A survey on predicting the popularity of web content," *Springer J. Internet Services and Applications*, vol. 5, no. 1, p. 8, 2014.

[72] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge, U.K.: Cambridge Univ. Press, 2005.

[73] R. H. Clarke, "A statistical theory of mobile radio reception," *Bell Syst. Tech. J.*, vol. 47, no. 6, pp. 957–1000, Jul.-Aug. 1968.

[74] M. Abadi, A. Agarwal, *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous systems," *arXiv:1603.04467*, Nov. 2015.

[75] "Tensorflow.org." https://www.tensorflow.org/.

[76] N. Zhao, F. R. Yu, H. Sun, and M. Li, "Adaptive power allocation schemes for spectrum sharing in interference alignment (IA)-based cognitive radio networks," *IEEE Trans. Veh. Tech.*, vol. 65, no. 5, pp. 3700–3714, May 2016.

[77] F. Cheng, Y. Yu, Z. Zhao, N. Zhao, Y. Chen, and H. Lin, "Power allocation for cache-aided small-cell networks with limited backhaul," *IEEE Access*, vol. 5, pp. 1272–1283, 2017.

[78] C. M. Yetis, T. Gou, S. A. Jafar, and A. H. Kayran, "On feasibility of interference alignment in MIMO interference networks," *IEEE Trans. Signal Proc.*, vol. 58, no. 9, pp. 4771–4782, 2010.

[79] H. S. Wang and N. Moayeri, "Finite-state Markov channel-a useful model for radio communication channels," *IEEE Trans. Veh. Tech.*, vol. 44, pp. 163–171, Feb. 1995.

[80] L. Cui, F. R. Yu, and Q. Yan, "When big data meets software-defined networking (SDN): SDN for big data and big data for SDN," *IEEE Network*, vol. 30, pp. 58–65, Jan. 2016.

[81] Y. Cai, F. R. Yu, C. Liang, B. Sun, and Q. Yan, "Software defined device-to-device (D2D) communications in virtual wireless networks with imperfect network state information (NSI)," *IEEE Trans. Veh. Tech.*, no. 9, pp. 7349–7360, Sept. 2016.

[82] K. Liu, J. K. Y. Ng, V. C. S. Lee, S. H. Son, and I. Stojmenovic, "Cooperative data scheduling in hybrid vehicular ad hoc networks: VANET as a software

defined network," *IEEE/ACM Trans. Networking*, vol. 24, pp. 1759–1773, June 2016.

[83] Q. Zheng, K. Zheng, H. Zhang, and V. C. M. Leung, "Delay-optimal virtualized radio resource scheduling in software-defined vehicular networks via stochastic learning," *IEEE Trans. Veh. Tech.*, vol. 65, pp. 7857–7867, Oct. 2016.

[84] R. dos Reis Fontes, C. Campolo, C. E. Rothenberg, and A. Molinaro, "From theory to experimental evaluation: Resource management in software-defined vehicular networks," *IEEE Access*, vol. PP, no. 99, pp. 1–1, 2017.

[85] K. Liu, J. K. Y. Ng, J. Wang, V. C. S. Lee, W. Wu, and S. H. Son, "Network-coding-assisted data dissemination via cooperative vehicle-to-vehicle/-infrastructure communications," *IEEE Trans. Intell. Transp. Sys.*, vol. 17, pp. 1509–1520, June 2016.

[86] M. Amadeo, C. Campolo, and A. Molinaro, "Information-centric networking for connected vehicles: a survey and future perspectives," *IEEE Commun. Mag.*, vol. 54, pp. 98–104, Feb. 2016.

[87] M. Armbrust, A. Fox, R. Griffith, and A. D. Joseph, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, Apr. 2010.

[88] H. Zhang, Q. Zhang, and X. Du, "Toward vehicle-assisted cloud computing for smartphones," *IEEE Trans. Veh. Tech.*, vol. 64, pp. 5610–5618, Dec. 2015.

