Solving Channel Allocation by Reinforcement Learning in Cognitive Enabled Vehicular Ad Hoc Networks

by

Yunfan Su
Bachelor of Science in Telecommunication Engineering, Nanjing University of Posts and Telecommunications, 2017

A thesis
presented to Ryerson University
in partial fulfillment of
the requirements for the degree of
Master of Applied Science
in the program of
Electrical and Computer Engineering

Toronto, Ontario, Canada, 2019

Author's Declaration For Electronic Submission Of A Thesis

I hereby declare that I am the sole author of this dissertation. This is a true copy of the dissertation, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this dissertation to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this dissertation by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my dissertation may be made electronically available to the public.

SOLVING CHANNEL ALLOCATION BY REINFORCEMENT LEARNING IN COGNITIVE ENABLED VEHICULAR AD HOC NETWORKS

Yunfan Su

Master of Applied Science

Electrical and Computer Engineering

Ryerson University, 2019

Abstract

Vehicular ad hoc network (VANET) is a promising technique that improves traffic safety and transportation efficiency and provides a comfortable driving experience. However, due to the rapid growth of applications that demand channel resources, efficient channel allocation schemes are required to utilize the performance of the vehicular networks. In this thesis, two Reinforcement learning (RL)-based channel allocation methods are proposed for a cognitive enabled VANET environment to maximize a long-term average system reward.

First, we present a model-based dynamic programming method, which requires the calculations of the transition probabilities and time intervals between decision epochs. After obtaining the transition probabilities and time intervals, a relative value iteration (RVI) algorithm is used to find the asymptotically optimal policy. Then, we propose a model-free reinforcement learning method, in which we employ an agent to interact with the environment iteratively and learn from the feedback to approximate the optimal policy. Simulation results show that our reinforcement learning method can acquire a similar performance to that of the dynamic programming while both outperform the greedy method.

Acknowledgments

The author graciously thanks his supervisor and mentor, Professor Cungang Yang, for his inspiration, technical and personal insight and unwavering support throughout this work.

The author also thanks Professor Lian Zhao for her advice and assistance in keeping the progress on schedule. Her enthusiasm, vision and experiences have been the greatest source and motivation to make possible the accomplishments herein.

I would like to acknowledge the members of Professor Zhao's Research Group at Ryerson University, Jie Gao, Frank Wang, Wei Yu and Xiaoqiang He for their helpful discussions, constructive suggestions, and invaluable technical inputs. Special acknowledgments go to former members of the research group, Dr. Qizhen Li, for his help in applying Reinforcement learning, and Mushu Li for her assistance in completing the system model.

I would also like to extend my thanks to Xin Nuo Bai for her efforts in helping me review the thesis paper and prepare for the thesis defense.

My MASc. program was funded by Graduate Fellowships endowed by Ryerson University. I remain grateful for the support.

Finally, I would like to thank my family for all the love and support they have provided through the years.

Contents

List of Tables

List of Figures

List of Algorithms

Acronyms

CR Cognitive Radio. 3

CR-VANET Cognitive enabled Vehicular ad hoc Network. 3

CTMDP Continuous-Time Markov Decision Process. 5

LTE Long-Term Evolution. 2

MANETs Mobile ad hoc Networks. 2

MDP Markov Decision Process. 9

OBUs On-Board Units. 2

PDEs Partial Differential Equations. 9

PUs Primary Users. 3

QoS Quality-of-Service. 3

RL Reinforcement Learning. 5

RSUs Roadside Units. 2

RVI Relative Value Iteration. 5

SMART Semi-Markov Average Reward Technique. 35

SMDP Semi-Markov Decision Process. 5

SUs Secondary Users. 3

V2I Vehicle-to-Infrastructure. 3

V2V Vehicle-to-Vehicle. 3

VANET Vehicular ad hoc Network. 2

Chapter 1

Introduction

1.1 Background

1.1.1 Cellular Network

Over the past few years, traditional cellular networks have been widely used for transmission of voice, data, and other types of content. A cellular network is a communication network where the last link is wireless. The network is distributed over land areas called "cells", each served by at least one fixed-location transceiver, but more normally three cell sites or base transceiver stations.

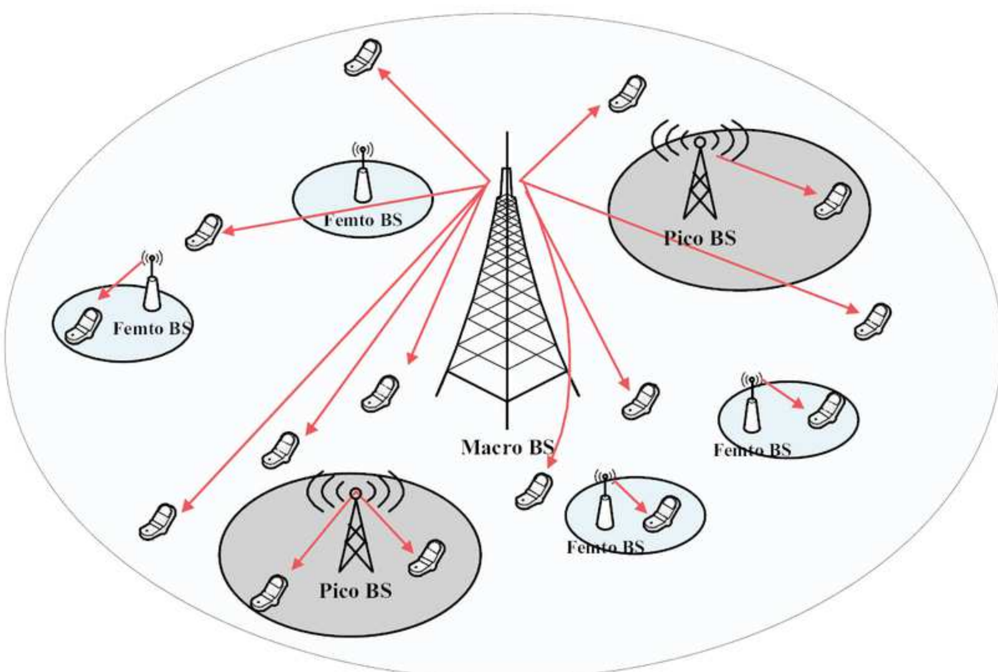

Figure 1.1: Structure of a three-tier heterogeneous cellular network.

These base stations provide the cell with the network coverage which can be used for transmission of all kinds of content. Figure 1.1 is the illustration of the structure of a three-tier heterogeneous cellular network. A cell typically uses a different set of frequencies from neighboring cells, to avoid interference and provide guaranteed service quality within each cell. Service based on a cellular network such as long-term evolution (LTE)/LTE Advanced network, benefiting from its large coverage area and centralized resource allocation, is capable of maintaining a long-distance yet stable transmission. Based on the location of the base station, a cellular network can serve both background users like all the residents in a residential community and vehicle users on the road.

1.1.2 Vehicular ad hoc Network

Vehicular ad hoc network (VANET), first mentioned and introduced in [1] under "car-to-car ad-hoc mobile communication and networking" applications, where network can be formed and information can be relayed among cars, is created by applying the principles of mobile ad hoc networks (MANETs) – the spontaneous creation of a wireless network of mobile devices – to the domain of vehicles [2]. We illustrate VANET in Figure 1.2. There are Roadside Units (RSUs) and On-Board Units (OBUs) in the VANET system [3], [4], [5]. The RSU is a wave device usually fixed along the

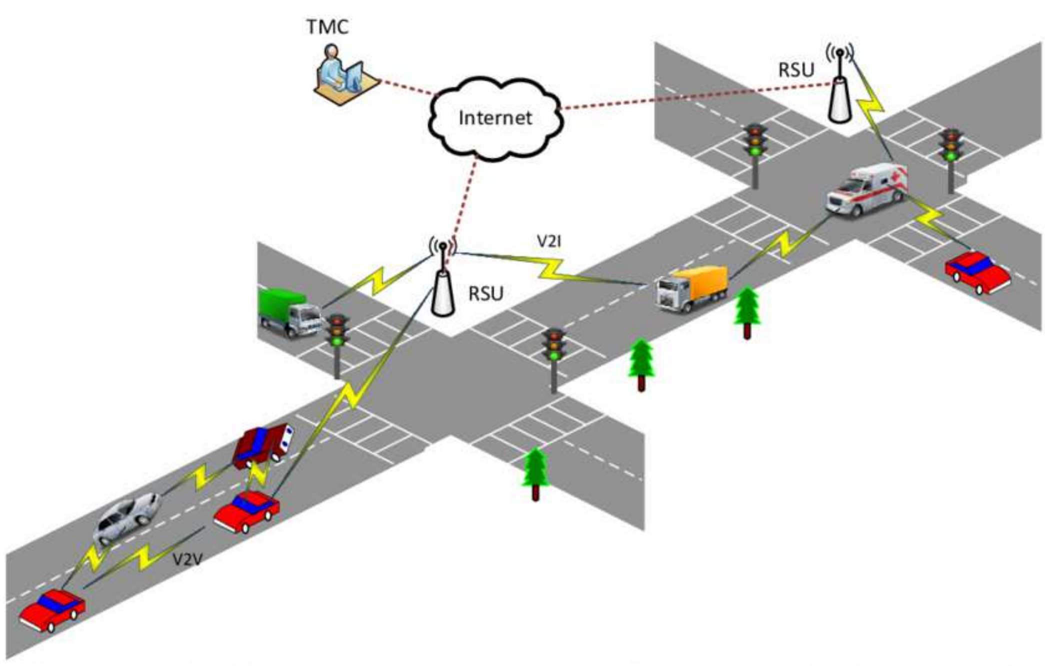

Figure 1.2: Illustration of Vehicular ad hoc network.

road side or in dedicated locations such as at junctions or near parking spaces. The RSU is equipped with one network device for a dedicated short range communication based on IEEE802.11p radio technology, and can also be equipped with other network devices so as to be used for the purpose of communication within the infrastructural network. The OBU is also a wave device usually mounted on-board a vehicle used for exchanging information with RSUs or with other OBUs. VANET employs vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) to improve road safety and driving experience. In V2V communications, vehicles communicate directly with the OBUs on other vehicles or use other moving vehicles as relay nodes to increase transmission range and transmission rate. In V2I communications, the vehicles can do uplink and downlink transmissions when driving close to RSUs. This can provide a high transmission rate and minimal latency between RSUs and vehicles [6].

1.1.3 Cognitive Radio

Cognitive Radio (CR) is an adaptive, intelligent radio and network technology that can automatically detect available channels in a wireless spectrum and change transmission parameters enabling more communications to run concurrently and also improve radio operating behavior [7]. When combined with VANET, Cognitive Radio becomes the Cognitive enabled Vehicular ad hoc network (CR-VANET). That is employed as a solution for overcoming the limitation of channel utilization at RSUs. With CR-VANET, two types of users are defined—the licensed users on their own spectrum bands, which are known as Primary users (PUs), and the unlicensed users that sense the idle licensed spectrum (also called as spectrum holes, illustrated in Figure 1.3) and use the channel opportunistically [8], which are known as Secondary users (SUs). We guarantee the priority of PUs' service requests in our work. And SUs, which have the lower priority, can only access the idle channels when all PUs' requests are satisfied. The network provides Quality-of-Service (QoS) provision by the cooperation with the base station covering the RSU and vehicles.

1.2 Motivation and Objectives

For a cellular network, in urban scenarios such as intersections and highways, the transmission rate will be seriously affected when the number of vehicle users becomes too large. VANET is proposed

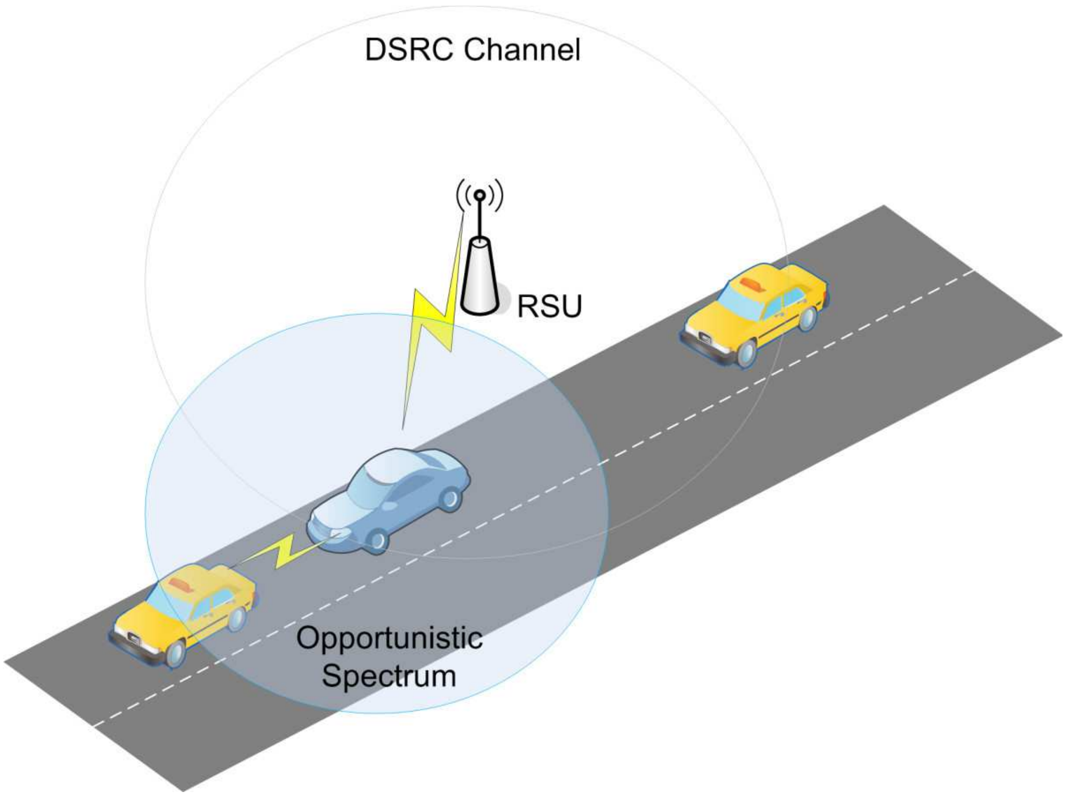

Figure 1.3: Opportunistic spectrum used by Secondary user.

as a solution to that problem. However, VANET still faces the problem of spectrum scarcity due to the following reasons: 1) the ever-increasing infotainment applications and various service requests demand a large amount of spectrum resource; 2) an RSU only has a limited number of channels to allocate and thus cannot handle high vehicle density in urban areas. Moreover, some vehicles in the system provide essential services related to public safety [9], such as police cars, fire trucks, and ambulances. The services provided by those vehicles should have higher priority and more channel resources. The channel allocation of RSUs in VANET must be fully utilized to satisfy these demands. An efficient channel allocation scheme is seriously demanded to solve the aforementioned spectrum scarcity and achieve channel resource utilization in VANET.

The main objectives of this research are to design a channel allocation scheme for RSUs in CR-VANETs and solve the channel allocation problem to get the optimal allocation policy.

1.3 Contributions

The related conference paper was accepted by 2019 IEEE Globecom Workshops (GC Wkshps): IEEE GLOBECOM Workshop on Artificial Intelligence for Next-Generations Wireless Communications [10]. In our work, we formulate the channel resource management problem in CR-VANETs as a Semi-Markov decision process (SMDP) and introduce two Reinforcement learning (RL)-based channel allocation methods, i.e., a model-based dynamic programming method and a model-free reinforcement learning method, to obtain an asymptotically optimal channel allocation policy and maximize the long-term average system reward. Moreover, we compare two of our methods with the Greedy method through simulations. The main contributions of this thesis are summarized as follows.

1) We model the channel resource allocation of a single RSU in CR-VANETs as an SMDP and propose a model-based dynamic programming method to find the asymptotically optimal policy. Based on some assumptions, we degrade the SMDP into a continuous-time Markov decision process (CTMDP) so that the Bellman equation can be used iteratively to calculate the best policy. We use a relative value iteration (RVI) algorithm to conduct the calculation. Besides, we focus on maximizing long-term average reward with our policies. In a decision-making problem like the channel allocation, it may be preferable to compare policies on the basis of their time-averaged expected reward rather than their expected total discounted reward [11].

