Contrôle adaptatif des feux de signalisation dans les carrefours : modélisation du système de trafic dynamique et approches de résolution

Biao Yin

SPIM

Thèse de Doctorat

utbm
université de technologie
Belfort-Montbéliard

école doctorale **sciences pour l'ingénieur et microtechniques**

UNIVERSITÉ DE TECHNOLOGIE BELFORT-MONTBÉLIARD

Contrôle adaptatif des feux de signalisation dans les carrefours: modélisation du système de trafic dynamique et approches de résolution

BIAO YIN

IRTES
sciences & ingénierie
Institut de Recherche sur les Transports, l'Énergie et la Société

SPIM

Thèse de Doctorat

école doctorale **sciences pour l'ingénieur et microtechniques**

UNIVERSITÉ DE TECHNOLOGIE BELFORT-MONTBÉLIARD

N° | X | X | X

THÈSE présentée par

BIAO YIN

pour obtenir le

Grade de Docteur de
l'Université de Technologie de Belfort-Montbéliard

Spécialité : **Automatique**

Contrôle adaptatif des feux de signalisation dans les carrefours: modélisation du système de trafic dynamique et approches de résolution

Unité de Recherche :
Institut de Recherche sur les Transports, l'Énergie et la Société (IRTES)

Soutenue le 11 décembre 2015 devant le Jury :

PIERRE BORNE	Rapporteur	Professeur à l'Ecole Centrale de Lille
CHRISTIAN PRINS	Rapporteur	Professeur à l'Université de Technologie de Troyes
SAÏD HAYAT	Examinateur	Chargé de recherche HDR à l'IFSTTAR
MOHAMED BENREJEB	Examinateur	Professeur à l'École Nationale d'Ingénieurs de Tunis
ABDELLAH EL MOUDNI	Directeur de thèse	Professeur à l'Université de Technologie de Belfort-Montbéliard
MAHJOUB DRIDI	Co-Directeur	Maître de Conférence à l'Université de Technologie de Belfort-Montbéliard

ACKNOWLEDGEMENT

I would like to give my sincere gratitude to my supervisors Prof. Abdellah EL MOUDNI and Associate Prof. Mahjoub DRIDI, for the continuous support of my Ph.D study and research in Lab IRTES-SeT. Their patience, motivation, and immense knowledge, and their inspiring guidance during my thesis deserve the most appreciation and respect in my heart. It is really a great experience working with them. With their abundant research experience, they show me how to become an independent researcher.

I would further like to give my gratitude to the financial support from the program of China Scholarships Council (CSC). I would like also to thank to UTBM so that I could do my thesis in such comfortable and inspiring environment.

Finally, I wish to take this opportunity to express my appreciation and thanks to all my friends for their emotional supports and research help. Especially, they are Xinyi LIU, Xuguang HAO, Hongjian WANG, Jianxing LIU, Binying YE, Jiawei ZHU, Abderrahim CHARIETE, Haode LIU. I really cherish and engrave the friendship and all the beautiful memories with them. I would also like to express special thanks to my parents who provide me eternal love and supports. They encourage me to explore knowledge and teach me to be a man with responsibility. With all the love and faith, tomorrow is going to be better.

CONTENTS

Acknowledgement **i**

Contents **vi**

List of Figures **ix**

List of Tables **xi**

General introduction **1**

Chapter 1 Generalities of traffic signal control **5**

1.1 Introduction 5

1.2 Descriptions of traffic signal control systems 6

1.2.1 Principle concepts 6

1.2.2 Traffic signal control systems 8

1.2.2.1 Established systems 8

1.2.2.2 Developing systems 10

1.3 Dynamic traffic models 12

1.3.1 Intersection signal control models 12

1.3.2 Network loading models 13

1.4 Traffic signal control methods review 15

1.4.1 DP and search algorithm 15

1.4.2 Artificial intelligence methods 16
1.4.3 Reinforcement learning and ADP methods 20
1.5 Conclusion and objective of the thesis 24

Chapter 2 Dynamic traffic system modeling 27
2.1 Introduction 27
2.2 Traffic flow organization patterns 28
2.2.1 Fixed phase sequence 29
2.2.2 Variable phase sequence 31
2.2.3 Adaptive phase sequence 32
2.3 System modeling at isolated intersection 34
2.3.1 Markov Decision Process 34
2.3.2 Model definitions of characteristic in MDP 35
2.3.2.1 Model assumption 35
2.3.2.2 Model framework 36
2.4 System modeling at traffic network 39
2.4.1 Network loading model in micro-simulation system 39
2.4.1.1 Network representations 39
2.4.1.2 Vehicle-following model 41
2.4.2 Coordinated signal control model at network 46
2.4.2.1 Multiagent MDP 46
2.4.2.2 Tunable state control for coordination 47
2.5 Summary 50

Chapter 3 Methods studied based on dynamic programming 53
3.1 Introduction 53
3.2 Backward DP algorithm for control and analysis 54
3.2.1 DP introduction 54
3.2.2 Value iteration algorithm for stochastic states system 60
3.2.3 Case study and analysis 63
3.3 Forward search algorithm for control and analysis 65
3.3.1 Forward search A* introduction 65
3.3.2 Forward search algorithm for deterministic states system 69
3.3.3 Case study and analysis 79

3.4 Limitation of DP based algorithms 82
3.5 Summary 84

Chapter 4 Approximate dynamic programming with RLS-TD(λ) learning algorithm 87
4.1 Introduction 87
4.2 Structure of ADP 88
4.2.1 Neural networks approximation 89
4.2.2 linear function approximation 93
4.3 RLS-TD(λ) for linear function approximation 96
4.3.1 Multi-step temporal difference (TD(λ)) learning 96
4.3.2 RLS-TD(λ) 98
4.3.3 Learning with multi-step value iteration 101
4.4 Algorithm for adaptive traffic signal control 102
4.4.1 Algorithm for isolated intersection 102
4.4.2 Algorithm for traffic network 104
4.4.2.1 Independent network control 105
4.4.2.2 Coordinated network control 106
4.5 Summary 108

Chapter 5 Applications and results 111
5.1 Introduction 111
5.2 Application in isolated intersection 112
5.2.1 Preparation 112
5.2.2 Functional parameters simulation 116
5.2.3 Comparisons and analysis 120
5.2.3.1 Different phase mode solutions 120
5.2.3.2 Fine planning solution 122
5.3 Application in traffic network 126
5.3.1 Preparation 126
5.3.2 Vehicle-following simulation 130
5.3.3 Comparisons and analysis 133
5.3.3.1 Independent traffic network 133
5.3.3.2 Coordinated traffic network 133

5.4 Summary 140

Conclusion and perspectives **143**

Appendices **147**

LIST OF FIGURES

1.1 Traffic time space diagram 7
1.2 Types of control logics 8
1.3 The fundamental diagram of the LWR model ($q = VK$) 14

2.1 Illustration of a typical intersection with conflicts 29
2.2 Illustration of fixed phase sequence (FPS) mode 29
2.3 Illustration of variable phase sequence (VPS) mode 31
2.4 Matrix of adaptive phase sequence (APS) mode 32
2.5 Example of APS mode 33
2.6 5-intersection traffic network 40
2.7 Inside link 41
2.8 Cases in vehicle-following model: (a) Case 1, (b) Case 2, (c) Case 3, and (d) Case 4 44
2.9 Explanation of X_1, X_2 in the case that $n = 5$ ($n \in u$) and $n_d = 7, 2, 8$ ($d = -1, 0, 1$. $n_d \in v$) 50

3.1 Time series and stages 54
3.2 A sample of state transition in stochastic dynamic programming 56
3.3 Value iteration of dynamic programming 58
3.4 Policy iteration of dynamic programming 59
3.5 A simple two-phase intersection 60