[89] ETSI, "Mobile-edge computing introductory technical white paper," *ETSI White Paper*, Sep. 2014.

[90] C. Liang, Y. He, F. R. Yu, and N. Zhao, "Energy-efficient resource allocation in software-defined mobile networks with mobile edge computing and caching," in *IEEE Infocom'17 Workshops*, May 2017.

[91] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular fog computing: A viewpoint of vehicles as the infrastructures," *IEEE Trans. Veh. Tech.*, vol. 65, pp. 3860–3873, Jun. 2016.

[92] N. Kumar, S. Zeadally, and J. J. P. C. Rodrigues, "Vehicular delay-tolerant networks for smart grid data management using mobile edge computing," *IEEE Commun. Mag.*, vol. 54, pp. 60–66, Oct. 2016.

[93] C. Campoloa, A. Molinaroa, and R. Scopigno, "From today's vanets to tomorrow's planning and the bets for the day after," *Elsevier Veh. Commun.*, vol. 2, no. 3, pp. 158–171, 2015.

[94] K. Abboud, H. A. Omar, and W. Zhuang, "Interworking of DSRC and cellular network technologies for V2X communications: A survey," *IEEE Trans. Veh. Tech.*, vol. 65, pp. 9457–9470, Dec. 2016.

[95] L. Ma, F. Yu, V. C. M. Leung, and T. Randhawa, "A new method to support UMTS/WLAN vertical handover using SCTP," *IEEE Wireless Commun.*, vol. 11, pp. 44–51, Aug. 2004.

[96] F. Yu and V. Krishnamurthy, "Optimal joint session admission control in integrated WLAN and CDMA cellular networks with vertical handoff," *IEEE Trans. Mobile Computing*, vol. 6, pp. 126–139, Jan. 2007.

[97] L. Zhang, A. Afanasyev, F. Burke, V. Jacobson, K.C. Claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named data networking," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 66–73, Jul. 2014.

[98] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, "A survey of information-centric networking research," *IEEE Commun. Surv. Tut.*, vol. 16, pp. 1024–1049, Second Quarter 2014.

[99] Y. Wei, F. R. Yu, and M. Song, "Distributed optimal relay selection in wireless cooperative networks with finite-state Markov channels," *IEEE Trans. Veh. Tech.*, vol. 59, pp. 2149 –2158, June 2010.

[100] H. Wang, F. R. Yu, L. Zhu, T. Tang, and B. Ning, "Finite-state Markov modeling for wireless channels in tunnel communication-based train control (CBTC) systems," *IEEE Trans. Intell. Transp. Sys.*, vol. 15, no. 3, pp. 1083–1090, June 2014.

[101] H. Gomaa, G. G. Messier, C. Williamson, and R. Davies, "Estimating instantaneous cache hit ratio using markov chain analysis," *IEEE/ACM Transactions on Networking*, vol. 21, no. 5, pp. 1472–1483, 2013.

[102] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *Proc. IEEE INFOCOM'99*, vol. 1, pp. 126–134, 1999.

[103] M. Patel, B. Naughton, C. Chan, N. Sprecher, S. Abeta, A. Neal, *et al.*, "Mobile-edge computing introductory technical white paper," *White Paper, Mobile-edge Computing (MEC) industry initiative*, Sep. 2014.

[104] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Trans. Wirel. Commun.*, vol. 16, pp. 4924–4938, Aug. 2017.

[105] Y. Meng, C. Jiang, H. H. Chen, and Y. Ren, "Cooperative device-to-device communications: Social networking perspectives," *IEEE Network*, vol. 31, pp. 38–44, May 2017.

[106] K. Wang, F. R. Yu, and H. Li, "Information-centric virtualized cellular networks with device-to-device (D2D) communications," *IEEE Trans. Veh. Tech.*, vol. 65, pp. 9319 – 9329, Nov. 2016.