2) We propose a model-free reinforcement learning method to solve the channel allocation problem without the need of any assumptions of the environment. This method employs an agent to interact with the environment and learn from the feedback. With iterations of exploring and updating its knowledge of the environment, the agent can obtain an optimal channel allocation policy in the end. Since it is model-free and we just need to leave the agent to explore the environment, the RL method can be used in many schemes where parameters like the distribution of the users' arrival could be different.

3) We make comparisons between the two proposed algorithms and the greedy algorithm. The results show that our RL algorithm can achieve a similar performance to that of the RVI

algorithm, which is better than that of the greedy algorithm. Furthermore, we evaluate the impact of various arrival rates and completion rates of service requests on the system performance.

1.4 Related Work

1.4.1 Spectrum Resource Allocation Management

In recent years, there have been a lot of works studying CR-VANETs. Figure 1.4 presents a taxonomy of recent advances in CR-VANTEs. [12] and [13] introduced coordinated spectrum sensing methods for CR-VANETs to achieve better sensing precision and efficiency. The network architecture based on cognitive vehicular networking was designed in [14] to provide wireless connectivity to both the general public and emergency responders in emergency cases. The problem of route planning was solved by studying TV band white space in CR based high-speed vehicle network in [15]. In [16], a peer-to-peer-based approach for content distribution in VANETs was proposed. In that paper, vehicle users can exchange data and complement the missing packets for other users with CR-based V2V communications. A distributed and adaptive resource management was proposed for optimal exploitation of cognitive radio and soft-input/soft-output data fusion in vehicular access networks [17] [18], in which the energy and computing limited car smartphones were enhanced

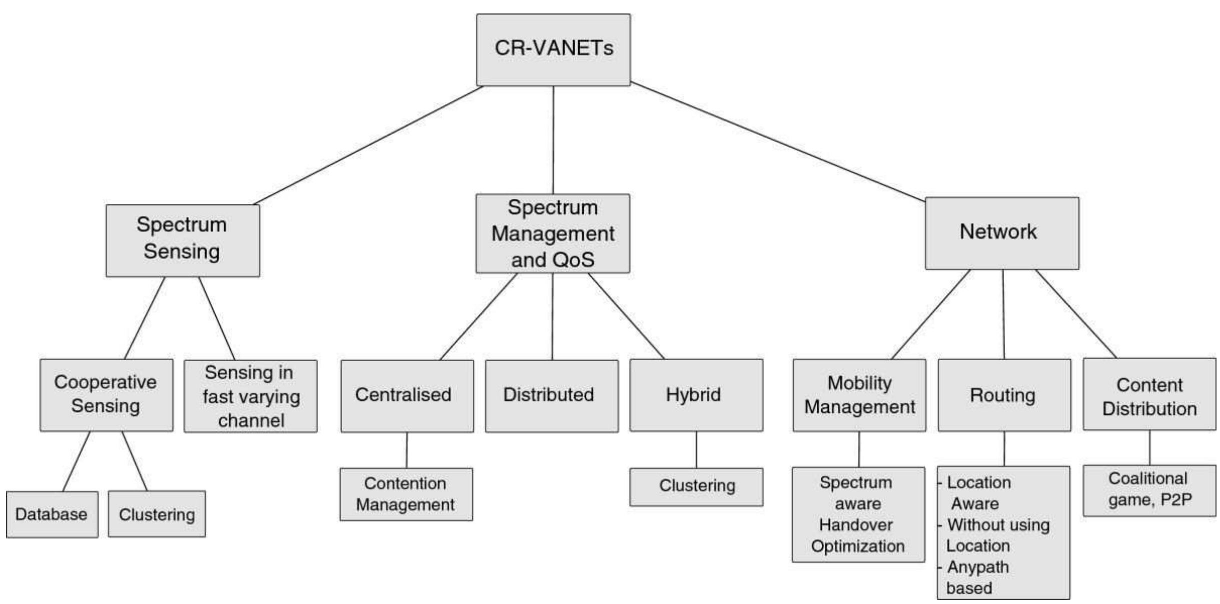

Figure 1.4: Recent advances in CR-VANETs.

by offloading their traffic to the local or remote cloud.

Furthermore, some works study spectrum resource allocation management in CR-VANETs by different methods including static optimization algorithms [19], [20], [21], [22] and dynamic optimization algorithms [23], [24], [25], [26], [27], [28]. In [19], a generalized Nash Bargaining solution was proposed to formulate the intercell resource allocation, and the convex optimization approach was used to solve the problem. The authors in [20] transformed the dynamic spectrum access into a convex bipartite matching problem by constructing a complete bipartite graph and defining proper weight vectors. The branch and bound method was employed to analyze the spectrum resource allocation optimization problem in [21]. A prediction-window-based channel allocation algorithm was proposed in [22], by which the number of delivered video layers can be chosen adaptively according to the relation between the data amount that the vehicle can receive in the current time slot and that it is possible to obtain in future time slots. In [23], the spectrum access process was modeled as a noncooperative congestion game, and a distributed algorithm was devised to achieve Nash Equilibrium. Huang $et\ al.$ used data mining to predict the channel with the greatest probability of channel availability in [24]. The author in [25] formulated the access of vehicles with access points as a finite-horizon sequential decision problem and solved it using dynamic programming. In [26], a joint channel allocation and adaptive video streaming algorithm were proposed and an auction mechanism was employed to solve the problem.

1.4.2 SMDP-based Resource Allocation

He $et\ al.$ proposed a Semi-Markov decision process (SMDP)-based resource allocation scheme to facilitate video streaming application in [27]. SMDP was also used in [28] to model the channel allocation of an RSU in CR-VANETs, and relative value iteration was used to achieve the maximization of the expected overall system reward. Moreover, there are many existing works focusing on resource allocation in vehicular cloud computing, mobile cloud networks, and software-defined Internet of Things networks using an SMDP-based model [29], [30], [31], [32], [33].

1.4.3 Model-free Reinforcement Learning

All these planning algorithms mentioned above are model-based, which means the system models need to be obtained before the execution of optimization. To simplify training the models, some strong assumptions have to be made in the model-based planning methods. Without these assumptions, it might be difficult or even impossible to employ algorithms to solve these problems. The model-free reinforcement learning method can be the perfect solution to that deficiency since it needs no assumption about the transition probabilities. However, only a few articles have adopted the RL method for resource allocation [34], [35], [36]. There is barely no reference using RL for channel allocation in CR-VANETs to the best of our knowledge.

1.5 Organization of the Thesis

The remainder of the thesis is organized as follows. In Chapter 2, we present our SMDP-based channel allocation scheme. Model-based dynamic programming, one of the two methods we propose to solve the channel allocation problem, is introduced in Chapter 3. Chapter 4 introduces another method to solve the channel allocation problem: model-free reinforcement learning method. Then these two methods are compared with the greedy method in Chapter 5. Finally, the conclusions and future work are presented in Chapter 6.

Chapter 2

SMDP-based Channel Allocation Scheme

2.1 Introduction

Due to some defects of vehicular ad hoc networks, an efficient channel allocation scheme for RSUs is needed to solve the spectrum scarcity and achieve channel resource utilization. We model our system environment as a Semi-Markov decision process (SMDP) [37] which is a special case of the Markov decision process (MDP).

An MDP is a discrete time stochastic control process [38]. It provides a mathematical framework for modeling decision making in situations where outcomes are partly random and partly under the control of a decision maker. MDPs are useful for studying optimization problems solved via dynamic programming and reinforcement learning. Markov decision processes are an extension of Markov chains; the difference is the addition of actions (allowing choice) and rewards (giving motivation). Conversely, if only one action exists for each state (e.g. ”wait”) and all rewards are the same (e.g. ”zero”), a Markov decision process reduces to a Markov chain. Figure 2.1 describes the Markov decision process. At each step during the process, the decision maker may choose to take an action available in the current state, resulting in the model moving to the next step and offering the decision maker a reward.

In discrete-time Markov Decision Processes, decisions are made at discrete time intervals. However, for continuous-time Markov decision processes (CTMDP) [39], decisions can be made at any time the decision maker chooses. In comparison to discrete-time Markov decision processes, continuous-time Markov decision processes can better model the decision making process for a system that has continuous dynamics, i.e., the system dynamics is defined by partial differential equations (PDEs).

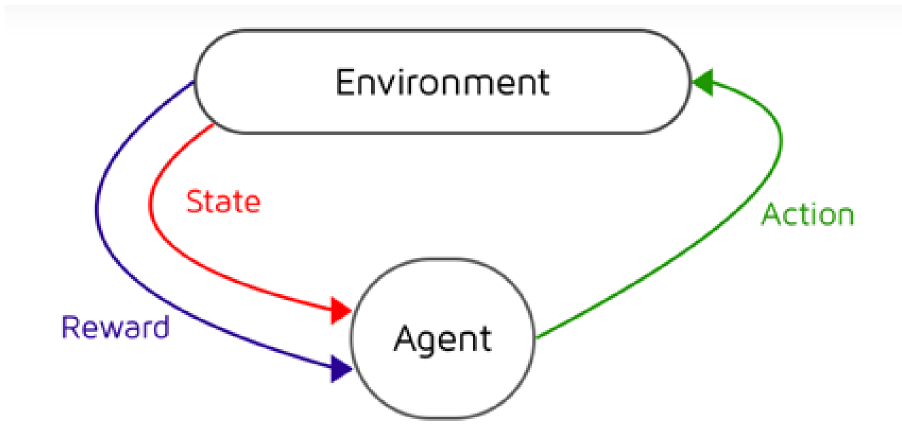

Figure 2.1: Markov decision process.

Semi-Markov decision processes, first introduced by Jewell [40] and De Cani [41], is a special case of continuous-time Markov decision processes where the sojourn time in a state is a continuous random variable with distribution depending on that state and the next state. Figure 2.2 contains dynamic diagrams representations of the Semi-Markov decision processes.

In this chapter, we present our SMDP-based channel allocation scheme. First, we introduce the system model in Section 2.2. Then, we model the process as an SMDP and give detailed information

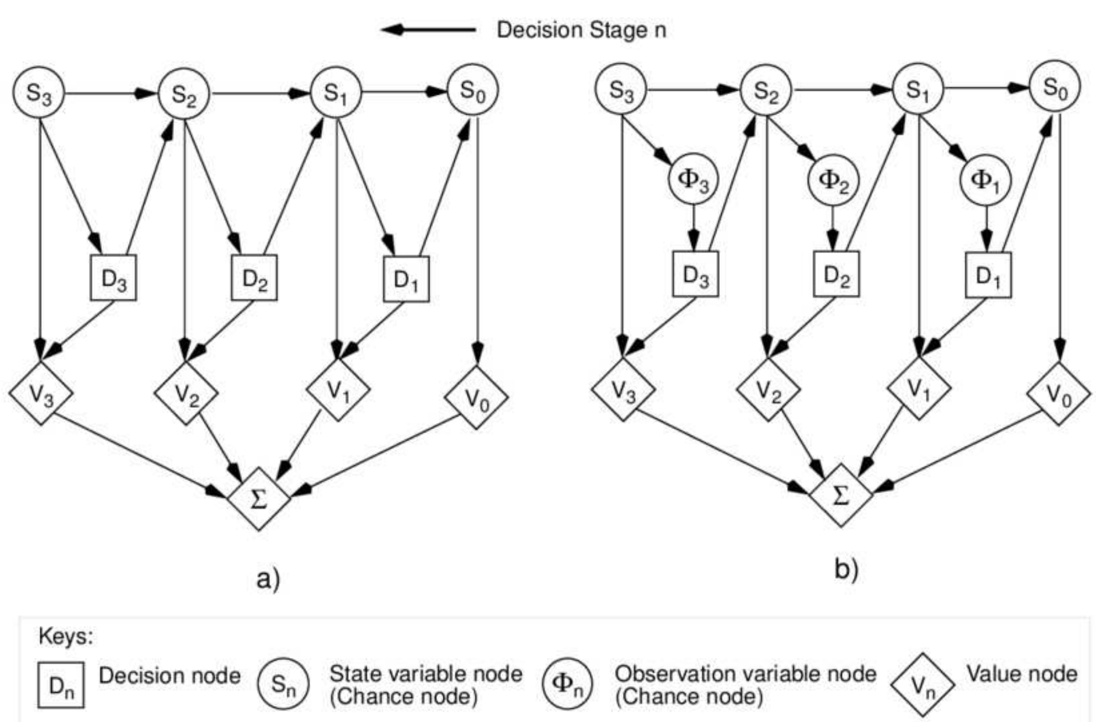

Figure 2.2: a) Structure of semi-Markov decision process. b)Structure of partially observable semi-Markov decision process.

about elements in the SMDP in Section 2.3. Section 2.4 describes the formulation of our problem and the objective of this thesis. A summary of this chapter is presented in Section 2.5.

2.2 System Model

We illustrate the channel allocation system as shown in Figure 2.3. In our system, we employ N RSUs along a straight road, each of which has a coverage diameter of d_R, where $R \in \{1, ..., N\}$. All RSUs and users are under one base station's coverage so that users in the system can always transmit and receive data either from the adjacent RSU or the base station. We assume that one RSU has K channels to allocate. The number of channels allocated to one service is denoted by c, where $c \in \{1, ..., C\}$, $C \leq K$. Thus, C is the maximum number of channels that can be allocated to one service. We assume that one channel can meet the minimum service requirement of any services, while more channels allocated mean higher transmission rate of that service request, and bring higher user satisfaction.

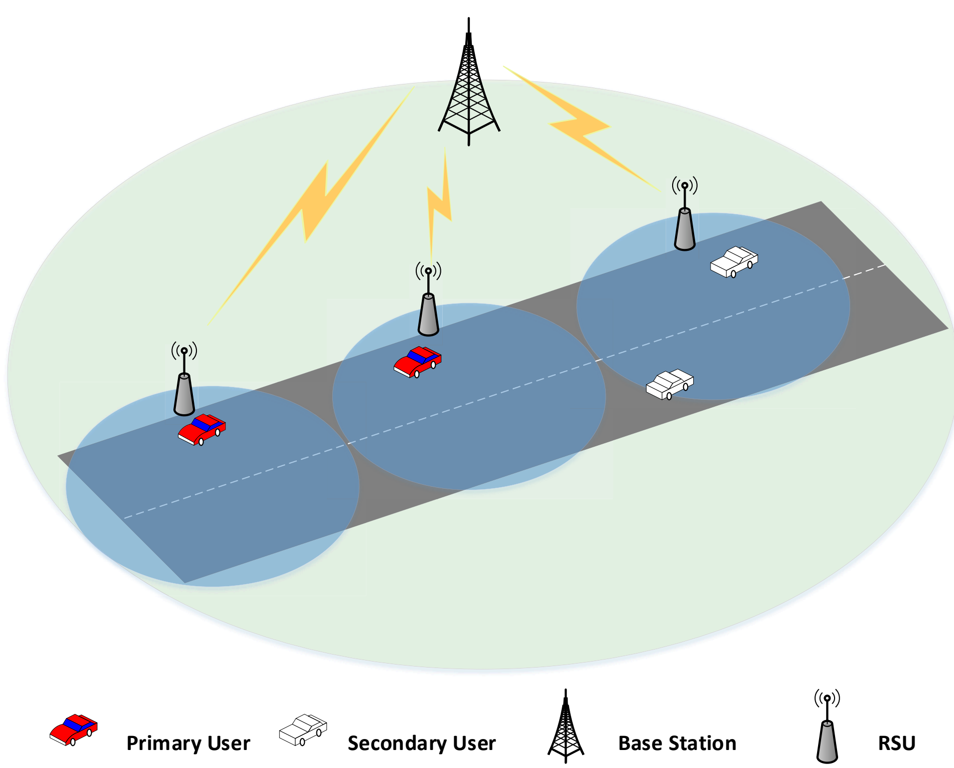

Figure 2.3: System model of cognitive enabled VANET.

The reason why we select VANET as our environment for resource allocation is that we want to test our methods under high arrival rates. In a dynamic system like VANET, the arrival rate of vehicle users is usually high so that we can get a better view of our methods.