3.6 Example of two-phase traffic signal planning 60
3.7 Traffic state transition model in decomposition MDP 62
3.8 Vehicle length comparisons in three control methods ($\rho = 0.6$) 65
3.9 An example of weighted directed graph 68
3.10 A* solution for shortest path problem 68
3.11 Illustration of phase decision under phase modes of (a) FPS (2-connectors), (b) VPS (4-connectors), and (c) APS (12-connectors) 70
3.12 Illustration of traffic signal control under decision tree 71
3.13 Explanation of different cases of the optimal positions (the dash-arrow in (a) means existing the chosen action to the same states, but for simplicity, we don't choose it for next planning any more; the dash-arrow with vertical bar in (b) means not existing the chosen action to the same state, but it may still reach to other state determined by the optimal action.) 74
3.14 Example of fixed phase sequence based on optimal position (assume initial $\Phi = 1$ and $\varpi = \hat{p}^*(\tau) \mathrm{mod}_2$) 76
3.15 Rolling horizon approach 77
3.16 Comparisons of queue evolutions and signal sequences among the Optimal FC, Q-learning(VPS), and FSDP-R(VPS) 81

4.1 The schematic diagram of the ADHDP framework (the solid lines represent signal flow, while the dashed lines are the paths for parameter tuning) 90
4.2 The structure of the action network and critic network 91
4.3 Linear feature-based architecture 94
4.4 Example of system action transition 108

5.1 Traffic Scenario C-flow profile on average arrival rates during the simulation 114
5.2 Evolutions of functional parameters by using ADP with RLS-TD(λ) learning (in Traffic Scenario A-2, APS mode with λ=0, n=1, 2, 6, and 7) . . . 117
5.3 Comparisons of functional parameters by using ADP method between RLS-TD(λ) and TD(λ) learning (in Traffic Scenario B-3, FPS mode with $\lambda = 0$, n=1) 118

5.4 Comparisons of functional parameters by using TD(λ) learning between 2-s solution and 0.5-s solution (in Traffic Scenario B-3, VPS mode with $\lambda = 0$, n=2) 118
5.5 Comparisons of functional parameters by using ADP method between RLS-TD(λ) and TD(λ) learning (in Traffic Scenario C, APS mode with n=1) 120
5.6 Comparisons of average delays by using ADP method between RLS-TD(λ) and TD(λ) learning in APS 121
5.7 Improvements of average delays by using different methods in FPS, VPS, and APS, comparing with the Haijema-MDP in FC 123
5.8 Evolutions of queue length by using ADP_RLS-TD(λ) ($\lambda = 0$ in APS at 0.5-s and 2-s solutions in Traffic Scenario B-3) 124
5.9 Evolutions of average queue length by using ADP_RLS-TD(λ) ($\lambda = 0$ in Traffic Scenario B-3) 125
5.10 Traffic demand for network, except the inside link lanes represented by short gray bars, (a) D-1: low; (b) D-2: medium; (c) D-3: high 128
5.11 Behavior of vehicle movements on lane (60 places) during samples of 100 steps 130
5.12 Examples of the relationship between vehicle position (X-axis) and velocity (Z-axis) in 3-Dimension 131
5.13 Evolutions of vehicle position on lane, ADP for (a), (b), (c), and FC for (d) 132
5.14 Comparing results by different tunable e 135
5.15 Comparing results of average queue length (mean of 100 sample intervals) at intersection I_3 in Traffic Scenario D-2 137
5.16 Comparing results of total average queue length at network using FC, SOA, Q-learning (VPS) and ADP_RLS-TD(λ) ($\lambda = 0$, with $e = 0.95$ in VPS and $e = 1$ in APS) methods in Traffic Scenario D-2 138
5.17 Comparing results of total average queue length at network using SOA, Q-learning (VPS) and ADP_RLS-TD(λ) ($\lambda = 0$, with $e = 0.90$ in VPS and $e = 1$ in APS) methods in Traffic Scenario D-3 138
5.18 Comparing results of current vehicle average speed using SOA, Q-learning (VPS) and ADP_RLS-TD(λ) ($\lambda = 0$, with $e = 0.90$ in VPS and $e = 1$ in APS) methods in Traffic Scenario D-3 139

LIST OF TABLES

1.1 Summary of design programs for traffic signal control 10

3.1 Simulation results of average traffic delays 64
3.2 Computation of convergence in value iteration algorithm for solving MDP 65
3.3 Phase numbered in FPS, VPS, and APS 69
3.4 Traffic scenarios of asymmetric and symmetric average flow rates 79
3.5 Results of average traffic delay (s) in Scenario A and B 80
3.6 Comparisons of run time (s) 80

5.1 Intersection system parameter settings 113
5.2 Asymmetric and symmetric average arrival rates for intersection 113
5.3 Results of average delay in asymmetric rates 122
5.4 Results of average delay in symmetric rates 122
5.5 Results of average delay in fine solution 123
5.6 Comparisons of run time for isolated intersection 126
5.7 Network system parameter settings 127
5.8 Methods comparing on average delay (s) and improvements (%) 136
5.9 Comparisons of run time for network 140

GENERAL INTRODUCTION

In urban area, traffic congestion has been a crucial problem for people's daily lives and environment. It results in excess delays, reduced safety and increased environmental pollution [1]. Therefore, making efforts for solving the congestion problem is necessary, especially for increasing traffic demand today. An efficient traffic signal control method at intersection is required urgently when too many vehicles attempt to use a common transportation infrastructure with limited capacity. As development of intelligent transportation system (ITS), the research on intelligent traffic signal control is one of the most important subject.

The term of intelligent or smart traffic signal control is that signal controllers can truly think for themselves. That is to say, the controller implemented intelligent algorithm adapts itself to traffic environment, using received traffic information. Of course, besides the controller, some existing technologies for traffic detection and information communication are also included in intelligent traffic system. These parts have a rich literature and practical study because of the importance. However, in the thesis, we mainly focus on the modeling and algorithm study for controller decision making at signalized intersections. And, the traffic information based on the level of entity is generated by simulation, which is declared to be a convenient and qualified way to investigate the control mechanism.

In traffic control field, the studies of intelligent and adaptive traffic signal control in real-time are very popular. Many researchers endeavor to develop online optimal algorithms in high quality and efficiency. In general, three issues must be addressed in

formulating an online optimal control problem [2]: (1) development of a dynamic mathematical model that represents the current, or expected traffic condition of the controlled system; (2) specification of the real-time control objective; (3) design of an appropriate optimization technique such that the controlled system meets the specified criteria.

Mathematical models that correspond to signalized intersections can be classified into two generalized categories. One refers to macroscopic models and the other one refers to microscopic models. The former focuses on the fundamental relationships between speed, flow, and density of traffic flow movements, such as kinematic wave models. Whereas microscopic control models concentrate on the behaviors of individual vehicle or vehicle queue length in discrete-time system. It is worth noting that, in recent years, the rise of new technologies and smart vehicles pushes for the studies of autonomous or connected vehicle control, as well as the related signal or autonomous management at intersections [3, 4]. It could be seen that the behavior of individual vehicle plays an important role in this trend. Unlike traffic flow theory study in a macroscopic way, this trend makes a microscopic perspective on vehicle with properties of position, speed, and direction. Each vehicle can be viewed as an agent, which can communicate with its surrounding. Evidently, an intersection is an agent too. Thus, traffic signal control model at intersections based on the framework of discrete time Markov Decision Process (MDP) attracts much attention, on account of its facile model framework for agent-based learning techniques [5, 6].

On the other hand, a specified objective of traffic signal problem can be well defined in model construction, and it usually corresponds to traffic delay. For easy implementation, traffic delay is often replaced by calculating queue length on lane. In dynamic planning system, such as the MDP modeled one, traffic state represented by queue length can be easily obtained after each time step. Considering the conditions of queue length calculation, control models often refer to the types of deterministic or stochastic, steady state or time-dependent. It is known that traffic arrivals are in random distribution. Fortunately, actual arrival data can be detected and communicated by some techniques. With limited information, some deterministic control models can also be well operated.

As for the appropriate optimization methods for traffic signal control, exact algorithm and near-optimal algorithm are two important parts. In dynamic control system, dynamic programming (DP) offers an exact optimal solution for this multistage decision making problem. Some well-known adaptive traffic signal control systems are based on the con-

cepts of DP. However, the limitation of DP is the "curse of dimensionality" caused by a large state space. The computational burden makes a constraint on DP applied to complex or fine planning optimization problems. Thus, much research is concentrated on the learning algorithm that supports a near-optimal solution with computation efficiency. This way is guaranteed to be more efficient and can be also designed online learning to adapt to the environment. Besides traditional artificial intelligence methods, reinforcement learning (RL) and approximate dynamic programming (ADP) are very popular and attract much attention in past decade. In practice, RL and ADP are preliminarily well applied to traffic signal control fields and have many advantages to real-time adaptive control, which interests us most in the thesis work. More details and reviews about the related methods can be seen in Chapter 1.