[107] Y. Zhang, E. Pan, L. Song, W. Saad, Z. Dawy, and Z. Han, "Social network aware device-to-device communication in wireless networks," *IEEE Trans. Wirel. Commun.*, vol. 14, pp. 177–190, Jan. 2015.

[108] P. Hui, J. Crowcroft, and E. Yoneki, "Bubble rap: Social-based forwarding in delay-tolerant networks," *IEEE Trans. Mobile Comput.*, vol. 10, pp. 1576–1589, Nov. 2011.

[109] E. Bulut and B. K. Szymanski, "Friendship based routing in delay tolerant mobile social networks," in *Proc. IEEE GLOBECOM'10*, (Miami, FL, USA), pp. 1–5, Dec. 2010.

[110] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, vol. 5, pp. 6757–6779, Mar. 2017.

[111] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, and Y. Zhang, "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks," *IEEE Access*, vol. 4, pp. 5896–5907, Aug. 2016.

[112] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proc. IEEE ISIT'16*, (Barcelona, Spain), pp. 1451–1455, Jul. 2016.

[113] 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, pp. 89–103, Jan. 2015.

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

[115] X. Chen, J. Wu, Y. Cai, H. Zhang, and T. Chen, "Energy-efficiency oriented traffic offloading in wireless networks: A brief survey and a learning approach for heterogeneous cellular networks," *IEEE J. Sel. Areas Commun.*, vol. 33, pp. 627–640, Jan. 2015.

[116] J. Gu, W. Wang, A. Huang, and H. Shan, "Proactive storage at caching-enable base stations in cellular networks," in *Proc. IEEE PIMRC'13*, (London, UK), pp. 1543–1547, Sept. 2013.

[117] B. Bai, L. Wang, Z. Han, W. Chen, and T. Svensson, "Caching based socially-aware D2D communications in wireless content delivery networks: a hypergraph framework," *IEEE Wirel. Commun.*, vol. 23, pp. 74–81, Aug. 2016.

[118] A. Sengupta, S. Amuru, R. Tandon, R. M. Buehrer, and T. C. Clancy, "Learning distributed caching strategies in small cell networks," in *Proc. ISWCS'14*, pp. 917–921, Aug. 2014.

[119] Y. Guan, Y. Xiao, H. Feng, C.-C. Shen, and L. J. Cimini, "Mobicacher: Mobility-aware content caching in small-cell networks," in *Proc. IEEE GLOBECOM'14*, (Austin, TX, USA), pp. 4537–4542, Feb. 2014.

[120] Y. He, C. Liang, F. R. Yu, N. Zhao, and H. Yin, "Optimization of cache-enabled opportunistic interference alignment wireless networks: A big data deep reinforcement learning approach," in *Proc. IEEE ICC'17*, (Paris, France), pp. 1–6, May 2017.

[121] Y. He, F. R. Yu, N. Zhao, V. Leung, and H. Yin, "Software-defined networks with mobile edge computing and caching for smart cities: A big data deep reinforcement learning approach," *IEEE Commun. Mag.*, Sept. 2017.

[122] Y. Zhou, F. R. Yu, J. Chen, and Y. Kuo, "Resource allocation for information-centric virtualized heterogeneous networks with in-network caching and mobile edge computing," *IEEE Trans. Veh. Tech.*, vol. 66, pp. 11339–11351, Dec. 2017.

[123] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Trans. Wirel. Commun.*, vol. 16, pp. 4924–4938, Aug. 2017.

[124] Y. He, N. Zhao, and H. Yin, "Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach," *IEEE Trans. Veh. Tech.*, vol. 67, pp. 44–55, Jan. 2018.

[125] Y. He, F. R. Yu, N. Zhao, V. C. Leung, and H. Yin, "Software-defined networks with mobile edge computing and caching for smart cities: A big data deep reinforcement learning approach," *IEEE Commun. Mag.*, vol. 55, pp. 31–37, Dec. 2017.