The system has two types of users: 1) Primary Users: Primary users are vehicles such as emergency vehicles and mobile studio vehicles. These users have priority to transmit data in the licensed spectrum bands. 2) Secondary Users: Secondary users are vehicles that can only use primary band when there are available empty channels. The principles of channel allocations for these two kinds of users are introduced below.

When a vehicle enters the coverage of an RSU and plans to request service from RSU, the RSU will detect the type of the vehicle user and decide whether to accept the request or reject it based on channel resources availability. If the request is from the secondary user, RSU will accept the SU request as long as there are empty available channels. The number of channels that are allocated to that service request should be decided by the RSU. More channels mean higher transmission rate, which means more satisfaction income from that user. However, accepting that user's service request with more channels means less available channels for other incoming users. So the RSU should make good decisions to optimize the channel allocation problem. The request will be rejected if all channels of this RSU are busy. Thus, once an SU request arrives, the RSU will accept it and allocate available channels to the service or reject it. If the request is from the primary user, the RSU will always guarantee the priority of primary users. So the RSU will always accept the PU requests whether there are available channels or not. When there are idle channels, the RSU will allocate a certain number of channels to the PU service. The number should also be decided by the RSU. Otherwise, the RSU will stop a certain number of SU services to offer channels for that PU request when there is no empty channel. Those SU services will be transferred to the base station. And here the decision of the number should also be optimized by the RSU. There is a tradeoff between the transmission rate and the transferring cost for those SU services. The flowchart describing the above process is shown in Figure 2.4. The notations and variables used in this thesis are listed in Table 2.1.

When all of the channels are busy, since the system always provides the channel resources to primary users, we may need to transfer SU's service which occupying the RSU channels to the base station or shrink channels of another PU service. Figure 2.5 gives examples of channel allocation

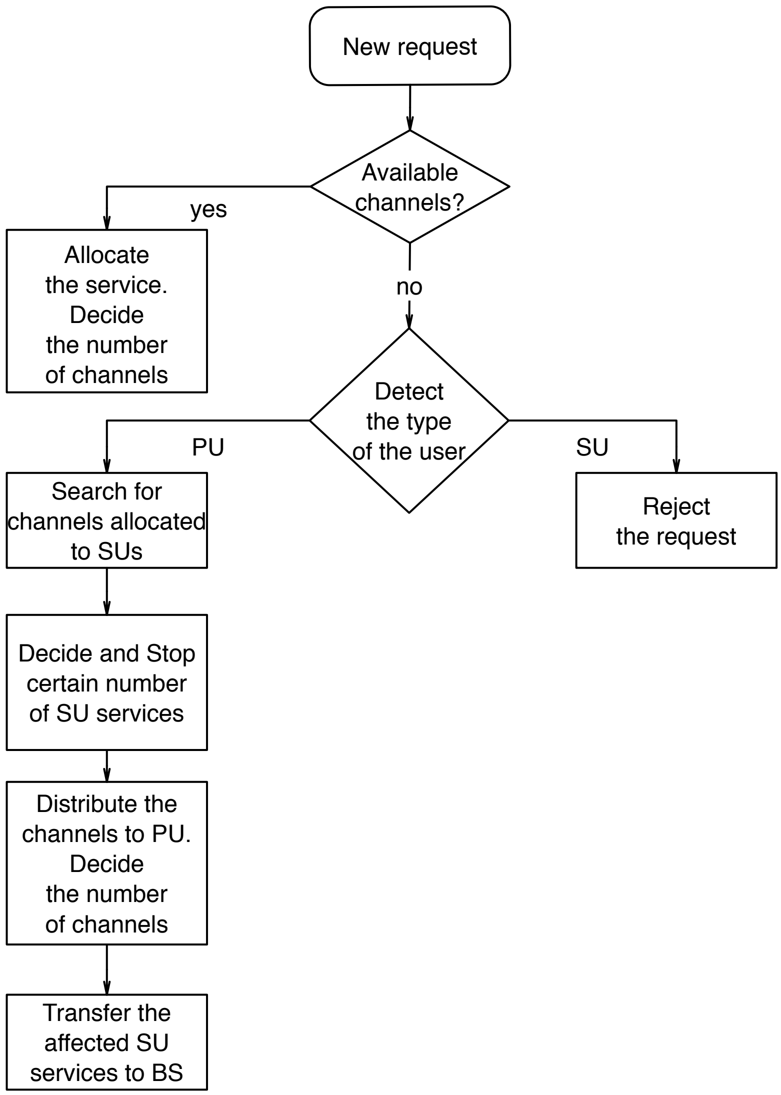

Figure 2.4: Flowchart of channel allocation process.

when channels are busy. When a request from a primary user arrives in (a), one or more services for

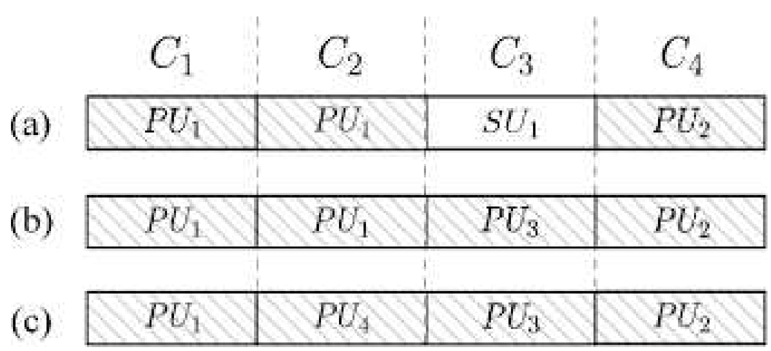

Figure 2.5: Channel allocation when channels are busy.

Table 2.1
LIST OF NOTATIONS AND VARIABLES

Symbol	**Description**
K	The maximum number of channels at one RSU
C	The maximum number of channels that can be allocated to one service
c	The number of channels that is allocated to one service
n_{pc}	The number of PU services allocated with c channels
n_{sc}	The number of SU services allocated with c channels
A_p	The arrival event of a PU service
A_s	The arrival event of an SU service
F_{pc}	The completion event of a PU service with c channels
F_{sc}	The completion event of a PU service with c channels
R_s	The user income when an SU service request is accepted
R_p	The user income when a PU service request is accepted
R_t	The transmission cost for occupying one channel
E_t	The transfer cost for one SU service
U_t	The dropping cost for one channel
λ_p	The arrival rate of PU service requests
λ_s	The arrival rate of SU service requests
μ_p	The completion rate of PU service requests
μ_s	The completion rate of SU service requests
μ_d	The completion rate of one vehicle associating with the RSU

SUs determined by SMDP policy will be transferred to the base station with certain cost to accept the priority request, which is shown in (b). Then, in some extreme scenarios, when all channels are occupied by primary users and a new primary user arrives, the system will find the service occupying two or more channels and shrink it to accept a new request, which is illustrated in (c). Besides, in the scenario where every single channel is occupied by one primary user, the next event won't be the arrival of a primary user.

2.3 SMDP Model

We model the channel allocation process as a generic SMDP. In this SMDP, the distribution of the time to the next decision epoch and the state at that time depend on the past. The dependence is only through the state and action chosen at the current decision epoch. Generally, an SMDP can be represented as a 5-tuple $\{t_i, \mathcal{S}, \mathcal{A}, q, r\}$, where t_i, $i \in \mathbb{N}$ ($\mathbb{N}$ denotes the set of non-negative integers) is a decision epoch; $\mathcal{S}$ is the state space; $\mathcal{A}$ is the action space; q is the transition probability; and r is the reward function.

2.3.1 Decision Epoch

In an SMDP, the time interval between two adjacent decision epochs can be a duration within $[0, \infty]$, so that it can process service requests flexibly and promptly compared with a discrete-time MDP.

2.3.2 State Space

An SMDP is one type of decision processes other than a natural process. Unlike a natural process that models the state evolution of a system as if it were observed continually throughout time, a decision process represents the evolution of a system at decision epochs only. The timeline of the considered SMDP is illustrated in Figure 2.6. Vertical bars represent the time instants of event occurrence. Solid dots represent the time instants of action execution. A decision-making state is observed after an event occurs and before an action is taken. In the SMDP model, we only concern the decision-making states which consist of conventional states and events at the decision epochs. There are three components of the system state. The first two components are channels allocated to two types of services, denoted as $\boldsymbol{n}_p = [n_{p1}, ..., n_{pC}]^T$ and $\boldsymbol{n}_s = [n_{s1}, ..., n_{sC}]^T$, where n_{pc} represents

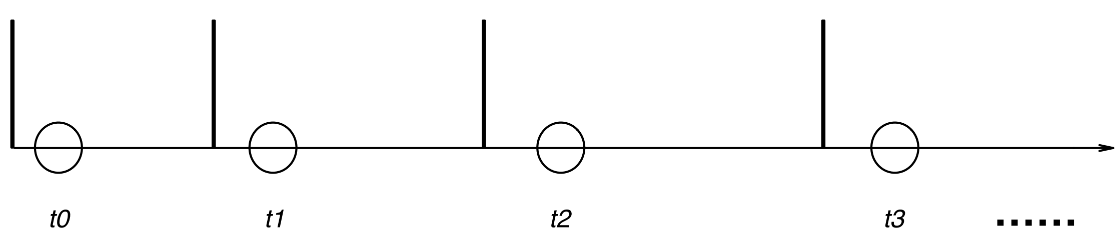

Figure 2.6: Timeline of the SMDP.

Algorithm 1 Searching all the states

Initialize state set $\mathcal{S} = \varnothing$.
For all $e \in \{A_p, A_s, F_{pc}, F_{sc}, F_{pm}, F_{sm}\}$
for $n_{sc1} = 0 : K$ **do**
  for $n_{sc2} = 0 : \frac{K - n_{sc1}}{2}$ **do**
    for $n_{pc1} = 0 : K - n_{sc1} - 2n_{sc2}$ **do**
      for $n_{pc2} = 0 : \frac{K - n_{sc1} - 2n_{sc2} - n_{pc1}}{2}$ **do**
        $\mathcal{S} \leftarrow \{s | s = \langle n_{sc1}, n_{sc2}, n_{pc1}, n_{pc2}, e\rangle\}$.
      end for
    end for
  end for
end for
Return $\mathcal{S}$ is the state set for all states s.

the number of PU services assigned with c channels and n_{sc} represents the number of SU services allocated with c channels. Obviously, n_{pc} and n_{sc} have to satisfy

$$\sum_{c=1}^{C} [c(n_{pc} + n_{sc})] \leq K. \tag{2.1}$$

The third component of the system state, denoted as e, where $e \in \{A_p, A_s, F_{pc}, F_{sc}, F_{pm}, F_{sm}\}$, is the system event. The events A_p and A_s represent the arrival events of a PU service request and an SU service request respectively. The elements F_{pc} and F_{sc} stand for the events that PU and SU services with c channels are completed or handed over. The event of handed over means that the services are handed over from one RSU to its base station or another RSU since the users are out of the RSU's coverage area. The elements F_{pm} and F_{sm} indicate the events that PU and SU services with m channels are completed or handed over, where "m channels" is a specific case of "c channels". We will discuss what this m stands for later in the state transition probabilities part of Chapter 3. Therefore, a system state can be formulated as $s = \langle \boldsymbol{n}_s, \boldsymbol{n}_p, e\rangle$. The system state space can be represented as

$$\mathcal{S} = \{s | s = \langle \boldsymbol{n}_s, \boldsymbol{n}_p, e\rangle\}. \tag{2.2}$$

We present an algorithm to search all the states s of the finite state space $\mathcal{S}$. The algorithm is introduced in Algorithm 1. We use the maximum number of channels that can be allocated to one service $C = 2$ as an example.

2.3.3 Action Space

When a new request occurs in the system, the system must choose the following actions. When the arrival event of an SU request occurs:

$$a_{\langle \boldsymbol{n}_s, \boldsymbol{n}_p, A_s \rangle} = \begin{cases} (s, m) & \text{accepts it with } m \text{ channels} \\ 0 & \text{rejects it.} \end{cases} \tag{2.3}$$

When the arrival event of a PU request occurs, the system will always accept the PU request. So the action of accepting the PU request with m channels is:

$$a_{\langle \boldsymbol{n}_s, \boldsymbol{n}_p, A_p \rangle} = (p, m, \boldsymbol{T}) \tag{2.4}$$

where $\boldsymbol{T} = [T_1, T_2, ..., T_C]^T$ is the transfer vector for SUs. Here T_c stands for the number of SU services allocated with c channels being transferred to the base station in order to offer the channel space for PU service. It can be seen that T_c has to satisfy

$$\sum_{c=1}^{C} cT_c \leq K \tag{2.5}$$

where $c \in \{1, ..., C\}$.

Once an ongoing service is completed, the channels will be released. Such action is denoted as $a_{\langle \boldsymbol{n}_s, \boldsymbol{n}_p, F \rangle} = -1$, where $F \in \{F_{pc}, F_{sc}, F_{pm}, F_{sm}\}$ is any completion event.

The action space is the set of all actions, as follows:

$$\mathcal{A} = \{a_{\langle \boldsymbol{n}_s, \boldsymbol{n}_p, A_s \rangle}, a_{\langle \boldsymbol{n}_s, \boldsymbol{n}_p, A_p \rangle}, a_{\langle \boldsymbol{n}_s, \boldsymbol{n}_p, F \rangle}\}. \tag{2.6}$$

We propose another algorithm to search a set of all the possible actions $\mathcal{A}$ for the finite state space $\mathcal{S}$. The algorithm description is stated in Algorithm 2.

Algorithm 2 Searching all the available actions

Initialize action sets $\mathcal{A}, A_1, A_2, A_3 = \varnothing$.
Set $X \leftarrow \{s|e(s) =\in \{F_{pc}, F_{sc}, F_{pm}, F_{sm}\}\}, Y \leftarrow \{s|e(s) = A_s\}, Z \leftarrow \{s|e(s) = A_p\}, c = 1, ..., C$. Ensure that $\sum_{c=1}^{C}[c(n_{pc} + n_{sc})] \leq K$ for state s.
if $s \in X$ **then**
    $A_1 \leftarrow \{a|a = -1\}$.
else if $s \in Y$ **then**
    for $m = 0 : C$ **do**
        $\Omega_s \leftarrow \{s| \sum_{c=1}^{C}[c(n_{pc} + n_{sc})] + m \leq K\}$.
        if $s \in \Omega_s$ and $m \neq 0$ **then**
            $A_2 \leftarrow \{a|a = \langle s, m\rangle\} \cup A_2$.
        else
            $A_2 \leftarrow \{a|a = 0\} \cup A_2$.
        end if
    end for
else if $s \in Z$ **then**
    for $m = 1 : C$ **do**
        $\Omega_p \leftarrow \{s| \sum_{c=1}^{C}[c(n_{pc} + n_{sc})] + m \leq K\}$.
        if $s \in \Omega_p$ **then**
            $A_3 \leftarrow \{a|a = \langle p, m, 0\rangle\} \cup A_3$.
        else
            Find all possible $\boldsymbol{T}$ that $\sum_{c=1}^{C}[c(n_{sc} - T_c + n_{pc})] + m \leq K$, where $T_c \leq n_{sc}, c = 1, ..., C$.
            $A_3 \leftarrow \{a|a = \langle p, m, \boldsymbol{T}\rangle\} \cup A_3$.
        end if
    end for
end if
Return $\mathcal{A} = A_1 \cup A_2 \cup A_3$, $\mathcal{A}$ is the possible action set for state s.

2.3.4 Transition Probabilities

The term $q(t, j|s, a)$ can represent the transition probability that next decision epoch whose system state is j occurs at or before time t. The state transition probability $P(j|s, a)$ and the expected time interval between two adjacent decision epochs $\tau(s, a)$ can be obtained based on $q(t, j|s, a)$, as follows:

$$P(j|s, a) = \int_{t=0}^{\infty} dq(t, j|s, a) \tag{2.7}$$

$$\tau(s, a) = \sum_{j \in \mathcal{S}} \int_{t=0}^{\infty} t dq(t, j|s, a). \tag{2.8}$$

In practice, $P(j|s, a)$ and $\tau(s, a)$ need to be trained using the data collected from the system environment.