Plan of the thesis

In the thesis, there are total five chapters as follows.

Chapter 1 introduces the state-of-the-art about traffic signal control systems, models, and methods. After that, we conclude it and set up the objective of the thesis, which aims to make a real-time adaptive traffic signal control in a distributed traffic network system.

Chapter 2 presents the model study of the system. At first, traffic flow organization patterns make up the control rules of signal phase, and firstly propose to use adaptive phase sequence (APS) mode. In the following parts, the modeling for intersection and network using MDP is introduced in detail. A new vehicle-following model for network loading and tunable state control for network coordination are two original points.

Chapter 3 and Chapter 4 are the studies of control methods. In Chapter 3, exact DP and related search algorithms are investigated at isolated intersection. A backup DP algorithm in steady-state stochastic problem is presented by using value iteration. A forward search algorithm based on A* is proposed under deterministic state transition. By the limitation of DP in practical case study, it is suggested to use an approximate optimal technique, especially for APS mode and the whole network. In Chapter 4, to overcome some shortcomings of DP algorithms, we try to use the ADP method for real-time adaptive traffic signal control both for the isolated intersection and the network. In particular, the recursive least-squares temporal difference (RLS-TD(λ)) learning for linear function

approximation is adopted. The related theory and proposed algorithms are emphasized on this chapter.

Finally, in Chapter 5, we do experiments in simulation to evaluate the proposed ADP with RLS-TD(λ) algorithm comparing with other approaches. Results of performance measures are illustrated and analyzed.

CHAPTER 1

GENERALITIES OF TRAFFIC SIGNAL CONTROL

1.1 Introduction

In this chapter, we introduce the state-of-the-art about traffic signal control systems, models, and methods. After that, we make some conclusions and present the objective of the thesis.

The reviews of three parts including systems, models, and methods are presented in this chapter, especially the control methods and the related work that are served to the thesis. In the first part, introduction of traffic signal control systems is given. We mainly present the characteristics of existing systems and compare the systems in different properties. As new technologies applied to traffic signal control field, some developing systems based on artificial intelligence and autonomous or connected vehicle techniques are very popular. For evaluation of system performance and the intention to rebuild analogous traffic environment in reality, in the second part, we focus on traffic signal control modeling and traffic network loading method, which contribute to the requirements of the investigation of traffic signal control algorithm. More importantly, Reviews of traffic signal control methods are emphatically presented in the last part, especially reinforcement

learning and approximate dynamic programming. With further insight into the control mechanisms, we drive basic ideas of adaptive traffic signal control algorithm in the thesis.

1.2 Descriptions of traffic signal control systems

1.2.1 Principle concepts

To recognize traffic signal control system, some terminologies and types of signal control logic will be firstly introduced. The definitions of some key terminologies of traffic signal control are described as follows [7, 8].

Phase: those green, change, and clearance intervals in a cycle assigned to any independent movement(s) of traffic. A phase may be timed considering complex criteria for determination of sequence and the duration of intervals.

Phase sequence: a predetermined order in which the phases of a cycle occur.

Phase split: the fraction of the cycle time that is allocated to each phase for a set of traffic movements. It includes the green split, yellow, and red clearance interval.

Offset: the time difference between the start of green phases at adjacent intersections. Offset is used for a continuous traffic movement at successive intersections that may give rise to a "green wave" along an arterial.

Cycle: a complete sequence of signal indications. Cycle time is the total time for a signal to complete one cycle.

Minimum (Maximum) Green: a parameter that defines the minimum (maximum) allowable duration of the green display.

Inter-green: the period between the end of the green display duration and the start of green display for the following phase, also called as all-red interval or red clearance interval.

Saturation flow rate: the equivalent hourly rate at which previously queued vehicles can traverse an intersection approach under prevailing conditions, assuming that the green signal is available at all times and no lost times are experienced.

Isolated intersection: an intersection located outside the influence of and not coordinated with other signalized intersections.

Coordinated intersections: at least two adjacent intersections sharing traffic information and making collaborative decisions among them in a global view.

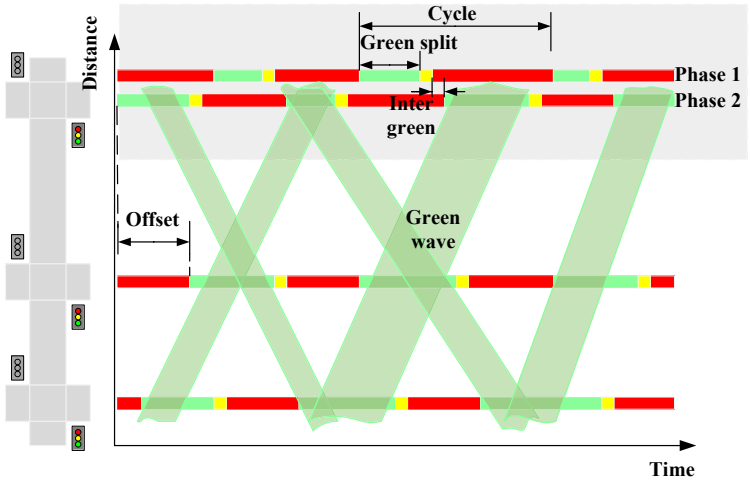


Figure 1.1: Traffic time space diagram

There are three types of traffic signal controllers, namely pre-timed, actuated, and adaptive controller. The control logics are given as follows.

Pre-timed (fixed-time) control: a signal control in which the cycle length, phase plan, and phase times are predetermined and fixed. Pre-timed (or fixed-time) controller, such as TRANSYT[9], applied historical data to determine appropriate time for traffic signals. Fixed-time controllers are best suitable for intersections where traffic volumes are predictable, stable, and fairly constant. It cannot handle unexpected conditions in traffic.

Actuated control: a type of signal control where time for each phase is at least partially controlled by detector actuations. Actuated control uses demand-responsive logic to set signal timing based on traffic demand as registered by detectors on upstream approaches. The common feature of actuated control is the ability to extend the length of green interval for a particular phase which changes the cycle length and phase split. MOVA [10] is the sample of actuated traffic control system.

Adaptive control: a real-time signal timing control which seeks continuous optimal system performance in response to variations based on measured and predicted traffic demands. It can change more parameters than just interval length in actuated control. Adaptive logic responds to traffic demand in real-time, realizing the adjustment of state parameters such as traffic volume, stop times, delay, and queue length. Additionally, it can change phase sequence and the allocation of cycle time with various phases of adjacent intersections to make them cooperative. Adaptive traffic control systems are becoming more widespread, such as SCAT[11], SCOOT[12], OPAC[13].

In fact, we cannot say which one of the control methods is good or not, especially the software package has been a successful application in real life. They are appropriate to different traffic environment. Meanwhile, the released versions are developed by improving the control mechanism progressively. However, the adaptive traffic signal control system is very popular nowadays and owns many advantages.

The applications of control logics in different control scopes can be seen in Fig. 1.2. In our study, we mainly focus on adaptive isolated intersection control and adaptive network control.

		Control Scope		
		Isolated Intersection	Arterial Coordination	Network Control
Control Logic	Pre-timed	×	×	×
	Actuated	×	×	×
	Adaptive	⊗	×	⊗

Figure 1.2: Types of control logics

1.2.2 Traffic signal control systems

1.2.2.1 Established systems

Several well-known packages of traffic signal control system are briefly introduced as follows.

TRANSYT (Traffic Network Study Tool [9]) is a software package for offline optimum fixed-time traffic signal timings. TRANSYT has two main elements. One is the traffic model which is used to calculate the performance index for a given set of signal timings. The other one is an optimizing process that makes changes to the settings and determines whether they improve the performance index or not. Because of TRANSYT's international appropriateness TRANSYT is now one of the most widely used signal timing programs in the world. It has continued to be developed by research institutes ever since its first release. For example, it is performed using a combination of a Cell Trans-

mission Model (CTM) and a Platoon Dispersion Model (PDM). The most recent developments (introduced in TRANSYT 14.1) include the addition of a traffic assignment model, various GUI improvements.