[126] Z. Su and Q. Xu, "Security-aware resource allocation for mobile social big data: A matching-coalitional game solution," *IEEE Trans. on Big Data*, vol. PP, no. 99, pp. 1–1, 2017.

[127] A. Asadi, Q. Wang, and V. Mancuso, "A survey on device-to-device communication in cellular networks," *IEEE Commun. Surv. Tut.*, vol. 16, pp. 1801–1819, Fourth Quarter 2014.

[128] C. Yang, J. Li, P. Semasinghe, E. Hossain, S. M. Perlaza, and Z. Han, "Distributed interference and energy-aware power control for ultra-dense D2D networks: A mean field game," *IEEE Trans. Wirel. Commun.*, vol. 16, pp. 1205–1217, Feb. 2017.

[129] K. Hamedani, L. Liu, R. Atat, J. Wu, and Y. Yi, "Reservoir computing meets smart grids: attack detection using delayed feedback networks," *IEEE Trans. Ind. Inf.*, vol. 14, pp. 734–743, Feb. 2018.

[130] R. Atat, L. Liu, H. Chen, J. Wu, H. Li, and Y. Yi, "Enabling cyber-physical communication in 5G cellular networks: challenges, spatial spectrum sensing, and cyber-security," *IET Cyber-Phys. Syst.Theory Appl.*, vol. 2, pp. 49–54, Apr. 2017.

[131] I. R. Chen, F. Bao, and J. Guo, "Trust-based service management for social Internet of things systems," *IEEE Trans. Dependable and Secure Comput.*, vol. 13, pp. 684–696, Nov. 2016.

[132] F. Parzysz, M. Di Renzo, and C. Verikoukis, "Power-availability-aware cell association for energy-harvesting small-cell base stations," *IEEE Trans. Wirel. Commun.*, vol. 16, no. 4, pp. 2409–2422, Apr. 2017.

[133] T. M. Chen and V. Venkataramanan, "Dempster-Shafer theory for intrusion detection in ad hoc networks," *IEEE Internet Comput.*, vol. 9, no. 6, pp. 35–41, Nov. 2005.

[134] R. Changiz, H. Halabian, F. R. Yu, I. Lambadaris, H. Tang, and C. M. Peter, "Trust establishment in cooperative wireless networks," in *Proc. IEEE MilCom'10*, (San Jose, CA), pp. 1074–1079, Nov. 2010.

[135] C. Zouridaki, B. L. Mark, M. Hejmo, and R. K. Thomas, "Hermes: A quantitative trust establishment framework for reliable data packet delivery in MANETs," *Journal of Computer Security*, vol. 15, pp. 3–38, Jan. 2007.

[136] S. Bu, F. R. Yu, P. Liu, P. Manson, and H. Tang, "Distributed combined authentication and intrusion detection with data fusion in high-security mobile ad hoc networks," *IEEE Trans. Veh. Tech.*, vol. 60, pp. 1025 –1036, Mar. 2011.

[137] Y. Zhao, W. Song, and Z. Han, "Social-aware data dissemination via device-to-device communications: Fusing social and mobile networks with incentive constraints," *IEEE Trans. Serv. Comput.*, vol. PP, no. 99, pp. 1–1, 2017.

[138] B. Kaufman and B. Aazhang, "Cellular networks with an overlaid device to device network," in *Proc. ACSSC'08*, (Pacific Grove, CA), pp. 1537–1541, Oct. 2008.

[139] D. H. Lee, K. W. Choi, W. S. Jeon, and D. G. Jeong, "Two-stage semi-distributed resource management for device-to-device communication in cellular networks," *IEEE Trans. Wirel. Commun.*, vol. 13, pp. 1908–1920, Apr. 2014.

[140] P. Blasco and D. Gunduz, "Learning-based optimization of cache content in a small cell base station," in *Proc. IEEE ICC'14*, (Sydney, Australia), pp. 1897–1903, Jun. 2014.