2.3.5 Reward Model

The reward function consists of two parts: users income $i(s,a)$ and system cost $c(s,a)$. Then the reward function can be formulated as:

$$r(s,a) = i(s,a) - c(s,a). \tag{2.9}$$

The system will lose secondary user's satisfaction income if the service is rejected. When the service is accepted, the completion time of the service will be less if more channels are assigned to this service, so the transmission cost for occupying the channels will be less. If SU services are handed over to base station when a PU service is accepted, there will be additional costs for transferring the SU services and dropping the channels. So $i(s,a)$ can be defined as follows:

$$i(s,a) = \begin{cases} 0, & a = -1 \\ -R_s, & a = 0 \\ R_s - R_t/m, & a = (s,m) \\ R_p - R_t/m - \sum_{c=1}^{C} T_c E_t & a = (p,m,\boldsymbol{T}) \\ \qquad - \sum_{c=1}^{C} c T_c U_t, & \end{cases} \tag{2.10}$$

where R_s and R_p stand for the user satisfaction income for SU and PU respectively. R_t is the transmission cost for occupying one channel. E_t is the transfer cost for one SU service and U_t is the dropping cost for one channel.

Then system cost $c(s,a)$ can be defined as:

$$c(s,a) = \tau(s,a) o(s,a) \tag{2.11}$$

where $\tau(s,a)$ is the time interval and $o(s,a)$ is the cost rate of the system, which is determined by the total number of occupied channels

$$o(s,a) = \sum_{c=1}^{C} c(n_{sc} + n_{pc}). \tag{2.12}$$

2.4 Problem Formulation

We formulate this channel allocation problem as an infinite-horizon average Semi-Markov decision problem in this thesis. We focus on the expected long-term average reward, which can be defined as [11]

$$g^{\pi}(s_0) = \lim_{N\to\infty} \frac{E^{\pi}_{s_0}\Big\{\sum_{m=0}^{N}\big[i(s_m,a_m) - \tau_m o(s_m,a_m)\big]\Big\}}{E^{\pi}_{s_0}\Big\{\sum_{m=0}^{N}\tau_m\Big\}} \tag{2.13}$$

where s_0 is the starting state and π is the policy we follow, s_m and a_m refer to the state and action at the decision epoch m, τ_m is the time interval between the mth and $(m+1)$th decision epoch. In this thesis, we assume that for every stationary policy, the embedded Markov chain has a unichain transition probability matrix (i.e., every stationary deterministic policy $\bar{\pi}$, which is a mapping $\bar{\pi} : \mathcal{S} \to \mathcal{A}$, has a single recurrent class plus a possibly empty set of transient states). Under this assumption, the expected average reward of every stationary policy does not vary with the initial state, namely $g^{\pi} = g^{\pi}(s_0)$, for all $s_0 \in \mathcal{S}$ [11].

Using the above average reward instead of other metrics, e.g., system throughput, we adopt the perspective of the users while finding the optimal policy. By doing so, we take user satisfaction and fairness into account.

The objective of this thesis is to find an optimal policy π^* to maximize the long-term average reward

$$\pi^* \in \arg\max_{\pi} g^{\pi}. \tag{2.14}$$

2.5 Summary

In this chapter, an efficient channel allocation scheme for RSUs in CR-VANET is proposed. Then an SMDP model is designed to formulate the channel allocation process. In addition, the objectives of this thesis, which are finding the optimal policy for our channel allocation problem and maximizing the long-term average system reward, are presented in the problem formulation section.

Chapter 3

Model-based Dynamic Programming Method

3.1 Introduction

Chapter 2 introduces our SMDP-based channel allocation scheme. It only describes our SMDP model and our aims of this problem. We still need methods to solve this channel allocation problem. In this chapter, we present our first method to solve the SMDP: a model-based planning method which is called dynamic programming.

Dynamic Programming is a method for solving a complex problem by breaking it down into a collection of simpler subproblems, solving each of those subproblems just once, and storing their solutions using a memory-based data structure (array, map, etc). Each of the subproblem solutions is indexed in some way, typically based on the values of its input parameters, so as to facilitate its lookup. The next time the same subproblem occurs, instead of recomputing its solution, one simply looks up the previously computed solution, thereby saving computation time. This technique of storing solutions to subproblems instead of recomputing them is called memoization. If sub-problems can be nested recursively inside larger problems, so that dynamic programming methods are applicable, then there is a relation between the value of the larger problem and the values of the sub-problems [42] [43]. In the optimization literature, this relationship is called the Bellman equation.

We illustrate the schematic of our model-based planning method in Figure 3.1. First, the state transition probabilities and the expected time intervals between decision epochs of the SMDP model are trained using the data collected from the environment. Then the planning algorithm is leveraged to find an optimal channel allocation policy. Generally, the model-based planning method can be closed-loop, i.e., the model is continuously being trained during the system operation so that

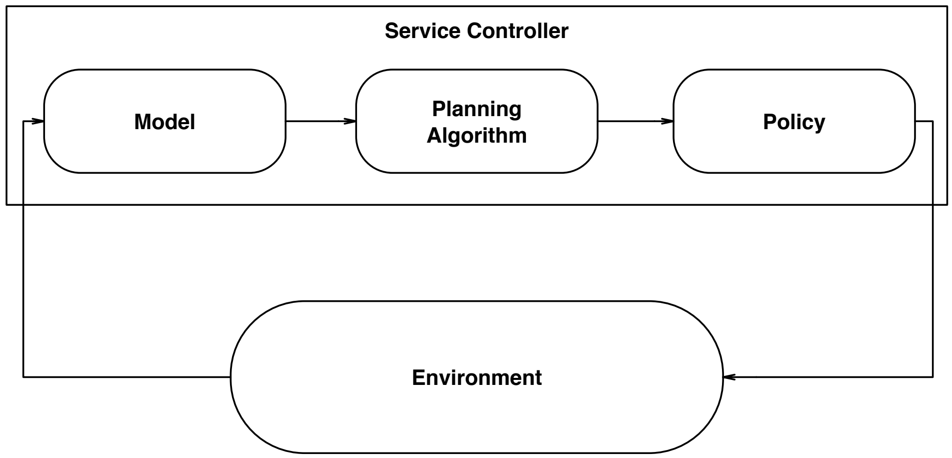

Figure 3.1: Schematic of model-based planning method.

the policy can be adaptively improved according to the change of the environment [35].

In this chapter, the proposed dynamic programming method is introduced in detail. We first present the optimization formulation of the SMDP and then degrade the SMDP to CTMDP to simplify the model and solve the planning for the control problem in Section 3.2. Next, we calculate the state transition probabilities that have a significant effect on deriving the optimal policy in Section 3.3. Then a relative value iteration algorithm is provided in Section 3.4 to solve the CTMDP. Finally, Section 3.5 summarizes this chapter.

3.2 Optimization Formulation and Degradation

The maximum long-term average reward can be obtained by the Bellman optimality equation:

$$v(s) = \max_{a \in \mathcal{A}} \left\{ r(s,a) - g^* \tau(s,a) + \sum_{j \in \mathcal{S}} P(j|s,a) v(j) \right\} \quad \forall s \in \mathcal{S} \tag{3.1}$$

where g^* is the average reward under an optimal policy π^*, $r(s,a)$ denotes the expected reward function. We can see from the Bellman equation that once we know the state transition probabilities $P(j|s,a)$ and expected time interval between two adjacent decision epochs $\tau(s,a)$, we can employ a dynamic programming algorithm (such as value iteration algorithm or policy iteration algorithm) to find the optimal policy that maximize $v(s)$.

To simplify the model, we assume that both users arrive with Poisson distribution, with mean rates λ_p for primary users and λ_s for secondary users. The residence time for one vehicle associated with one RSU follows the exponential distribution with the parameter of μ_d. And the service operating time follows an exponential distribution with the parameters of $c\mu_p$, $c\mu_s$ if c channels are assigned to that PU or SU service, respectively. These assumptions are reasonable in certain scenarios according to some practical measurements [44]. Under these assumptions, the time interval between two adjacent decision epochs follows the exponential distribution. The SMDP proposed above can be degraded to CTMDP.

3.3 State Transition Probabilities

Suppose that the system is at the state s, and an action a is selected at the current state, then the system will transit to the new state j. The time duration from these two continuous decision epochs is denoted by $\tau(s, a)$. Since the system model is a CTMDP now, from any state-action pair (s, a), the time intervals to all the next service request arrivals and ongoing service completions or being handed over are assumed to be independent exponential variables. The expected value of the minimal time interval is namely the expected time interval $\tau(s, a)$. Therefore, the mean occurrence rate $\gamma(s, a)$ of event for a state-action pair (s, a) is the summation of the rates of all events in the system, which is the reciprocal of $\tau(s, a)$

$$\gamma(s, a) = \frac{1}{\tau(s, a)}. \tag{3.2}$$

In order to get the specific expression of the event occurrence rate for every available state-action pair, we present the following proposition.

Proposition 1. *Suppose that $X_1, X_2, ..., X_n$ are independent exponential variables, with X_i having parameter $\phi_i, i = 1, 2, ..., n$. It turns out that the minimum of X_i is an exponential random variable with a parameter equal to the sum of the ϕ_i.*

Proof: Since

$$
\begin{aligned}
P\{\min(X_1, ..., X_n) > x\} &= P\{X_i > x \text{ for each } i = 1, ..., n\} \\
&= \prod_{i=1}^{n} P\{X_i > x\} \text{ (by independence)} \\
&= \prod_{i=1}^{n} e^{-\phi_i x} = \exp\left\{-\left(\sum_{i=1}^{n} \phi_i\right) x\right\}
\end{aligned}
$$

the cumulative distribution function of random variable $X = \min(X_1, ..., X_n)$ is

$$F_X(x) = 1 - \exp\left\{-\left(\sum_{i=1}^{n} \phi_i\right) x\right\}.$$

Therefore, the minimum of X_i is an exponential random variable with a parameter equal to $(\sum_{i=1}^{n} \phi_i)$. □

According to Proposition 1, the time interval from state-action pair (s, a) to the first event occurrence among the service request arrivals and the ongoing service completions follows an exponential distribution with parameter $\gamma(s, a)$. Using Proposition 1 with $\{\phi_i\}$ substituted by $\{\lambda_p, \lambda_s\}$ and $\{\mu_p, \mu_s, \mu_d\}$, the mean rate $\gamma(s, a)$ of events can be expressed as

$$
\gamma(s,a) = \begin{cases}
\gamma_0(s,a) + (m\mu_s + \mu_d), & e = A_s \quad a = (s, m) \\
\gamma_0(s,a) - \sum_{c=1}^{C} T_c(c\mu_s + \mu_d) & e = A_p \quad a = (p, m, \boldsymbol{T}) \\
\qquad +(m\mu_p + \mu_d), & \\
\gamma_0(s,a), & \text{others}
\end{cases} \tag{3.3}
$$

where $\gamma_0(s, a)$ can be further denoted by

$$\gamma_0(s,a) = \lambda_p + \lambda_s + \sum_{c=1}^{C} [n_{sc}(c\mu_s + \mu_d) + n_{pc}(c\mu_p + \mu_d)] \tag{3.4}$$

where λ_p and λ_s are the arrival rates for PUs requests and SUs requests, respectively. The completion rates for one PU service and SU service which is allocated with c channels are $c\mu_p$ and $c\mu_s$. The hand-off rate for one service is μ_d.

When a PU or SU service is completed, or a SU service is rejected by the network, there exists $\sum_{c=1}^{C}(n_{sc} + n_{pc})$ services in the system, then the completion rate of all these services is $\sum_{c=1}^{C}(c\mu_s n_{sc} + c\mu_p n_{pc})$, and the hand-off rate is $\sum_{c=1}^{C} \mu_d(n_{sc} + n_{pc})$.

When an SU service is accepted with m channels, the number of the services in the system will increase by one, the completion rate and hand-off rate will increase by $m\mu_s$ and μ_d respectively.

When a PU service is accepted with m channels, and $\sum_{c=1}^{C} T_c$ of SU services allocated with c channels are transferred to the base station, the number of services in the system will increase by $1 - \sum_{c=1}^{C} T_c$. The completion rate and hand-off rate will increase by $m\mu_p - \sum_{c=1}^{C} cT_c\mu_s$ and $\mu_d - \sum_{c=1}^{C} T_c\mu_d$, respectively.

Now that the mean rate of events is derived, the state transition probability $P(j|s,a)$ can be calculated according to the following proposition.

Proposition 2. *Suppose X_1 and X_2 are two independent exponential variables with respective parameters ϕ_1 and ϕ_2. Then*

$$P(X_1 < X_2) = \frac{\phi_1}{\phi_1 + \phi_2}.$$

Proof: Since the random variables X_1 and X_2 are independent, the joint probability density function is

$$f(x_1, x_2) = \phi_1 e^{-\phi_1 x_1} \cdot \phi_2 e^{-\phi_2 x_2}.$$

Then

$$P(X_1 < X_2) = \int_0^{\infty} \int_0^{x_2} f(x_1, x_2)\, dx_1\, dx_2 = \frac{\phi_1}{\phi_1 + \phi_2}.$$

□

For the current state $s = \langle \boldsymbol{n}_s, \boldsymbol{n}_p, e \rangle$, where $e \in \{F_{pc}, F_{sc}\}$ and the action a = -1, the next state may be one of the following states:

$$\begin{cases} j_1 = \langle \boldsymbol{n}_s, \boldsymbol{n}_p, A_s \rangle \\ j_2 = \langle \boldsymbol{n}_s, \boldsymbol{n}_p, A_p \rangle \\ j_3 = \langle \boldsymbol{n}_s, \boldsymbol{n}_p - \boldsymbol{I}_c, F_{pc} \rangle \\ j_4 = \langle \boldsymbol{n}_s - \boldsymbol{I}_c, \boldsymbol{n}_p, F_{sc} \rangle \end{cases} \tag{3.5}$$

where the vector $\boldsymbol{I}_c$ represents a vector with C elements, where the c-th element is 1, and others are 0. $P(j|s,a)$ can be obtained as

$$P(j|s,a) = \begin{cases} \frac{\lambda_s}{\gamma_(s,a)}, & j = j_1 \\ \frac{\lambda_p}{\gamma_(s,a)}, & j = j_2 \\ \frac{n_{pc}c\mu_p + n_{pc}\mu_d}{\gamma_(s,a)}, & j = j_3 \\ \frac{n_{sc}c\mu_s + n_{sc}\mu_d}{\gamma_(s,a)}, & j = j_4. \end{cases} \tag{3.6}$$

For the current state $s = \langle \boldsymbol{n}_s, \boldsymbol{n}_p, A_s \rangle$ and the action $a = 0$, the next state will be one of the following states:

$$\begin{cases} j_1 = \langle \boldsymbol{n}_s, \boldsymbol{n}_p, A_s \rangle \\ j_2 = \langle \boldsymbol{n}_s, \boldsymbol{n}_p, A_p \rangle \\ j_3 = \langle \boldsymbol{n}_s, \boldsymbol{n}_p - \boldsymbol{I}_c, F_{pc} \rangle \\ j_4 = \langle \boldsymbol{n}_s - \boldsymbol{I}_c, \boldsymbol{n}_p, F_{sc} \rangle \end{cases} \tag{3.7}$$

The transition probabilities $P(j|s,a)$ are:

$$P(j|s,a) = \begin{cases} \frac{\lambda_s}{\gamma_(s,a)}, & j = j_1 \\ \frac{\lambda_p}{\gamma_(s,a)}, & j = j_2 \\ \frac{n_{pc}c\mu_p + n_{pc}\mu_d}{\gamma_(s,a)}, & j = j_3 \\ \frac{n_{sc}c\mu_s + n_{sc}\mu_d}{\gamma_(s,a)}, & j = j_4. \end{cases} \tag{3.8}$$