MOVA (Microprocessor Optimised Vehicle Actuation [10]) generates signal timings cycle-by-cycle for isolated intersection. MOVA uses vehicle gap detected from pairs of upstream detectors to determine green extension. The criterion for extension is whether the gap reaches certain critical values. The system typically uses the actuated control logic.

SCATS (Sydney Coordinated Adaptive Traffic System [11]) and SCOOT (Split Cycle and Offset Optimasation Tool [12]) are two well-known and widely used as the coordinated centralized systems. They are basically online variants of offline optimisation signal plan. The online capability then enables the selection of the most appropriate plan from the library according to detected traffic, adjusts offsets between adjacent intersections to facilitate traffic flow, and makes small adjustments to the signal plan.

UTOPIA (Urban Traffic Optimisation by Integrated Automation [14]) is a hybrid control system that combines online and offline optimisation. The system is constructed in a hierarchy with an area level and a local level. The area controller generates reference plan, and local controllers adapt this reference plan and dynamically coordinate signals in adjacent intersections. UTOPIA offers unmatched performance, especially in congested and unpredictable traffic conditions.

DYPIC (Dynamic Programmed Intersection [15]), PRODYN ([16]), and OPAC (Optimised Policies for Adaptive Control [13]) are developed based on the dynamic programming (DP) approach or related optimization schemes. DYPIC uses a backward DP and PRODYN optimizes timings via a forward DP. OPAC makes a distributed strategy featuring a dynamic optimization algorithm and has progressed through four versions. As for a practical issue, a rolling horizon approach is all used to allow the optimization to take advantage of the most recent predictions and observations. This rolling approach implies that: firstly, a planning horizon is split into a ‘head’ period with detected traffic information and a ‘tail’ period with predicted traffic information; secondly, an optimal policy is calculated for the entire horizon, but is only implemented for the ‘head’ period; finally, when the next time step arrives and new information becomes available, the process rolls forward and repeats itself.

RHODES (Real-time Hierarchical Optimized Distributed Effective System [17]) also uses a DP based algorithm. It has an architecture in three levels. From the highest level to the lowest one, they refer to dynamic network loading model, network flow control, and the intersection control, respectively. RHODES does not set timing plans in terms of cycle times, splits, and offsets, but rather in terms of phases duration for any given phase sequence. Additionally, the PREDICT algorithm is well designed in RHODES. Therefore, the emphasis shifts from changing timing parameters in reacting to traffic conditions just observed to pro-actively setting phase duration for predicted traffic conditions [18]. RHODES appears to take advantage of the natural stochastic variations in traffic flow.

In summary, see Table 1.1, we make some comparisons of the related traffic signal control systems on the areas of system design programs [19].

Table 1.1: Summary of design programs for traffic signal control

Program	Decision on signal settings	Signal profile	Signal coordination	Performance measures	Organization	Origin country
TRANSYT	Splits, offsets	Cyclic	Offset optimization	Stops, delay	Centralized/ Offline	UK
MOVA	Green extension or not	N/A	Nil	Stops, delay and capacity	Decentralized/ Online	UK
SCATS	Predetermined signal plan selection	Cyclic	Offset optimization	Capacity	Centralized/ Online	Australia
SCOOT	Adjustment of signal timing increments, offsets	Cyclic	Offset optimization	Stops, delay and congestion	Centralized/ Online	UK
UTOPIA	Green start times, durations and offsets	Cyclic	Offset optimization	Stops, delay	Centralized/ Online	Italy
OPAC	Change of current signal settings rolling forward	Acyclic	A virtual cycle length, offset	Stops, delay	Decentralized Online	USA
PRODYN	Change of current signal settings	Acyclic	Possible	Total delay	Decentralized/ Online	France
DYPIC	Complete signal settings	Acyclic	Nil	Delay	Decentralized/ Offline	UK
RHODES	Change of phase duration and sequence	Acyclic	Bandwidth based on platoons	Stops, delay	Decentralized/ Online	USA

1.2.2.2 Developing systems

Although the commercial systems are implemented widely in real world, some research on this field has been continued to develop more intelligent and autonomous traffic control systems.

Intelligent traffic signal control system

Intelligent traffic signal control system is that the system combines existing technology with artificial intelligence to create traffic signal timings. For example, it makes the use of sensor networks along with embedded technology to receive the information about the position, speed, and direction of vehicles. After that, intelligent traffic signal control algorithm is programmed to make decisions in real-time to adapt to certain traffic conditions. Artificial intelligence algorithms are normally used to make signals decisions which can change the traffic conditions to avoid congestion wherever possible. It attracts much attention in the past decades. A detailed review about this domain can be seen in this chapter, Section 1.4.

Autonomous traffic control system

Intelligent vehicle technology is progressing very rapidly and recent advances suggest that autonomous vehicle navigation will be possible in the near future.

Recently, autonomous traffic management system at un-signalized intersection interests many researchers [4, 20, 21, 22]. An early typical example is the research of K. Dresner and P. Stone [23]. They proposed a reservation-based system for alleviating traffic congestion, specifically at intersections, and under the assumption that the vehicles are controlled by agents. The research figured out that the reservation-based approach drastically outperforms the traffic light system. They extended their work in [3], which suggested an alternative mechanism for coordinating the movement of autonomous vehicles through intersections.

Actually, an important factor for improving traffic control efficiency is that autonomous traffic control system provides a two-way wireless communication environment enabling vehicle-to-infrastructure (V2I) [21] and vehicle-to-vehicle (V2V) [24] communications. Moreover, the cooperation among vehicles associated to Cooperative Adaptive Cruise Control system [25], is designed to optimally manipulate vehicles maneuvers based on nearby vehicles conditions. In [22], authors present that the connected vehicles can pass through the intersection with 99% and 33% of stop delay and total travel time reductions, respectively, comparing with the conventional actuated intersection control.

However, many challenges in this field need to be overcome in the future work, such as safety, faithful communication, and priority.

1.3 Dynamic traffic models

Dynamic traffic models mainly refer to two aspects. One is dynamic traffic modeling and the other one is dynamic traffic assignment associated with route choice model and traffic network loading model. On the scale of the thesis, we focus on dynamic traffic modeling at intersections and traffic network loading model in a simulation environment.

1.3.1 Intersection signal control models

In conventional traffic signal control system, the intersection model represented in a mathematic way is often modeled by using static data for optimization. The optimal signal timings are calculated by empirical formula. A classical method is using Webster's method for fixed-time signal control. Subsequently, dividing all day history data into various traffic condition periods, pre-defined signal timings are set, according to the dynamic traffic data in different periods. As the development of detected technologies, it is convenient to receive the limited traffic arrival information in real-time. Traffic signal control based on the dynamic traffic information is possible and proved to be more efficient.

Here, two kinds of dynamic control models at intersections are mentioned. One is the macroscopic model and the other one is the microscopic model. The former focuses on the fundamental relationships between speed, flow, and density of traffic flow movements controlled by the conventional phases with split and offset settings. The fundamental diagram is normally implemented for traffic network control. In the works of Lo et al. [26, 27], authors proposed the cell-based traffic dynamics representations for traffic signal control formulation, which automatically adjusts to the changing traffic conditions. In [28], the traffic flow process is modeled and the constraint problem of network-wide signal control is formulated as a quadratic-programming one that aims at minimizing and balancing the link queues so as to minimize the risk of queue spillback.

Whereas the microscopic control model concentrates on the behaviors of individual vehicle or vehicle queue length in a discrete-time system. The various phase duration and sequence may be considered for variant traffic conditions. The intersections are coordinated for the consideration of network equilibrium as well. Recently, traffic signal control model based on the framework of discrete time Markov Decision Process (MDP) [29] attracts much attention [19, 30, 31]. An MDP is characterized by a set of states, actions, reward function, and state transition function. For traffic signal control, the states

can be defined by queue lengths and signal status; the actions are the available control strategies for each state; the reward function defines the immediate reward of each action under a specific state; and the state transition function defines the probabilities for the system to shift from one state to another given the current state and action taken. The MDP traffic model can be solved via DP and reinforcement learning. Sometimes, in order to reduce state space, traffic density is classified as low, medium, or high level [30]. It is more popular to use reinforcement learning and agent-based techniques to achieve an efficient solution of traffic signal control problem formulated by MDP. Reviews of the related literature can be seen in Section 1.4.3.

1.3.2 Network loading models

To seek a solution of traffic network control, an important way is to simulate behaviors of traffic flow or vehicles at network. There are two common kinds of network loading models, namely macroscopic model and microscopic (micro-simulation) model, the same classification for the intersection signal control models mentioned above. Even combining these two models, some research works on the so-called "mesoscopic" approach for traffic simulation [32, 33] and it will not be discussed here.

In macroscopic models, the earliest dynamic network loading methods are mainly based on the kinematic wave model [34, 35] also frequently referred to LWR model (see Fig. 1.3) and subsequently developed by many researchers. These models assume that traffic behaves like an incompressible fluid and they are space-continuous. Limitation was found that hydrodynamic analogy is available only for high traffic densities. Other macroscopic models such as cell transmission model [36, 37] and other space-discrete models [38] for flow propagation are implemented. Actually, they are not suitable enough for the real-time adaptive control by using learning optimization approach because of a complex mathematic model and control variables.

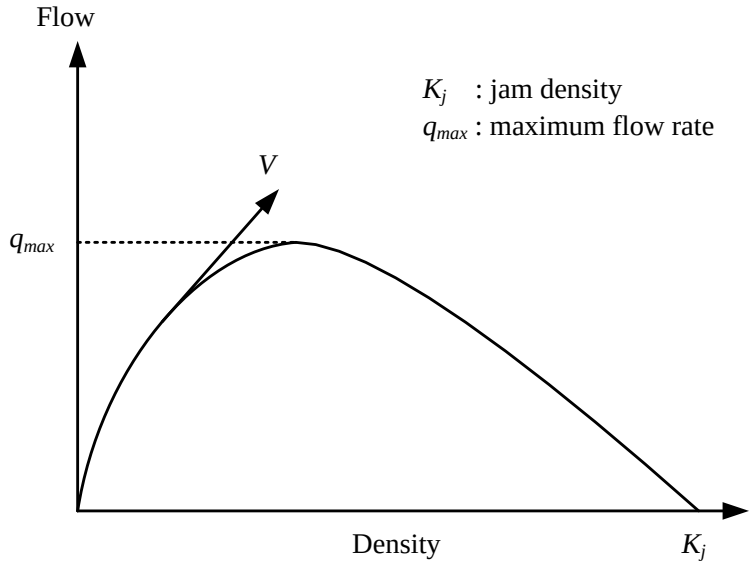


Figure 1.3: The fundamental diagram of the LWR model ($q = VK$)

The micro-simulation model can provide the traffic flows composed of individual vehicles in one network loading step. There are many popular micro-simulation models developed in universities and industries, such as PARAMICS [39], MITSIM [40], and VISSIM [41]. These models use some basic approaches, which make vehicles to move on network in equal small time intervals. In Nagel-Schreckenberg (NaSch) model [42], some basic rules of vehicle movement are proposed for traffic freeway based on cellular automata (CA) theory. In [43], a new CA-based approach is developed for traffic mobility model in urban area. Research has shown that CA-based model can yield realistic behavior [44, 45]. However, in [46], it indicates that the successful use of micro-simulation is commonly limited to relatively small size networks. The application for large network may lead to high computation time. We believe that it is easy to overcome this challenge as the development of computation techniques and embedded devices in the future. Moreover, the small size network subsystem in a distributed system, as well as the precise planning in discrete-time procedure for adaptive traffic signal control, is well suitable for the study based on a micro-simulation model [47, 48].

1.4 Traffic signal control methods review

In past years, traffic signal timing and optimization has been researched in a wide range of approaches. This section reviews sorts of these approaches, which refer to intelligent traffic signal controllers.

1.4.1 DP and search algorithm

The exact algorithms such as dynamic programming (DP) [49], search algorithm [50], are traditional optimization methods.

DP method is capable of solving multistage decision making problems. It decomposes a complex problem into a series of sub-problems with discrete time steps between them, often using backward search algorithms to obtain a global optimum policy. Some well-known adaptive traffic signal control systems based on the concepts of DP are widely applied, such as OPAC, PRODYN, and RHODES. They all uniformly recognized the importance of DP in solving sequential decision making for multistage systems. Although, DP for these complex systems is not directly used because of its weakness in computation. Sometimes, DP needs to connect heuristic techniques or simplify the state variables defining the real-word problem in some assumptions. There is some literature related DP for traffic control problems. In [51], DP with forward recursion is employed directly to derive the green time for each phase with objective of traffic delay reductions. In [52], the forward DP algorithm is used to calculate the shortest path problem about the optimal traffic control decisions, according to the total released time of intelligent vehicles. In [53], an "intelligent" traffic signal control algorithm is proposed based on the combination of DP and neural networks. DP is used to find the optimal green times for all the approaches at isolated intersection. But it assumes that the future vehicle arrival pattern is known.

A search algorithm is an algorithm for finding an item with specified properties among a collection of items. Some forward research algorithms, such as branch and bound, A* algorithm are used to find the optimal policy under the decision tree. In [54, 55], author uses branch and bound algorithm to solve autonomous intersection management problems via V2I communications. The traffic performance is significantly improved by comparing with other traffic signal control techniques. In [56], a forward search method using A* algorithm is proposed for real-time adaptive traffic signal control at isolated intersection.

A rolling forward approach is applied, considering the limited future information to make the short term optimal planning of variable signal phase duration and sequence.

Actually, DP algorithm as well as some search algorithms, is hard and impractical to apply in global optimization of traffic signal control problems by some limitations. For example, the computation burden and the incomplete information are limited for optimization when the high-dimension of problem and the limitation of available detected information have to be considered. The related studies about these are investigated and discussed in Chapter 3.

1.4.2 Artificial intelligence methods

The use of artificial intelligence (AI) methods to control traffic signals started in 1990's. In some research, it is found that traffic signal planning models usually involve the simultaneous optimization of phasing sequence, split, cycle length, and offset. It is often difficult to find the globally optimal solutions to such models within a reasonable amount of time using exact algorithms [57]. AI methods have been extensively researched as an alternative to exact algorithms to address this issue. Multiple optimization and estimation methods, such as evolutionary algorithm, fuzzy logic, neural networks, have been well applied for adaptive traffic signal control. In recent years, reinforcement learning [58] and agent-based control, are very popular with the ability to control unpredictable traffic condition issues, which will be more detailed in Section 1.4.3.

Evolutionary algorithms

Evolutionary algorithms use mechanisms inspired by biological evolution, such as reproduction, mutation, recombination, and selection. The common types of evolutionary algorithms include genetic algorithm (GA), particle swarm optimization (PSO), and ant colony optimization (ACO), etc. They often perform well approximating solutions to all types of problems, and also play an important role in traffic signal control field.

GA is very popular for traffic signal timing and optimization [59, 60, 61] in past years. Advances in the optimization of fixed-time traffic signal timings have provided evidence of GA optimization of traffic network performance evaluated via simulator [62]. For example, GA is implemented by calling TRANSYT traffic model for optimization of traffic control parameters (i.e., cycle length, green split, offset, and phase sequence). In a

typical research [59], Park et al. proposed a near-optimization traffic signal timing plan for oversaturated conditions generated by GA optimizer on the basis of a fitness value obtained from the mesoscopic simulator. In [61], GA is used to optimize the traffic signal timing with performance improved at road network. The proposed system can solve the equilibrium network design problem, by integrating the GA, traffic assignment, and traffic control. On the other hand, in [63], an acyclic adaptive traffic signal control system using real-time genetic optimization was proposed, with components of a genetic optimizer, a database manager, and an internal traffic simulator for fitness evaluation.

The metaheuristic PSO was proposed by Kennedy and Eberhart [64]. PSO has not been widely used for solving traffic problems, but it is a very promising technique which is capable to solve complex traffic problems [65]. Wei et al. applied the PSO to fine tune the parameters of a fuzzy-logic traffic signal controller and found that it can effectively improve the performance of the original fuzzy logic controller [66]. Garcia-Nieto proposed a PSO approach to find successful cycle programs of traffic lights, using a microscopic traffic simulator [67, 68].