For the current state $s = \langle \boldsymbol{n}_s, \boldsymbol{n}_p, A_s \rangle$ and the action $a = (s, m)$, the next state will be one of the following states:

$$\begin{cases} j_1 = \langle \boldsymbol{n}_s + \boldsymbol{I}_m, \boldsymbol{n}_p, A_s \rangle \\ j_2 = \langle \boldsymbol{n}_s + \boldsymbol{I}_m, \boldsymbol{n}_p, A_p \rangle \\ j_3 = \langle \boldsymbol{n}_s, \boldsymbol{n}_p, F_{sm} \rangle \\ j_4 = \langle \boldsymbol{n}_s + \boldsymbol{I}_m, \boldsymbol{n}_p - \boldsymbol{I}_m, F_{pm} \rangle \\ j_5 = \langle \boldsymbol{n}_s + \boldsymbol{I}_m - \boldsymbol{I}_c, \boldsymbol{n}_p, F_{sc} \rangle \\ j_6 = \langle \boldsymbol{n}_s + \boldsymbol{I}_m, \boldsymbol{n}_p - \boldsymbol{I}_c, F_{pc} \rangle \end{cases} \tag{3.9}$$

where $\boldsymbol{I}_m$, similar to $\boldsymbol{I}_c$, indicates a vector with C elements, where the m-th element is 1, and others are 0 with $m \in \{1, ..., C\}$. The events F_{sm} and F_{pm} here are special cases where a service allocated with m channels is completed while the next coming event is an arrival event and that service is allocated with the same number of channels, where the number $c \in \{1, ..., C\}$ and $c \neq m$. $P(j|s,a)$ can be obtained as

$$P(j|s,a) = \begin{cases} \frac{\lambda_s}{\gamma_(s,a)}, & j = j_1 \\ \frac{\lambda_p}{\gamma_(s,a)}, & j = j_2 \\ \frac{(n_{sm}+1)(m\mu_s+\mu_d)}{\gamma_(s,a)}, & j = j_3 \\ \frac{n_{pm}(m\mu_p+\mu_d)}{\gamma_(s,a)}, & j = j_4 \\ \frac{n_{sc}(c\mu_s+\mu_d)}{\gamma_(s,a)}, & j = j_5 \\ \frac{n_{pc}(c\mu_p+\mu_d)}{\gamma_(s,a)}, & j = j_6. \end{cases} \tag{3.10}$$

For the current state $s = \langle \boldsymbol{n}_s, \boldsymbol{n}_p, A_p \rangle$ and the action $a = (p, m, \boldsymbol{T})$, the next state will be one of the following states:

$$\begin{cases} j_1 = \langle \boldsymbol{n}_s - \boldsymbol{T}, \boldsymbol{n}_p + \boldsymbol{I}_m, A_s \rangle \\ j_2 = \langle \boldsymbol{n}_s - \boldsymbol{T}, \boldsymbol{n}_p + \boldsymbol{I}_m, A_p \rangle \\ j_3 = \langle \boldsymbol{n}_s - \boldsymbol{T} - \boldsymbol{I}_m, \boldsymbol{n}_p + \boldsymbol{I}_m, F_{sm} \rangle \\ j_4 = \langle \boldsymbol{n}_s - \boldsymbol{T}, \boldsymbol{n}_p, F_{pm} \rangle \\ j_5 = \langle \boldsymbol{n}_s - \boldsymbol{T} - \boldsymbol{I}_c, \boldsymbol{n}_p + \boldsymbol{I}_m, F_{sc} \rangle \\ j_6 = \langle \boldsymbol{n}_s - \boldsymbol{T}, \boldsymbol{n}_p + \boldsymbol{I}_m - \boldsymbol{I}_c, F_{pc} \rangle \end{cases} \tag{3.11}$$

where $\boldsymbol{T} = [T_1, T_2, ..., T_C]^T$ is the transfer vector for SUs. $P(j|s,a)$ can be obtained as

$$P(j|s,a) = \begin{cases} \frac{\lambda_s}{\gamma_(s,a)}, & j = j_1 \\ \frac{\lambda_p}{\gamma_(s,a)}, & j = j_2 \\ \frac{(n_{sm} - T_m)(m\mu_s + \mu_d)}{\gamma_(s,a)}, & j = j_3 \\ \frac{(n_{pm} + 1)(m\mu_p + \mu_d)}{\gamma_(s,a)}, & j = j_4 \\ \frac{(n_{sc} - T_c)(c\mu_s + \mu_d)}{\gamma_(s,a)}, & j = j_5 \\ \frac{n_{pc}(c\mu_p + \mu_d)}{\gamma_(s,a)}, & j = j_6. \end{cases} \tag{3.12}$$

3.4 Relative Value Iteration Algorithm

We can directly iterate the Bellman optimality equation to get the optimal policy that gives us the maximum $v(s)$ in a discrete-time MDP since the event occurrence rate $\gamma(s,a)$ is identical for every state-action pair (s,a). However, what we have is a continuous-time MDP. Therefore, the uniformization transformation of event occurrence rate should be added to apply the algorithm of the discrete-time MDP to the CTMDP. To realize the uniformization, we should uniformize the event occurrence rates of all state-action pairs by adding extra fictitious decisions. The method is introduced as follows.

For all $s \in \mathcal{S}$ and $a \in \mathcal{A}$, the uniform constant event occurrence rate ω has to satisfy the follow inequality:

$$[1 - P(s|s,a)]\gamma(s,a) \leq \omega. \tag{3.13}$$

We set ω as follows:

$$\omega = \lambda_p + \lambda_s + \sum_{c=1}^{C} \left[\frac{K}{c}\right] c(\mu_p + \mu_s) + \sum_{c=1}^{C} \left[\frac{K}{c}\right] \mu_d. \tag{3.14}$$

Thus, the uniformed reward function $\tilde{r}(s,a)$ and uniformed transition probability $\tilde{P}(s,a)$ are denoted as

$$\tilde{r}(s,a) = r(s,a)\frac{\gamma(s,a)}{\omega} \tag{3.15}$$

$$\tilde{g} = \frac{g^*}{\omega} \tag{3.16}$$

$$\tilde{P}(j|s,a) = \begin{cases} 1 - \frac{[1-P(s|s,a)]\gamma(s,a)}{\omega}, & j = s \\ \frac{P(j|s,a)\gamma(s,a)}{\omega}, & j \neq s. \end{cases} \tag{3.17}$$

Therefore, the Bellman equation can be rewritten as

$$\tilde{v}(s) = \max_{a \in \mathcal{A}} \left\{ \tilde{r}(s,a) - \tilde{g} + \sum_{j \in \mathcal{S}} \tilde{P}(j|s,a)\tilde{v}(j) \right\} \quad \forall s \in \mathcal{S}. \tag{3.18}$$

Since we have the uniform Bellman equation for our SMDP, we can apply a relative value iteration to solve the Bellman equation as shown in Algorithm 3. It conducts an exhaustive sweep over the whole state space at each step. An asymptotically optimal policy can be obtained by iteratively calculating the state value of the Bellman optimality equation [45]. Compared with the value iteration, the relative value iteration offers a much faster rate of convergence by substracting a constant after each iteration. Let $\Phi(\boldsymbol{v})$ represent the span of vector $\boldsymbol{v}$, which is defined as follows

$$\Phi(\boldsymbol{v}) = \max_{s \in \mathcal{S}} v(s) - \min_{s \in \mathcal{S}} v(s) \tag{3.19}$$

where $\boldsymbol{v}$ represents the vector whose elements are the values of all states. In the result, the relative

Algorithm 3 RVI Algorithm

1: Initialize $v^0(s) = 0$ for all states, choose a fixed state k^*, specify a small constant $\epsilon > 0$, and set $i = 0$.

2: For all $s \in \mathcal{S}$, calculate $v^{i+1}(s)$ as follows:

$$v^{i+1}(s) = \max_{a \in \mathcal{A}} \left\{ \tilde{r}(s,a) - v^i(k^*) + \sum_{j \in \mathcal{S}} \tilde{P}(j|s,a) v^i(j) \right\}$$

3: **if** $\Phi(\boldsymbol{v}^{i+1} - \boldsymbol{v}^i) < \epsilon$ **then**

4: $\quad i = i + 1$ and back to step 2.

5: **else**

6: $\quad$ For all $s \in \mathcal{S}$, choose an ϵ-optimal policy as follows:

$$d_\epsilon(s) \in \arg\max_{a \in \mathcal{A}} \left\{ \tilde{r}(s,a) - v^i(k^*) + \sum_{j \in \mathcal{S}} \tilde{P}(j|s,a) v^i(j) \right\}$$

7: **end if**

value iteration algorithm can obtain a vector of decision rules $d_\epsilon(s)$ that constitutes the optimal policy π^*.

3.5 Summary

In this chapter, we present a model-based dynamic programming method to solve our channel allocation problem. First, we introduce the Bellman equation and make some assumptions to degrade the SMDP model to CTMDP. Then state transition probabilities are calculated for each state action pair. Besides, an algorithm called RVI is applied to conduct the iterations of the Bellman equation. Since it is a model-based method, the algorithm can obtain the optimal policy within a short period with the knowledge of the system environment. After calculating the best policy, the system can get a better performance compared to model-free method by following the best policy.

Chapter 4

Model-free Reinforcement Learning Method

4.1 Introduction

Reinforcement learning is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward. This is accomplished by assigning rewards and punishments for agents' actions based on the temporal feedback obtained during the path of interactions of the learning agents with dynamic systems. Reinforcement learning is one of the three basic machine learning paradigms, alongside supervised learning and unsupervised learning. It differs from supervised learning in that labeled input/output pairs need not be presented, and sub-optimal actions need not be explicitly corrected. Instead, the focus is finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge) [46]. The main difference between the classical dynamic programming methods and reinforcement learning algorithms is that the latter does not assume knowledge of an exact mathematical model of the MDP and they target large MDPs where exact methods become infeasible [47] [48].

We make assumptions about the arrivals and operating times of two types of users to simplify the model in Chapter 3. However, these assumptions may not be valid for all scenarios in practice [49]. Therefore, the policy obtained from the model-based dynamic programming method is not necessarily optimal. We need a more practical, more general, and more intelligent method to solve this SMDP. Thus, we introduce a model-free reinforcement learning method to solve the channel allocation problem in this chapter. This method doesn't need the calculation of state transition probabilities and the expected time intervals between adjacent decision epochs. The working principle of reinforcement learning is first introduced in Section 4.2 and then we present an average reward RL algorithm to obtain the optimal policy in Section 4.3. Section 4.4 is the summary of this chapter.

4.2 Working Principle

We illustrate the reinforcement learning model in Figure 4.1. Any RL model basically contains 4 elements which are the environment, the agent, a set of actions, and the environmental response. The environment is typically formulated as a Markov decision process, as many reinforcement learning algorithms for this context utilize dynamic programming techniques [47] [48] [50]. There is a learning algorithm and a knowledge base inside the agent. The learning algorithm gathers the environmental response after the agent takes an action, and it derives information about the new state and the immediate reward, which can be used to update the knowledge base and select the next action. The knowledge base resembles a library that stores the experience data. The data could be tabular action values (Q learning), the weights of artificial neural networks (Deep Q learning), or the weights of linear function, etc. Knowledge bases for Q learning and Deep Q learning are

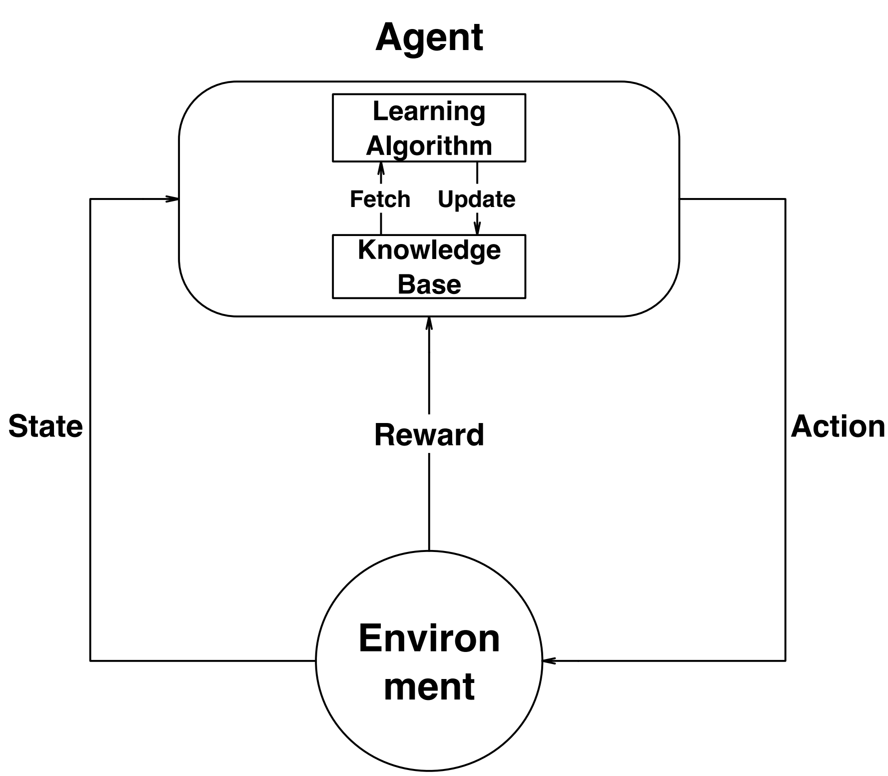

Figure 4.1: Reinforcement learning model.

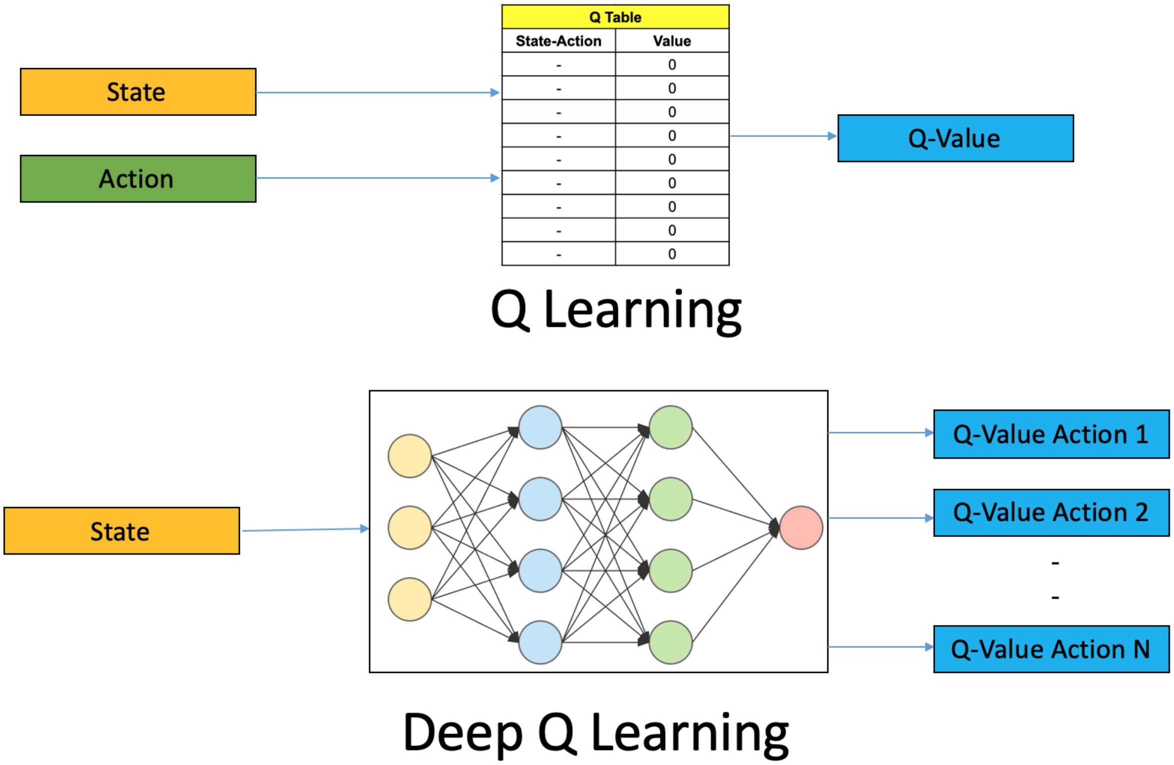

Figure 4.2: Two kinds of knowledge base.

illustrated in Figure 4.2. In this thesis, we consider that the knowledge base is made up of a table that stores the action value of each state-action pair. Initially, the action values for all state-action pairs are assigned arbitrary equal values (e.g., zeros). When the system visits a state for the first time, a random action will be selected since all the action values are equal. Then the environment will respond to the action, namely output the new state and immediate reward to the agent. The agent perceives the environmental response, updates the action value for that original state-action pair, and selects a new action based on the action values for the new state stored in the table. This completes one step in the iteration process. As this process repeats, the learning agent continues to improve its performance.