Another interesting evolutionary method for traffic signal timing is ACO. In the research of Putha et al. [69], ACO is used to solve oversaturated network traffic signal coordination problem. It demonstrates that ACO is consistently more effective for a larger number of trials and to provide more reliable solutions than GA, traditionally employed to solve oversaturated conditions. The authors further pointed out that the structure of the ACO algorithm makes it particularly suitable for parallel computing, which can substantially shorten the computation time.

Techniques from evolutionary algorithms applied to the modeling of biological evolution are generally limited to explorations of micro-evolutionary processes and planning models based upon cellular processes. This optimization approach requires a large amount of simulation to determine the performance of the proposed signal plans and may require re-optimization with changing traffic conditions. As the network size increases, the cost of simulation increases, as does the population/generation size. With sufficiently large networks, then, real-time control may be impossible as the computation may not complete in a suitable amount of time [70]. Thus, the applications of evolutionary algorithms in traffic signal control have to date been primarily for small-scale or offline optimization problems. With continued advancements in computational technology and new developments in evolutionary algorithms, it is expected that more applications will

emerge, applying evolutionary algorithms for large-scale or real-time traffic signal optimization and coordination in the future [57].

Fuzzy logic

Numerous studies of traffic signal control have been developed based on fuzzy logic [71, 72, 73, 74, 75, 76]. The majority of existing fuzzy logic traffic control studies use queue lengths (e.g., short, medium, and long) and traffic arrivals (e.g., low, medium, and high) as the input to set fuzzy rules, and the control action usually is to either extend or terminate the current green phase.

Some previous studies ignore the traffic left-turn movement, or only consider two phases for each intersection in order to defining easier fuzzy rules [71, 72]. More realistic multiple-phase control are studied in [73, 74]. In [73], authors use a two-stage fuzzy logic to design the traffic signal timing plan for an isolated intersection. In the first stage, the observed approaching traffic flows are used to estimate relative traffic intensities. These traffic intensities are then used in the second stage to determine whether the current signal phase should be extended or terminated. In both studies [73, 74], fuzzy logic is used to decide whether to extend the current green phase or not. The phasing sequence optimization is not explicitly considered in their works.

Obviously, it is desirable to consider phase sequence optimization for better control performance. Motivated by this view, Murat and Gedizlioglu [75] proposed a Fuzzy Logic Multi-phased Signal Control (FLMuSiC) model for isolated signalised intersections. FLMuSiC consists of two modules. One is the signal time controller to arrange phase green splits and the other is the phase sequencer to schedule phase sequences using traffic queue lengths.

Lee et al. [77] designed a more complicate fuzzy control method to adjust phase sequences and splits for coordination between intersections. Recently, fuzzy logic with combination of multiagent learning technique is implemented for traffic networks [76, 78]. In [76], Choy et al. implemented cooperative, hierarchical, multiagent system for the real-time traffic signal control of a complex traffic network. The subproblem of the distributed system is handled by an intelligent agent with fuzzy neural decision making (FNDM) module. An example of a rule in the FNDM is given as follows:
IF{(*overall aggregate occupancy* is high) and (*overall aggregate flow* is high) and (*overall*

aggregate rate of change of traffic volume is high)}
THEN{(*traffic loading* is high) and (*level of cooperation needed* is high)}

Using fuzzy logic for traffic signal control has two advantages. One is that the implementation cost of fuzzy controllers is low. So fuzzy systems have attracted increased attention for traffic signal control. The other one is that a priori expert knowledge of objects can be easily reflected in well-designed fuzzy rules, which make it simpler and more intuitive to construct a fuzzy controller of traffic signal control [79].

Neural networks

Neural networks (NNs) have a potential capacity for traffic signal control problem, especially integrated with other AI methods, such as fuzzy logic [80, 81], multi-agent control [82]. The fuzzy NNs integrates and coordinates objectives and activities of agents hierarchical architecture in the traffic network, see [83].

Actually, Bingham [84] designed a two-phased single intersection fuzzy controller that is formulated by an NN and constructed an additional critic NN to optimize the controller, providing an early basic idea of fuzzy NNs and reinforcement learning in traffic signal control. There are several limitations of the approach adopted in [84] as reported by authors. Firstly, the neural learning is not effective under certain circumstances due to the lack of stochastic exploration. Secondly, the time needed to adjust the membership functions is too long. Finally, it is not known if the fuzzy-NNs implemented in [84] can yield good performance in a more complex traffic network. In [80], Choy et al. extented this work and proposed a hybrid NN model of a real-world traffic network to seek the solution of the limitations mentioned above. A multistage online learning process has been introduced and implemented in the hybrid NN model. The performance of the hybrid NN model indicated the efficacy in solving large-scale traffic signal control problem in a distributed way.

In the similar related works of [80], the research in [82] and [83] by Srinivasan et al. presented hybrid NNs model to solve real-time traffic signal control problem based on independent and cooperative multi-agent system, respectively. The weight update algorithm for the hybrid NNs model is performed at various stages, each involving tuning the weight parameters, learning rate, and the neural connection in response to the changes in the environment. This indicates that the hybrid NN-based multiagent system is able to adjust its weights parameters effectively throughout the duration of the simulation, so that

the signal plans it generates can accommodate the periodic as well as random fluctuations of traffic volumes.

1.4.3 Reinforcement learning and ADP methods

Classical DP algorithm has some drawbacks in high dimension problem. As for search algorithm, it is difficult to find appropriate heuristic information. Machine learning techniques can support a near-optimal solution by using an approximation in learning method on policy search. The related theory research is developed in recent years. Such as reinforcement learning (RL) [58, 85], approximate dynamic programming (ADP) [86, 87], have been already established for solving difficult multi-stage decision problems in the fields of operation research, computer science, and robotics, etc.

Introduction of RL and ADP

Reinforcement learning, one of machine learning methods, attracts much attention, especially the Q-learning. RL method is essentially to solve the problem formulated by the fundamental framework of Markov Decision Process (MDP). On the other hand, an agent-based control is quite well combined with RL method [88]. Similar to neural networks, agents need to be trained by RL algorithms before they can actually be used. Different from neural networks, the training of an agent is a dynamic process based on the continuous interactions between the agent and environment (unsupervised), not a fixed set of paired input-output training samples (supervised). It is well-known that RL can optimize system behavior by interacting with the environment and learning from the feedback. Thus, RL is beneficial to create adaptive controller, which is able to process unpredictable traffic conditions. The multi-agent by using RL can online learn from the environment independently or share the information between agents in a coordinated way. These advantages attract the considerable attention in the application of a distributed traffic network control system.

The formulation of ADP is firstly proposed by Werbos [89] and developed by many researchers. The related literature about the development of ADP can be seen in [90]. Several synonyms of ADP are used in literature. The representative related works are like the “Neuro-Dynamic Programming” by Bertsekas and Tsitsiklis [86], “Heuristic Dynamic Programming” by Si et al.[91]. The main idea of ADP is to use a structure of approxima-

tion function to estimate the cost-to-go value function in Bellman's equation. So, it can effectively avoid the "curse of dimensionality" caused by large state space in the recursive calculation of Bellman's equation. In ADP, it is possible to step forward in time to calculate objective function of current state, unlike classical DP algorithm which requires that we loop over all possible states. Essentially, ADP works in a similar way of RL. Differently, ADP operates as a model-based RL with function approximation [92]. That is to say, ADP optimizes the user-defined cost function conditioned on prior knowledge of the system and its state, while RL maximizes performance by the way of exploration and exploitation about the environment and does not require any a priori knowledge.

Application of RL and ADP

For online and multistage decision making, RL and ADP have the advantages on computation and learning techniques to solve complex system problems. Many researchers focus on RL and ADP approaches in applications of traffic signal control fields [5, 19, 93, 94, 95, 96]. Two perspectives of the application in this field using RL and ADP are figured out. One is the learning process of function approximation and the other one is the point of view in agent-based coordination. We thereby select a representative set of approaches that allow us to get insight into the state-of-the-art.

Learning process: The learning process in RL and ADP includes two parts: approximate structure and learning technique. Generally, with regard to different approximations, such as tabular Q-value, function approximation, several kinds of learning techniques are used for the update process. Some important ones are like gradient descent [97], temporal difference (TD(λ)) [98], least-squares TD(λ) [99, 100], and kernel based [101].

In [6, 102, 103], Q-learning method is adopted to update the cooperative multiagent, according to the best-response of Q-value at next state. The likelihood of Q-value is evaluated by using the count of visit states. This is a tabular Q-value representation method. However, in complex environments, it takes a long simulation period for the RL agent to visit each state action pair infinitely often to ensure convergence.

Thus, a technique called function approximation is described in RL. In ADP, the function approximation is normally applied. The explicit tabular representations of each state-action pair are not required. Instead, it is possible to generalize across different state-action pairs using the estimated values, which are defined by using a set of tunable

parameters [104]. Three typical function approximation methods with the corresponding learning techniques are reviewed as follows.

In [93, 94, 105], the authors proposed RL with function approximation for traffic network control, where neural networks are trained to provide approximation to the state-action value function. Action and critic neural networks are adopted. The learning parameters of these networks use gradient descent method or single-step TD. In the study of [19], it suggests that a simple linear approximation is sufficient for online operation. Because non-linear functions for exploring complex approximation may not prove cost-effective. Therefore, the linear function approximation of ADP based on feature-extraction function are successfully applied at isolated intersection, using the TD learning and perturbation learning to update the parameters. In [95], underlying RL, the linear approximation with the features like time elapsed and queue lengths are used in different traffic network scenarios. The parameter update process adopts the rule of gradient descent.

Another function approximation is the tile coding method [106]. In tile coding, the receptive fields of the features are grouped into partitions called tilings in which each element is called a tile, and the overall number of features that are present at one time is strictly controlled and independent of the input state [58]. Pham et al. [107] presented the RL traffic signal control system based on SARSA using tile coding. Each SARSA agent is completely independent, and the tile coding is used only as a method of approximating the value function for the local agent states. By contrast, Abdoos et al. [108] presented hierarchical control of traffic signal system considering the coordination between agents, using Q-learning combined with the tile coding approximation.

Coordinated multi-agent RL: One of the most significant early works of multi-agent RL (MARL) for traffic signal control is that of Wiering [47]. Wiering developed a model-based approach and found that the RL systems clearly outperformed fixed-time controllers at high levels of network saturation, when testing on a simple 3 x 2 grid network. Another interesting aspect of this research is that a type of co-learning is implemented; value functions are learned by signal controllers and driver agents, and the drivers also learn to compute policies that allow them to select optimal routes through the network. Much research has extended the work of Wiering and has a successful application for traffic network control system [48, 102, 109, 110].

As individual agents at local intersections do not coordinate their behaviors, recently, more and more research has been done on the coordination of multi-agent for traffic network control [6, 109, 111, 112, 113, 114, 115, 116]. The coordination mechanisms mainly focus on the agent hierarchical architecture[113, 114], learning in group games[6, 115], and coordination graphs[116, 117].

In [113], a multi-agent system based on a hierarchical architecture is proposed to achieve a balance between the local and global aspects of an urban traffic system. Local Traffic Agents (LTAs) and Coordinator Traffic Agents (CTAs) make up the fundamental levels of the hierarchy, in which the LTAs meet the needs of the specific intersection, and the CTAs determine if the chosen patterns of an LTA are suited to meet any global concerns. In addition, a solitary Global Traffic Agent (GTA) may exist for networks of sufficient size, and an Information Traffic Agent (ITA) provides a central location for the storage of all shared information within the system. It shows that the system efficiently managed the network in traffic accident and morning rush hour scenarios.

In Bazzan's work [109], the coordination of game theory is used for MARL. This is a simple stage game for synchronization of traffic signals. Interactions are modeled as coordination games where the highest reward is given when neighbor traffic signals coordinate their actions so that they synchronize their green phases. His extended work in [115], presents a supervised learning with three stage games by the approach, which is to have agents divided into groups that are then supervised by further agents. In the work of El-Tantawy and Abdulhai [6, 102], authors deal with the dimensionality problem by utilizing the principle of locality of interaction among agents, and the modular Q-learning technique based on agent groups. The former principle means that each agent communicates only with its immediate neighbors, while the latter allows partitioning of the state space into partial state spaces consisting of only two agents. This approach significantly reduces the complexity of the problem, while still producing promising results. The results presented in [102] are very encouraging, and this work is one of the largest and most realistic simulation tests of an RL traffic signal control approach to date, due to the use of a real urban network, along with real world traffic data and signal timings.

Kuyer et al. [116] developed a coordinated model-based MARL traffic signal control system using the Max-Plus algorithm [118] as a coordination strategy. Max-Plus algorithm is used to approximate the optimal joint action by means of message passing between connected agents in the coordination graph. The experimental result outperforms

the comparing studies of Wiering [47]. Similar work is in [117], where the MARL is combined with an implementation of the max-plus algorithm to control the traffic signals in a network with congested conditions.

Challenges of applying RL and ADP

According to the above reviews of RL and ADP applications in traffic signal control field, we conclude two main challenges as follows.

In order to design an online adaptive traffic signal control system, the computation efficiency is required. With respect to model framework of RL and ADP, the "curse of dimensionality" is frequently encountered, especially in the case when problem complexity increases vastly in larger road networks. Beside the use of model-free RL to reduce complexity, approximation method is usually used to tackle these difficulties both for RL and ADP. Of course, it is not difficult to deal with the challenge of computation in the future when available computational power is increased. Currently, true challenges are how we can approximate value function effectively and how to pick up strategies even knowing value function [119].

Another significant challenge is the coordination implementation and the information sharing between agents. A control policy selected by a local agent can generate a local optimum in terms of traffic movements, but may have a detrimental effect on traffic flows in network as a whole, limiting the effectiveness of other agents. Thus, having multiagent in a greedy or self-interested way is not a proper choice, and some sort of coordination or information sharing mechanism is necessary to implement the system relevant to the real world [120].

1.5 Conclusion and objective of the thesis

As we can see from the review of the existing traffic signal control systems and the related traffic signal optimization research, the approaches in offline system, using historically measured data to determine optimal signal timings, is inferior than those operate online. Because historical data typically does not accurately describe current traffic states, and traffic conditions do not remain static over time. Secondly, the signal control system using centralized control architecture is not good as the one designed in a distributed way. Because many centralized systems may be unable to make real-time signal plan

updates. Moreover, failures of communication in centralized system maybe appear. By comparing, the online distributed control system is superior to the offline or centralized one. As the development of intelligent transportation system, the adaptive traffic signal control system which operates on real-time in the coordinated and distributed network attracts much attention. In addition, research on autonomous traffic control system is emerging recently. This field exceeds the scale of the thesis, but it is worth learning from the related research.

Today, it is required to satisfy the increasing traffic demand, as well as the reductions of traffic delay and traffic congestion. To seek an efficient traffic signal control mechanism is an urgent task. Much research focuses on this field by using artificial intelligent methods, especially the reinforcement learning technique combining with agent theory. Notably, RL is an important branch of AI and has shown promising potential in solving adaptive traffic signal control problems. RL method offers a convenient and efficient way to obtain a near-optimal solution of traffic signal control problem. Meanwhile, the coordinated MARL has some successful applications at network. It usually makes decisions with model-free traffic environment or simplifies traffic conditions in real-word. Of course, there are many interesting topics in this area that deserve further exploration, including how to properly define the reward function, how to identify the best state variables, etc.

In another aspect of the review, RL and ADP offers efficient solution of the traffic signal control problem formulated by MDP. Under the umbrella of MDP, it is beneficial for us to model a problem with discrete states and decisions in discrete-time. Therefore, by using a micro-simulation model, traffic signal control algorithm at network will be studied based on the similar traffic environment in reality. Knowing from the literature, CA-based model can update dynamic system in parallel for all behaviors of individual vehicles. It means all vehicles moving on network in a discrete-time procedure, which is suitable for the case study by using RL or ADP approach.

As for control algorithm, MDP is usually solved by the conventional DP with global optimum solution. However, this may cause “curse of dimensionality” when large state space appears in traffic control problems. Previous work interests us to do some investigation by using DP algorithm and related research algorithm. More importantly, with function approximation, RL and ADP are designed to estimate the value function in DP

algorithm. This can achieve computational efficiency as opposed to traverse all the states in DP.