Let $Q^*(s,a)$ be the expected average adjusted value of taking action a at state s, and then continuing infinitely by choosing actions optimally. Then the value function $v(s)$ of state s when the initial action is also optimal can be written as $v(s) = \max_{a \in \mathcal{A}_s} Q^*(s,a)$. The average reward

Bellman optimality equation for the SMDP (3.1) can be rewritten in action value form as

$$Q^*(s,a) = r(s,a) - g^*\tau(s,a) + \sum_{j\in\mathcal{S}} P(j|s,a) \max_{b\in\mathcal{A}_j} Q^*(j,b) \quad \forall s \in \mathcal{S}. \tag{4.1}$$

Then we have $\pi^*(s) = \arg\max_{a\in\mathcal{A}_s} Q^*(s,a)$ as an optimal policy. The value of the state-action pair (s,a) visited at the mth decision epoch is updated iteratively by the following equation [11]:

$$Q_{m+1}(s,a) = Q_m(s,a) + \alpha_m \left\{ r(s,a,j) - g_m\tau(s,a,j) + [\max_{b\in\mathcal{A}_j} Q_m(j,b) - Q_m(s,a)] \right\} \tag{4.2}$$

where α_m is the learning rate of the mth decision epoch. $r(s,a,j)$ is the cumulative reward between two decision epochs whose states are s and j, respectively. The term g_m is the average reward until the mth decision epoch, and $\tau(s,a,j)$ is the actual sojourn time between those two decision epochs.

We update the action value with the rate of α_m (α_m should be a properly set small positive value to avoid learning path divergence) to the direction of the difference between the maximum action value of the new state and the original action value for that state-action pair at each iteration. As the process goes, the states continue to be revisited and consequently, the agent refines the action values and the corresponding decision making process. Finally, the algorithm will converge to the optimality.

For a random state $s \in \mathcal{S}$, the agent can choose an action a^* satisfying $a^* \in \arg\max_{a\in\mathcal{A}_s} Q_m(s,a)$. This is called greedy policy. However, when the agent has no knowledge of the system initially, choosing the greedy actions all the time means it will miss some potentially beneficial actions which have more future rewards. Therefore, sometimes, the agent chooses an action other than that suggested by the greedy policy. This is called exploration. This will help the system ensure that all possible states are visited. The most common exploration method is the ϵ-greedy exploration, where all actions are tried with non-zero probabilities. The agent chooses the greedy action with probability $1-\epsilon$ and chooses an action at random with probability ϵ. In this thesis, the learning rate α_m, the exploration probability ϵ_m and the average reward update β_m at the mth decision epoch are decayed slowly to 0 according to a Darken-Chang-Moody search-then-converge procedure [51] as follows

$$\Theta_m = \frac{\Theta_0}{1+u} \tag{4.3}$$

Algorithm 4 SMART

Initialization:

Let the count of decision epoch $m = 0$. For all $s \in \mathcal{S}$ and $a \in \mathcal{A}$, initialize action values $Q_0(s,a) = 0$. Set the cumulative reward $R = 0$, the total time $T = 0$ and the initial average reward $g_0 = 0$. The DCM scheme is used to select the learning rate α_0, α_ω, the exploration probability $\epsilon_0, \epsilon_\omega$ and the average reward updating rate β_0, β_ω. Select the initial state s and initial action a arbitrarily. Set the immediate reward $r(s,a,j) = 0$, the time interval $\tau(s,a,j) = 0$ and the learning time $t = 0$. Let the maximal learning time be T_{max}.

Execution:

while $t < T_{max}$ **do**
  Monitor the occurrence of events
  if a user's service request arrives **then**
    Use the DCM scheme to update α_m, ϵ_m and β_m.
    Update the immediate information $r(s,a,j)$ and $\tau(s,a,j)$.
    Choose a greedy action b^* with probability $1-\epsilon_m$, otherwise randomly choose an action from the set $\{\mathcal{A}_j/b^*\}$.
    Use (30) to update the new action value function $Q_{m+1}(s,a)$.
    If the greedy action as chosen at the mth decision epoch, update the following parameter: the cumulative reward $R = (1-\beta_m)R + \beta_m r(s,a,j)$, the total time $T = (1-\beta_m)T + \beta_m \tau(s,a,j)$ and the average reward $g_m = R/T$. Else, set $g_m = g_{m-1}$.
    Set the current state s to the new state j, the count of decision epoch $m = m+1$ and update the learning time t.
    Reset $r(s,a,j) = i(s,a)$ and $\tau(s,a,j) = 0$.
  else{a service completes}
    Update $r(s,a,j)$, $\tau(s,a,j)$ and t.
  end if
end while

where $u = \frac{m^2}{\Theta_\omega + m}$; Θ here can be substituted by α, ϵ and β for learning, exploration and updating respectively. Θ_0 and Θ_ω are constants that should be determined at the beginning.

4.3 Average Reward RL Algorithm

We apply a model-free average reward RL algorithm called SMART (Semi-Markov average reward technique) [11] to solve our SMDP in this section. The algorithm is presented in Algorithm 4. The agent continuously monitors the occurrence of the events. When the event of request arrival occurs, the agent uses ϵ-greedy policy selection criterion to select an action a according to the current state s and the tabular action values for all possible actions stored in the knowledge base. That action will take the system to a new state j. The action value $Q_{m+1}(s,a)$ for that state action pair (s,a) is updated based on the temporal information (the immediate reward and the time interval between those two decision epochs) and the previous action value stored in the knowledge base. When the event is the completion of a service, the agent doesn't need to take an action and update the action value. It only updates the temporal information and the learning time.

As the algorithm runs for many iterations, for every state, the action values of a smaller subset (one or more) of all the available actions become dominant. The algorithm ends when a clear trend appears, and the dominant actions constitute the decision policy vector.

4.4 Summary

In this chapter, we propose a model-free reinforcement learning method to solve the SMDP for channel allocation. Four elements of reinforcement learning model and the update equation are introduced in the working principle of the RL method. Then we apply an average reward RL algorithm to solve our SMDP. During the path of interacting with the environment, the agent keeps learning from the feedback. Therefore, compared with the model-free planning method, our model-free RL method may have a worse performance at the beginning with no knowledge of the environment. However, after taking some actions and learning from the feedback, the agent will have its own understanding of the environment and try to take the optimal action each time. At last, our RL method can acquire a similar performance to that of the model-based dynamic programming method.

Chapter 5

Performance Evaluation

5.1 Introduction

In this chapter, we evaluate the performance of our SMDP channel allocation scheme. First, we examine the convergence of the model-free RL method by testing the average rewards with fixed parameters versus time in Section 5.2. Then we change the arrival rates and the completion rates of two types of users to test how the average system reward varies with two kinds of rates in Section 5.3 and Section 5.4, respectively. In addition, we also evaluate the rejection probabilities of SUs' service requests versus the arrival rates and the completion rates of two types of users in Section 5.5. Finally, Section 5.6 compares different action probabilities of two types of users' requests.

To monitor the occurrence of different events, we use multiple event clocks for corresponding events. The results of our dynamic programming method and reinforcement learning method are compared with the Greedy Policy. The Greedy Policy is an algorithm that doesn't explore the environment to get more statistical information about the system and only considers the short-sighted maximal immediate reward. That is, when a user arrives, the service request will always be allocated to the maximum available channels.

In our first dynamic programming method, we made some assumptions to acquire the transition probabilities so that we can obtain an asymptotically optimal policy. Then we can use the policy to calculate the expected long-term average reward. The second reinforcement learning method is a way of learning an approximately optimal control policy. This is accomplished by assigning rewards and punishments for their actions based on the temporal feedback obtained during active interactions of the learning agent with the dynamic system.

In the simulation we set the maximum number of channels for one cognitive enabled RSU

Table 5.1
SIMULATION PARAMETERS

K	6	R_p	40
C	2	R_s	30
λ_p	2	R_t	8
λ_s	5	E_t	5
μ_p	2	U_t	4
μ_s	3	ϵ	10^{-10}

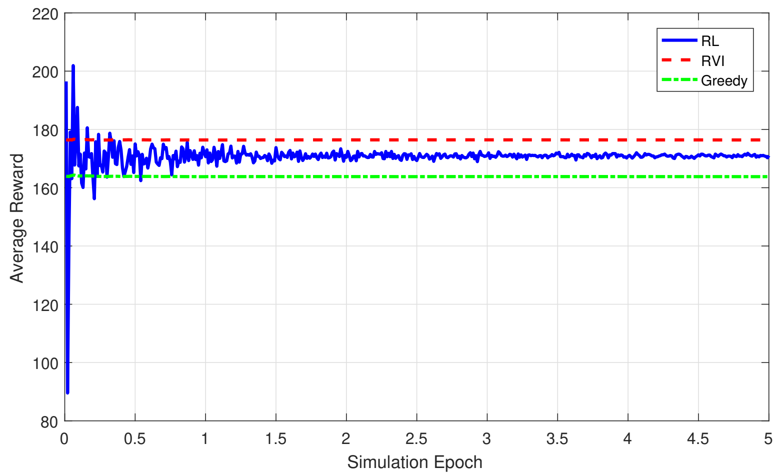

Figure 5.1: Average reward versus simulation epoch.

$K = 6$, the maximum number of channels for one service is $C = 2$. All other default simulation parameters are summarized in Table 5.1. The unit of the arrival rate λ is number of users per second. The constant ϵ in our relative value iteration algorithm is set to 10^{-10}. The constants used in the reinforcement learning algorithm for the learning rate, the exploration probability and the average reward update rate are set to $\alpha_0 = \epsilon_0 = \beta_0 = 10^{-1}$ and $\alpha_\omega = \epsilon_\omega = \beta_\omega = 10^{11}$, which are the common values according to reference [51]. The simulation is run by 10^6 unit times and it is repeated by 5 times to obtain the average performance except for the the case of convergence in Section 5.2.

5.2 Convergence of the Model-free RL Method

First, we test the average rewards with fixed user arrival rates and completion rates versus simulation epoch in Figure 5.1 to examine the convergence of our model-free RL method. Each simulation epoch is run by 10^6 unit times. Since the relative value iteration algorithm generates the optimal policy for each state with known transition probabilities and time intervals and then acts based on the policy, the average reward curve for RVI doesn't fluctuate much. The system has the similar average reward from the beginning to the end and the curve is basically a straight line. In reinforcement learning algorithm, the agent needs to interact with the environment and learn the optimal policy from the feedback. The average reward for RL takes a longer time to converge so the curve has significant fluctuations at the beginning, and then it stabilizes at a certain level. As for Greedy algorithm, it chooses actions based on the Greedy policy so that basically its curve has no fluctuation. We can see from the figure that our RL algorithm can get an approximately optimal average reward compared with the optimal value from RVI algorithm, 171.8 versus 176.4, and definitely better than the Greedy algorithm's value for 163.8.

5.3 Average System Reward versus Arrival Rates

Then we change the arrival rates of two types of users to test the performance of three algorithms in Figure 5.2. The sub-figure (a) of Figure 5.2 is the simulation result when PU arrival rate λ_p increases

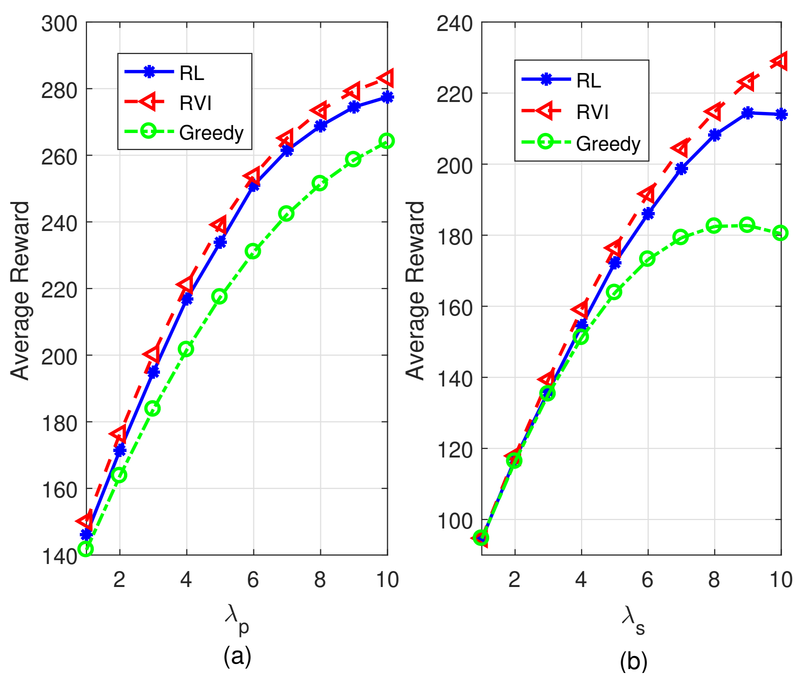

Figure 5.2: Average reward versus the arrival rates of (a) PUs' requests and (b) SUs' requests.

from 1 to 10, $\lambda_s = 5, \mu_p = 2$ and $\mu_s = 3$. The sub-figure (b) of Figure 5.2 shows the result when SU

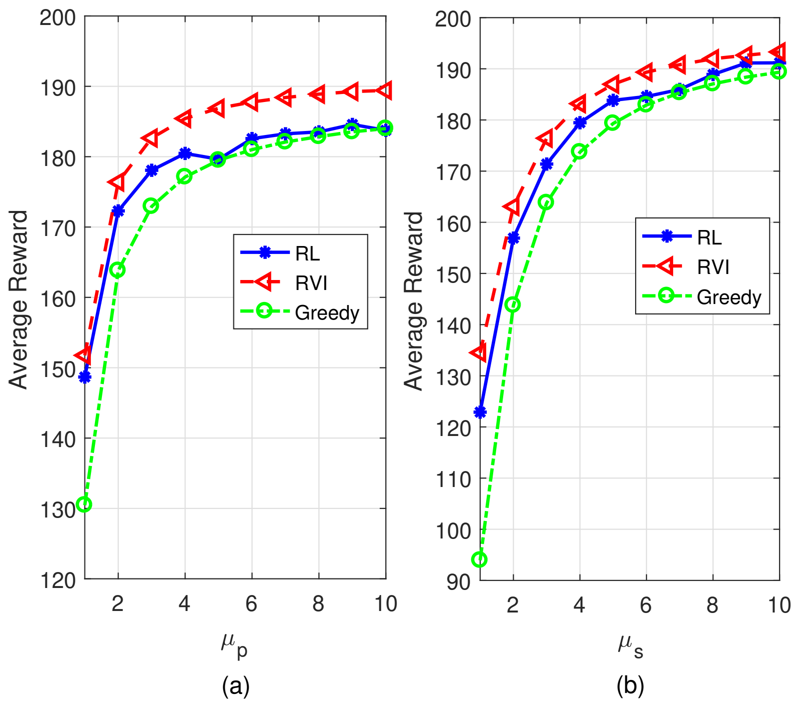

Figure 5.3: Average reward versus the completion rates of (a) PUs' requests and (b) SUs' requests.

arrival rate λ_s varies from 1 to 10 with $\lambda_p = 2, \mu_p = 2$ and $\mu_s = 3$. From the simulation results, the average reward increases as the service request rate increases. That is because the system hasn't reached its saturation under given arrival rates. However, the slope of the average reward curves decreases as the service request arrival rates increase. This is because that the opportunity of the new request being allocated with channel resources will decrease with the increase of arrival rates due to the limited number of channels in one RSU. The system is about to reach its saturation. There is a special case where the average reward decreases when the SUs' arrival rates are high for the greedy algorithm. This is because we set a negative reward for rejection of SUs' service requests. When the SUs' arrival rates are high and we still take the greedy policy to handle these requests, the rejection will happen frequently and the average reward will be decreased. We can see that the performance of the RL algorithm is close to that of the RVI algorithm and is better than that of the greedy algorithm.

5.4 Average System Reward versus Completion Rates

Figure 5.3 shows the average rewards versus the completion rates of two different service requests. From the simulation results, the average rewards increase with the increase of completion rates while the increasing speed gradually shrinks. And our RL algorithm outperforms the greedy algorithm when the completion rate of the PU is less than 5, while the greedy algorithm is almost as good as the RL algorithm when the completion rate of the PU exceeds 5. This is because we only update our Q-table when there is an arrival event. With high PU completion rate, there will be more completion events rather than arrival events so that the Q-table won't be updated frequently. Then the agent won't be trained enough. That is to say, the agent is not smart enough to make good choices. So, in that case, our RL algorithm won't be that much better than the greedy algorithm.

5.5 Rejection Probabilities of SUs' Service Requests

Figure 5.4 and Figure 5.5 present the rejection probabilities of SUs' service requests versus the arrival rates and completion rates of two types of users' requests. We can see from the simulation

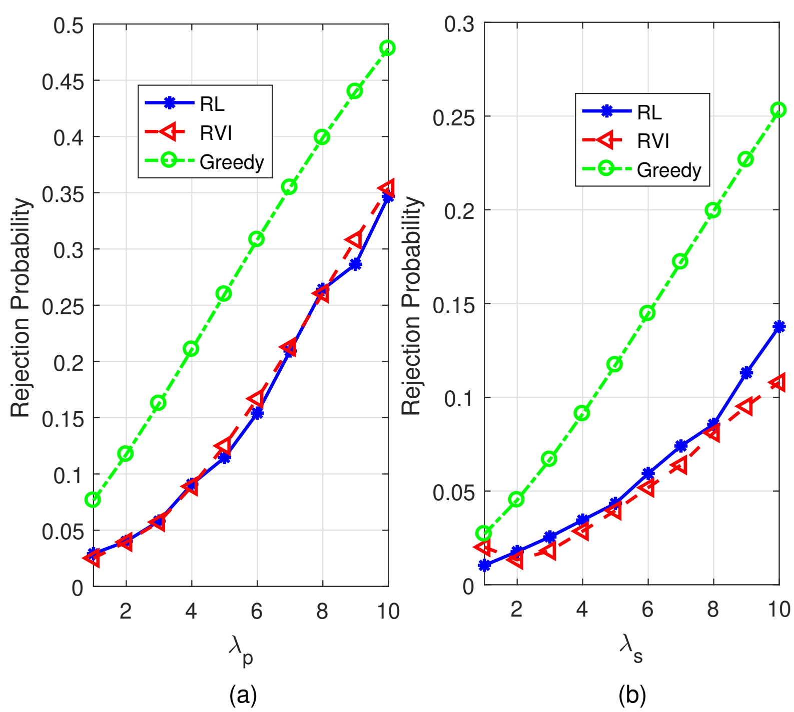

Figure 5.4: Rejection Probability of SUs' requests versus the arrival rates of (a) PUs' requests and (b) SUs' requests.

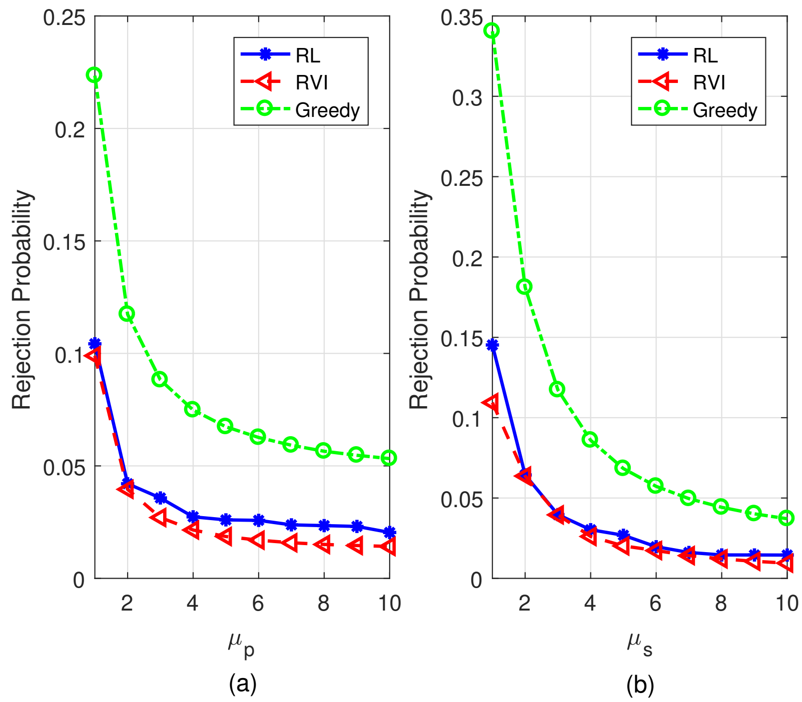

Figure 5.5: Rejection Probability of SUs' requests versus the completion rates of (a) PUs' requests and (b) SUs' requests.

results that the rejection probability increases as the arrival rates increase and decreases as the completion rates increase. And the rejection probability of our RL algorithm is pretty close to that of the RVI algorithm while the rejection probability of the greedy algorithm is quite high. This is because that both RL and RVI algorithms take future rewards into consideration so that they will save some empty channels for incoming users. However, the greedy policy only considers immediate reward and it will always allocate a maximum number of channels to the current user.

5.6 Action Probabilities

We compare different action probabilities of two types of users' requests in this section. We illustrate the action probabilities when the system event is the arrival of a primary user versus the arrival rates of primary users λ_p, the arrival rates of secondary users λ_s, and the completion rates of primary users μ_p in Figure 5.6, Figure 5.7 and Figure 5.8, respectively. The action probabilities when the system event is the arrival of a secondary user versus the arrival rates of primary users λ_p, the arrival rates of secondary users λ_s, and the completion rates of primary users μ_p are compared in Figure 5.9, Figure 5.10 and Figure 5.11, respectively. Since the system will always accept PUs'

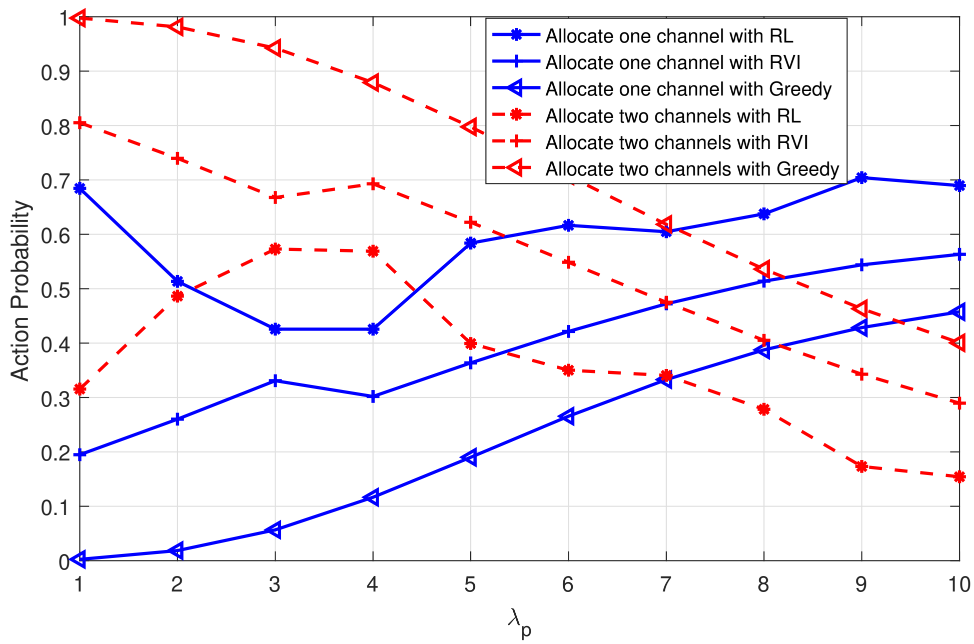

Figure 5.6: Action probabilities of PUs' request versus the arrival rates of PUs

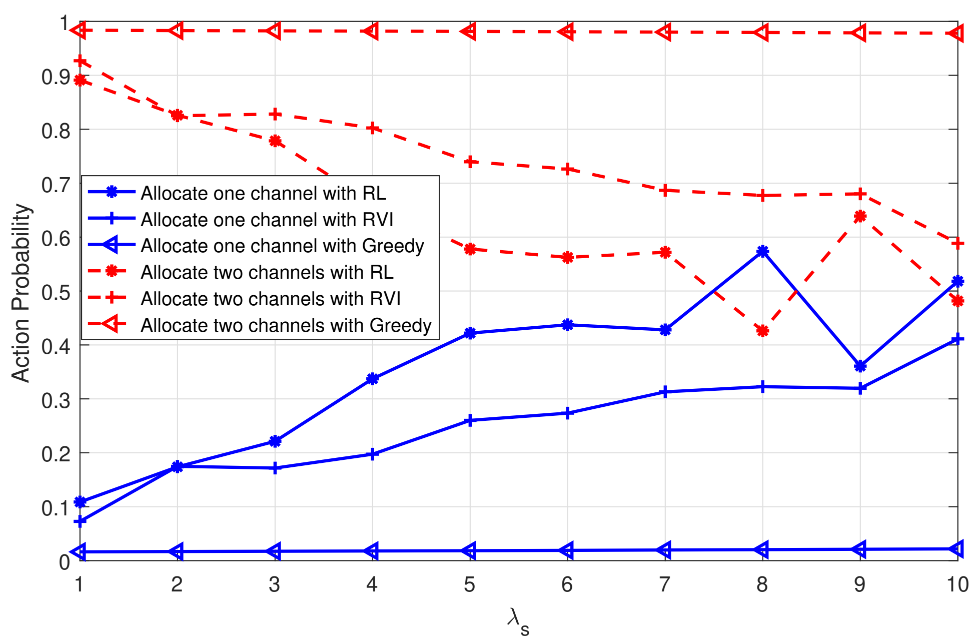

Figure 5.7: Action probabilities of PUs' request versus the arrival rates of SUs

requests, the rejection probability remains zero during the simulation. Therefore, in the first three figures, when adding the probabilities of accepting the PU's request with one channel and those with two channels with fixed parameter for each method, the sum should equal one. We can see

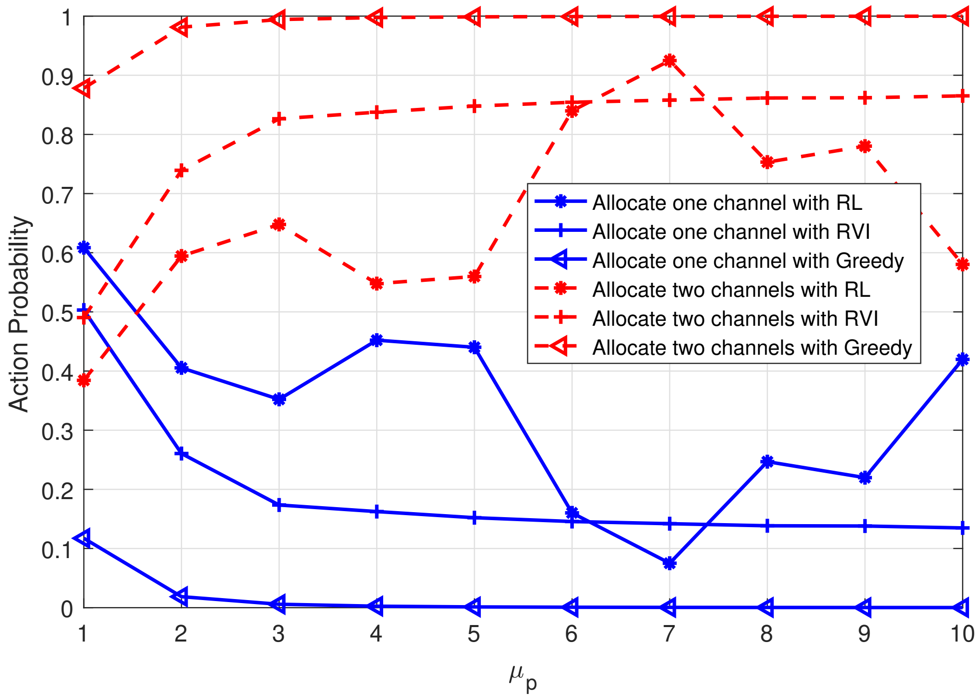

Figure 5.8: Action probabilities of PUs' request versus the completion rates of PUs

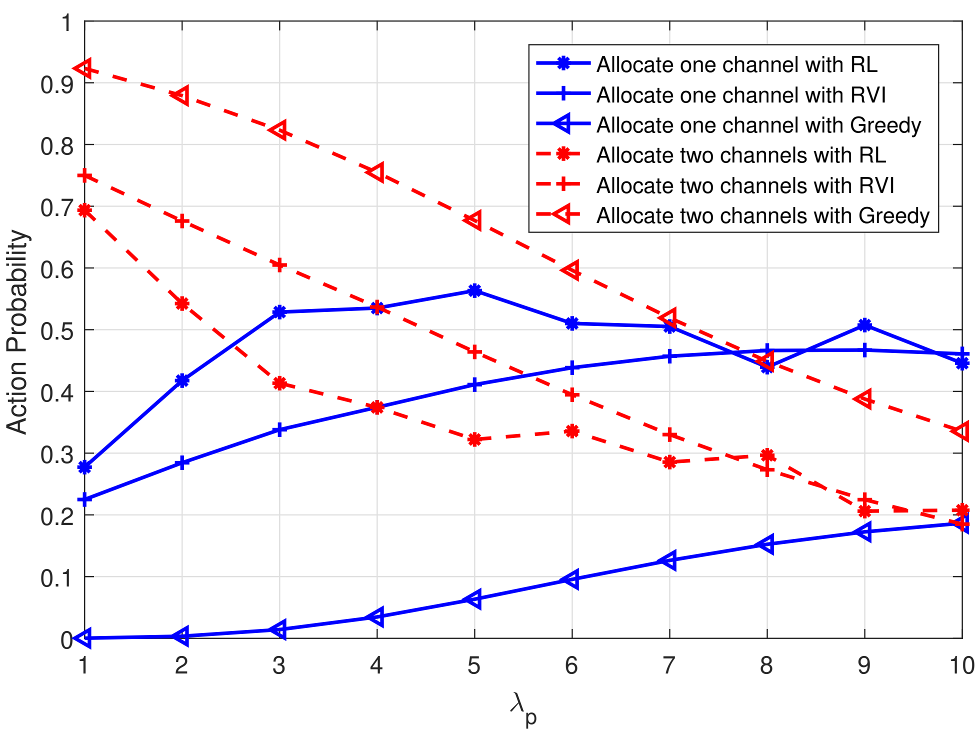

Figure 5.9: Action probabilities of SUs' request versus the arrival rates of PUs

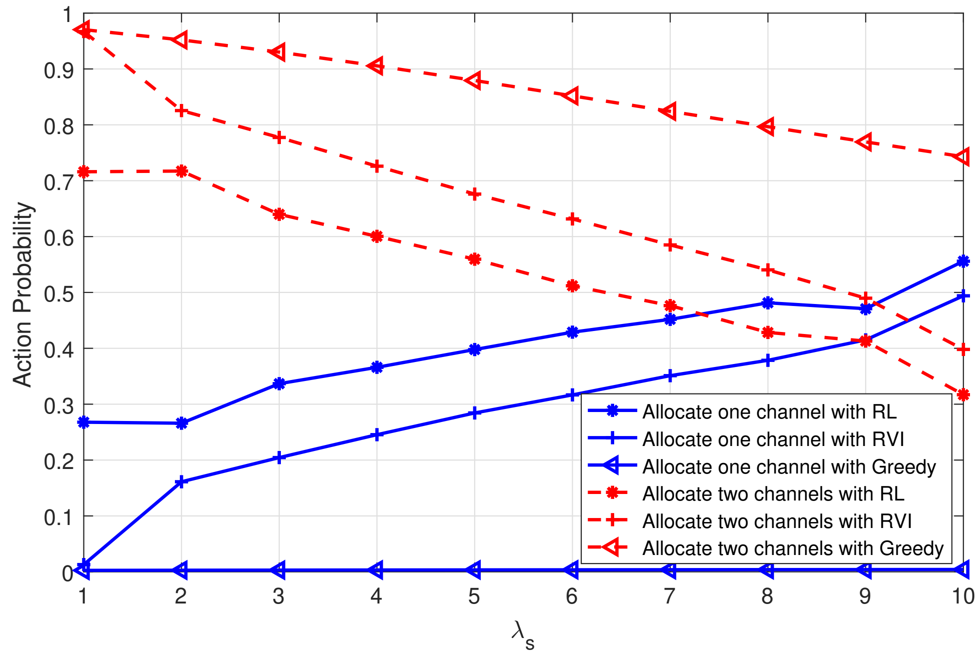

Figure 5.10: Action probabilities of SUs' request versus the arrival rates of SUs

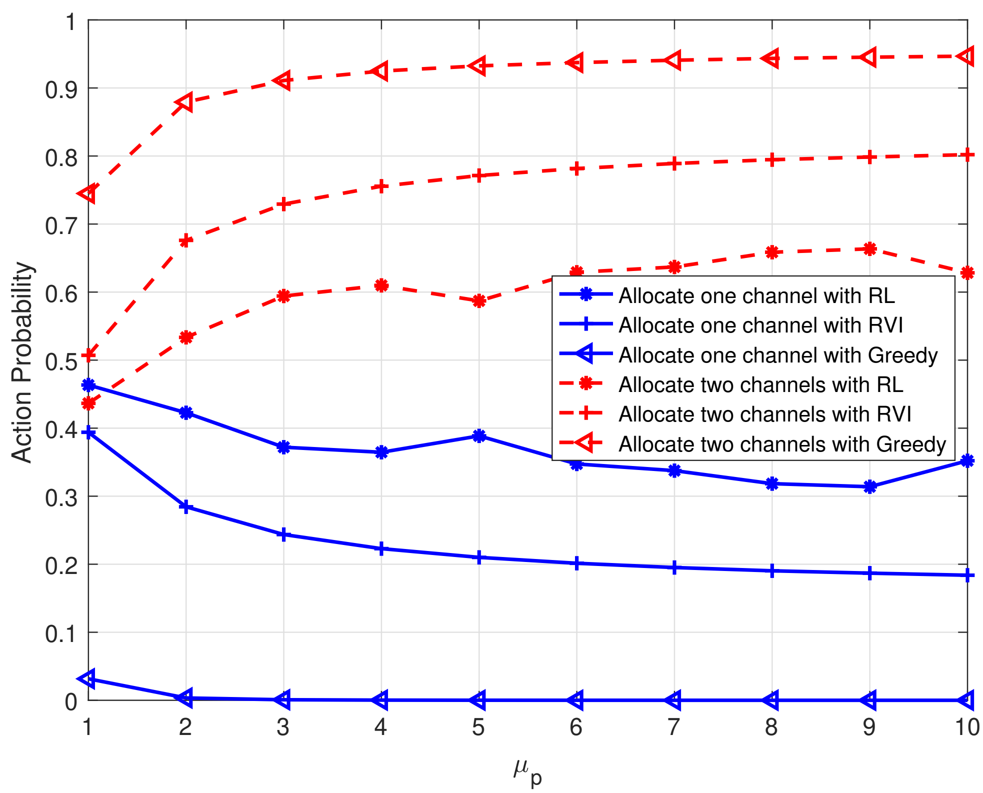

Figure 5.11: Action probabilities of SUs' request versus the completion rates of PUs

from the figures that the probabilities of two types of service requests being allocated with two channels decrease as the arrival rates increase while they increase as the completion rates increase. The probabilities of two types of service requests being allocated with one channel increase as the arrival rates increase while they decrease as the completion rates increase.

5.7 Summary

In this chapter, we evaluate the performance of the two methods. First, we introduce some parameters we use during the simulation. Then we examine the convergence of the model-free RL method, average system reward versus arrival rates, average system reward versus completion rates, rejection probabilities of SUs' service requests, and different action probabilities of two types of users' requests, respectively. Through extensive performance evaluation, we have demonstrated that our reinforcement learning method can acquire a similar performance to that of the dynamic programming while both outperform the greedy method.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, we proposed two RL-based channel allocation methods in a CR-VANET. We analyzed the SMDP model, made some assumptions and calculated the transition probabilities for the model-based dynamic programming method. The RVI algorithm was used to find an asymptotically optimal channel allocation policy. Due to the consideration of the validity of those assumptions in real life, we presented another RL algorithm to approximate the optimal channel allocation policy. The numerical results show that our model-based dynamic programming method can get better performance with a shorter time while our model-free RL method can converge to a more reliable performance in real-life scenarios.

6.2 Future Work

Many possible extensions could be further explored for the proposed channel allocation scheme. The connections and interference between two types of vehicle users are not considered in our proposed scheme. For future works, the influence of interference between SU and PU channels in the channel allocation process needs to be investigated in our model. Moreover, an alternative reward scheme needs to be studied considering diverse traffic environments, such as the direction and speed for vehicles.

Bibliography

[1] C. K. Toh, *Ad Hoc Mobile Wireless Networks: Protocols and Systems*. Prentice Hall, Dec. 2001.

[2] M. Z. Morteza and L. Hadi, "A Survey on Centralised and Distributed Clustering Routing Algorithms for WSNs," *IEEE Vehicular Technology Conference*, May 2015.

[3] J. Gao, M. S. Li, L. Zhao, and X. Shen, "Contention intensity based distributed coordination for V2V safety message broadcast," *IEEE Trans. Vehicular Tech.*, vol. 67, no. 8, pp. 7523–7537, Aug. 2018.

[4] K. Hafeez, A. Anpalagan, and L. Zhao, "Optimizing the control channel interval of the DSRC for vehicular safety applications," *IEEE Trans. Vehicular Tech.*, vol. 65, no. 5, pp. 3377–3388, May 2016.

[5] K. A. Hafeez, L. Zhao, B. Ma, and J. W. Mark, "Performance Analysis and Enhancement of the DSRC for VANET's Safety Applications," *IEEE Trans. Vehicular Tech.*, vol. 62, no. 7, pp. 3069–3083, Sep. 2013.

[6] H. Hartenstein and L. P. Laberteaux, "A tutorial survey on vehicular ad hoc networks," *IEEE Communications Magazine*, vol. 46, no. 6, pp. 164–171, Jun. 2008.

[7] J. Miltola, "Cognitive radio: Making software radios more personal," *IEEE Personal Communications Magazine*, vol. 6, no. 4, pp. 13–18, Aug. 1999.

[8] A. Alshamrani, X. S. Shen, and L. L. Xie, "QoS provisioning for heterogeneous services in cooperative cognitive radio networks," *IEEE J. Select. Areas Communications*, vol. 29, no. 4, pp. 819–830, Apr. 2011.

[9] Y. Zhuang, J. Pan, Y. Luo, and L. Cai, "Time and location-critical emergency message dissemination for vehicular ad-hoc networks," *IEEE J. Select. Areas Communications*, vol. 29, pp. 187–196, Jan. 2011.

[10] Y. Su, J. Gao, C. Yang, and L. Zhao, "SMDP-Based prioritized channel allocations in vehicular ad hoc networks," in *2019 IEEE Globecom Workshops (GC Wkshps): IEEE GLOBECOM Workshop on Artificial Intelligence for Next-Generations Wireless Communications (GC 2019 Workshop - AINGWC)*, Waikoloa, USA, Dec. 2019.

[11] T. K. Das, A. Gosavi, S. Mahadevan, and N. Marchalleck, "Solving semi-Markov decision problems using average reward reinforcement learning," *Manage. Sci.*, vol. 45, no. 4, pp. 560–574, Apr. 1999.

[12] W. Zhang, R. K. Mallik, and B. K. Letaief, "Cooperative spectrum sensing optimization in cognitive radio network," *in Proc. IEEE Intl. Conf. Communications*, pp. 262–266, May 2008.

[13] X. Y. Wang and P. H. Ho, "A novel sensing coordination framework for CR-VANETs," *IEEE Trans. Vehicular Tech.*, vol. 59, no. 4, pp. 1936–1948, May 2010.

[14] Y. Sun and K. Chowdhury, "Enabling emergency communication through a cognitive radio vehicular network," *IEEE Commun. Mag.*, vol. 52, no. 10, pp. 68–75, Oct. 2014.

[15] C. Jiang, Y. Chen, and K. J. R. Liu, "Data-driven optimal throughput analysis for route selection in cognitive vehicular networks," *IEEE J. Select. Areas Communications*, vol. 32, no. 11, pp. 2149–2162, Nov. 2014.

[16] T. Wang, L. Song, and Z. Han, "Coalitional graph games for popular content distribution in cognitive radio vanets," *IEEE Trans. Vehicular Tech.*, vol. 62, no. 8, pp. 4010–4019, Oct. 2013.

[17] N. Cordeschi, D. Amendola, and E. Baccarelli, "Reliable adaptive resource management for cognitive cloud vehicular networks," *IEEE Trans. Vehicular Tech.*, vol. 64, no. 6, pp. 1–10, May 2016.

[18] N. Cordeschi, "Distributed and adaptive resource management in cloud-assisted cognitive radio vehicular networks with hard reliability guarantees," *Vehicular Commun.*, vol. 2, no. 1, pp. 1–12, Aug. 2014.

[19] L. Zhang, T. Luo, W. Liu, S. Zhu, and J. Li, "Cooperative spectrum allocation with QoS support in cognitive cooperative vehicular ad hoc networks," *China Commun.*, vol. 11, no. 10, pp. 49–59, Oct. 2014.

[20] N. Zhang, H. Liang, N. Cheng, Y. Tang, J. W. Mark, and X. S. Shen, "Dynamic spectrum access in multi-channel cognitive radio networks," *IEEE J. Select. Areas Communications*, vol. 32, no. 11, pp. 2053–2064, Nov. 2014.

[21] T. Jiang, Z. Wang, L. Zhang, D. Qu, and Y. C. Liang, "Efficient spectrum utilization on TV band for cognitive radio based high speed vehicle network," *IEEE Trans. Wireless Communications*, vol. 13, no. 10, pp. 5319–5329, Oct. 2014.

[22] L. Sun, A. Huang, H. Shan, M. Xing, and L. Cai, "Quality-driven adaptive video streaming for cognitive vanets," *in Proc. IEEE VTC.*, pp. 1–6, Sep. 2014.

[23] N. Cheng, N. Zhang, N. Lu, X. Shen, J. W. Mark, and F. Liu, "Opportunistic spectrum access for CR-VANETs: A game-theoretic approach," *IEEE Trans. Vehicular Tech.*, vol. 63, no. 1, pp. 237–251, Jan. 2014.

[24] X. L. Huang, J. Wu, W. Li, Z. Zhang, F. Zhu, and M. Wu, "Historical spectrum sensing data mining for cognitive radio enabled vehicular ad-hoc networks," *IEEE Trans. Dependable Secure Comput.*, vol. 13, no. 1, pp. 59–70, Jan. 2016.

[25] M. H. Cheung, F. Hou, V. W. S. Wong, and J. Huang, "DORA: Dynamic optimal random access for vehicle-to-roadside communications," *IEEE J. Select. Areas Communications*, vol. 30, no. 4, pp. 792–803, May 2012.

[26] L. Sun, H. Shan, A. Huang, L. Cai, and H. He, "Channel allocation for adaptive video streaming in vehicular networks," *IEEE Trans. Vehicular Tech.*, vol. 66, no. 1, pp. 734–747, Jan. 2017.

[27] H. He, H. Shan, A. Huang, and L. Sun, "Resource allocation for video service in heterogeneous cognitive vehicular networks," *IEEE Trans. Vehicular Tech.*, vol. 65, no. 10, pp. 7917–7930, Oct. 2016.

[28] M. Li, L. Zhao, and H. Liang, "An SMDP-based prioritized channel allocation scheme in cognitive enabled vehicular ad hoc networks," *IEEE Trans. Vehicular Tech.*, vol. 66, no. 9, pp. 7925–7933, Sep. 2017.

[29] K. Zheng, H. Meng, P. Chatzimisios, L. Lei, and X. Shen, "An SMDP-based resource allocation in vehicular cloud computing systems," *IEEE Trans. Ind. Electron.*, vol. 62, no. 12, pp. 7920–7928, Dec. 2015.

[30] H. Liang, L. X. Cai, D. Huang, X. Shen, and D. Peng, "An SMDP-based service model for interdomain resource allocation in mobile cloud networks," *IEEE Trans. Vehicular Tech.*, vol. 61, no. 5, pp. 2222–2232, Jun. 2012.

[31] X. Xiong, L. Hou, K. Zheng, W. Xiang, M. S. Hossain, and S. M. M. Rahman, "SMDP-based radio resource allocation scheme in software-defined internet of things networks," *IEEE Syst. J.*, vol. 16, no. 20, pp. 7304–7314, Oct. 2016.

[32] L. Lei, Z. Zhong, K. Zheng, J. Chen, and H. Meng, "Challenges on wireless heterogeneous networks for mobile cloud computing," *IEEE Trans. Wireless Communications.*, vol. 20, no. 3, pp. 34–44, Jun. 2013.

[33] L. Lei, Y. Kuang, N. Cheng, X. Shen, Z. Zhong, and C. Lin, "Delay-optimal dynamic mode selection and resource allocation in device-to-device communications–Part II: Practical algorithm," *IEEE Trans. Vehicular Tech.*, vol. 65, no. 5, pp. 3491–3505, May 2016.

[34] A. Das, S. C. Ghosh, N. Das, and A. D. Barman, "Q-learning based co-operative spectrum mobility in cognitive radio networks," *IEEE 42nd Conf. Local Comput Netw.*, pp. 502–505, Oct. 2017.

[35] Q. Li, L. W. Zhao, J. Gao, H. Liang, L. Zhao, and X. Tang, "SMDP-based coordinated virtual machine allocations in cloud-fog computing systems," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1977–1988, Jun. 2018.

[36] E. Barrett, E. Howley, and J. Duggan, "Applying reinforcement learning towards automating resource allocation and application scalability in the cloud," *Concurrency Comput. Pract. Exp.*, vol. 25, no. 12, pp. 1656–1674, Aug. 2013.

[37] S. M. Ross, *Introduction to Probability Models*, 9th ed. New York: Elsevier, 2007.

[38] R. Bellman, "A markovian decision process," *Indiana Univ. Math. J.*, vol. 6, no. 4, pp. 679–684, 1957.

[39] X. P. Guo and O. Hernandez-Lerma, *Continuous-Time Markov Decision Processes*. Springer, 2009.

[40] W. S. Jewell, "Markov renewal programming i: Formulation, finite return models," *J. Oper. Res.*, vol. 11, pp. 938–948, 1963.

[41] J. S. D. Cani, "A dynamic programming algorithm for embedded markov chains when the planning horizon is at infinity," *Mgt. Sci.*, vol. 10, no. 4, pp. 716–733, 1964.

[42] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. MIT Press, 2001.

[43] M. I. Kamien and N. L. Schwartz, *Dynamic Optimization: The Calculus of Variations and Optimal Control in Economics and Management*. New York: Elsevier, 1991.

[44] C. G. Rommen, "The probability of load balancing success in a homogeneous network," *IEEE Trans. Softw. Eng.*, vol. 17, no. 9, pp. 922–933, Sep. 1991.

[45] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York, Wiley, 2005.

[46] P. Auer, T. Jaksch, and R. Ortner, "Near-optimal regret bounds for reinforcement learning," *Journal of Machine Learning Research*, vol. 11, pp. 1563–1600, 2010.

[47] D. P. Bertsekas and J. Tsitsiklis, *Neuro-Dynamic Programming*. Nashua, NH: Athena Scientific, Sep. 1996.

[48] D. P. Bertsekas, *Dynamic Programming and Optimal Control: Approximate Dynamic Programming, Vol.II*. Nashua, NH: Athena Scientific, 2012.

[49] V. Paxson and S. Floyd, "Wide area traffic: The failure of Poisson modeling," *IEEE/ACM Trans. Netw.*, vol. 3, no. 3, pp. 226–244, Jun. 1995.

[50] L. Busoniu, R. Babuska, B. D. Schutter, and D. Ernst, *Reinforcement Learning and Dynamic Programming using Function Approximators*. Taylor Francis CRC Press, Apr. 2010.

[51] C. Darken, J. Chang, and J. Moody, "Learning rate schedules for faster stochastic gradient search," *in Proc. IEEE Workshop Neural Netw. Signal Process.*, pp. 3–12, Aug. 1992.