Research in related works also show that with function approximation, ADP and RL are appropriate to be used in adaptive controllers, especially in high-dimension setting of a multi-intersection network. In particular, ADP has some advantages to make a fine planning based on dynamic state model. Moreover, there are still two purposes driving us to study ADP approach and related schemes. One is that having the function approximation in ADP with efficient learning process needs to be studied further. The other one is that coordination in ADP for traffic network control is rarely studied in the literature. These interest us to seek an ADP approach with a special learning technique for adaptive traffic signal control at coordinated network.

In total, the objective of the thesis is that we attend to make a real-time adaptive traffic signal control in a distributed traffic network system. In order to pursue this goal, in this study, three main works are required as follows.

- Traffic dynamic model in a microscopic way will be formed. It will support for traffic signal control modeling and network loading environment which aims to investigate a proper traffic signal control algorithm.

- Study on exact DP based algorithms for an isolated intersection and develop a new near-optimal learning algorithm to improve decision making efficiency.

- A real-time adaptive traffic signal control algorithm is required to develop for isolated intersection and traffic network, considering the coordination. Experimental validation of the algorithm needs in simulation.

CHAPTER 2

DYNAMIC TRAFFIC SYSTEM MODELING

2.1 Introduction

In general, mathematical model and appropriate optimization technique are two main issues that must be addressed for an optimal control problem. In this chapter, we focus on the first issue about the model study of adaptive traffic signal control system. Three aspects of the system model are briefly introduced as follows.

From the knowledge about characteristics of "adaptive" system, we know that adaptive traffic signal controller works in an intelligent way by using detected traffic information to make real-time decisions. Control performance could be improved by adaptive capability for various traffic conditions. Thus, optimizations of signal plan are very important not only for phase intervals, e.g., when to extend or terminate green split, but also for phase sequence, e.g., variable phase sequence or more adaptive one. Conventional adaptive signal control mechanism is that control action is to either extend or terminate current green phase, and phase sequence is either fixed or variable. In this case, whatever, the flow combination related to phase is always constant. We will discuss the fixed and variable phase sequences, additionally propose a new adaptive phase control mode. This part refers to traffic flow organization pattern, which is detailed in Section 2.2.

As we know that the basic control unit of traffic signal control system is signal controller at intersection. The fields of operation research and artificial intelligence work a lot with discrete states and decisions (or actions). The problems that are modeled with continuous states and decisions (and typically in continuous time) are often addressed under the umbrella of "control theory", whereas the problems modeled in discrete time with discrete states and decisions, are often studied at length under the umbrella of "Markov decision processes". In Section 2.3, we formulate the signal control model at intersection by using the framework of MDP.

As for traffic network modeling, the problem formulation is more complex than MDP based model at isolated intersection. Two subproblems are considered in this part. One is the network loading model. The other one is the coordination mechanism between intersections. By the reviews of related research, microscopic simulation model for traffic network loading is very popular and it is appropriate for the investigation of vehicle behaviors and control performances, such as travel time, stops. Besides network layout representation, more importantly, we propose a new vehicle-following model based on cellular automata theory for network loading. On the other hand, coordinated signal control model at network will use the idea of tunable system state. In this multiagent problem, joint action is generated. All about these will be studied in Section 2.4.

2.2 Traffic flow organization patterns

A typical four-approach intersection is shown in Fig. 2.1. There are eight movements and the numbers of the movements are labeled according to NEMA (National Electrical Manufacturers Association) convention [121]. The right-turn movement is integrated into the straight one sharing same signal. It is known that for safety, flow organization is necessary and it requires traffic movements avoiding to the conflicts between them. Traffic movements can be partitioned into combinations, which are grouped by the non-conflicting flows that will have the right-of-way to occupy the conflict zone.

To authors' knowledge, the works on traffic flow organization at a typical intersection are usually about fixed flow combinations, which consist of the fixed non-conflicting movements in conventional four-phase signal mechanism, and the phase sequence operates in a fixed or variable way, see the survey in [103]. We call these two phase sequence control modes as fixed phase sequence (FPS), variable phase sequence (VPS), respec-

tively. Moreover, we will propose a more adaptive way for signal phase control, named adaptive phase sequence (APS). These three patterns of traffic flow organization will be illustrated in detail, and pedestrian passing is not considered in the thesis.

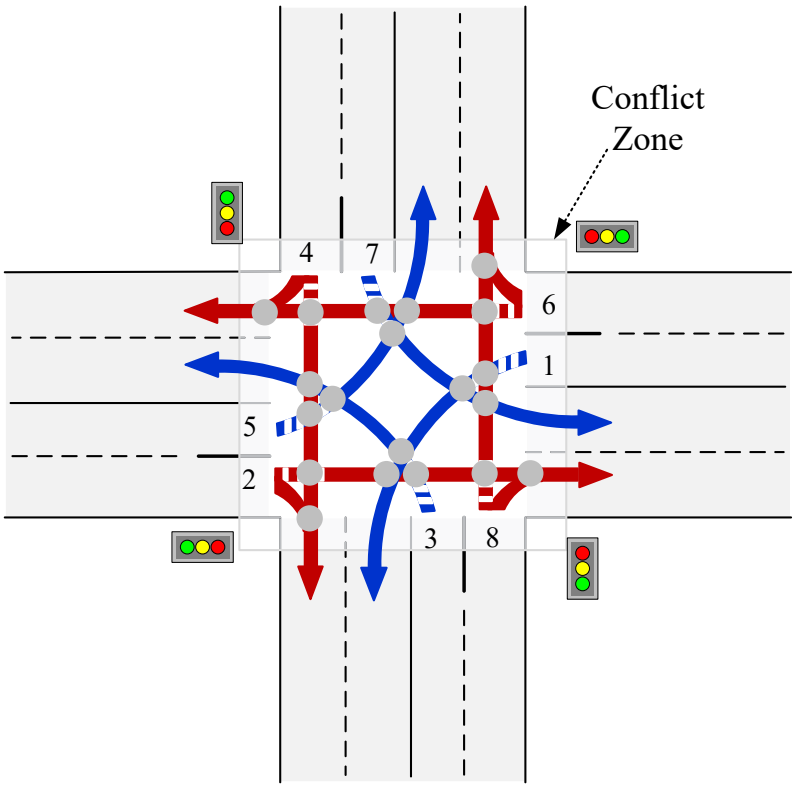


Figure 2.1: Illustration of a typical intersection with conflicts

2.2.1 Fixed phase sequence

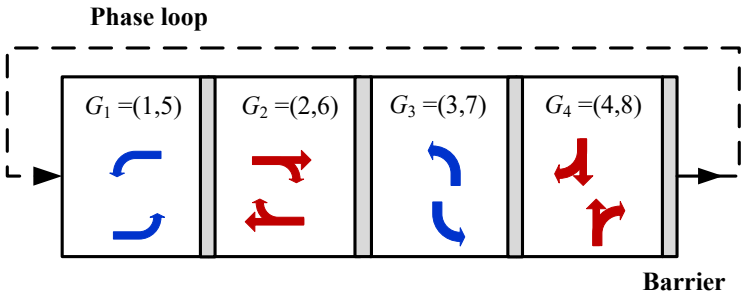


Figure 2.2: Illustration of fixed phase sequence (FPS) mode

Traffic flows or movements can be grouped into flow combinations, which have compatible flows sharing the same signal simultaneously. In FPS mode, as shown in Fig. 2.2, the intersection is controlled typically by four-phase signal, providing green indication to the flow combination of each phase. In this case, the eight movements are divided into four flow combinations, denoted by G_1 , G_2, G_3, and G_4 that

$$G_1 = (1, 5), G_2 = (2, 6), G_3 = (3, 7), G_4 = (4, 8). \tag{2.1}$$

Notice that, the definition in (2.1) is not the only way to group the movements. For example, it can also be defined by

$$G_1 = (1, 6), G_2 = (2, 5), G_3 = (3, 8), G_4 = (4, 7) \tag{2.2}$$

or

$$G_1 = (1, 4), G_2 = (2, 7), G_3 = (3, 6), G_4 = (5, 8). \tag{2.3}$$

Whatever, in conventional signal control system, these flow combinations are predetermined and not changed any more during the operation. In real-world, it is often to see the combinations in (2.1), which will be chosen for our case study in FPS mode.

On the other hand, signal controller organizes these phases by grouping movements in a continuous phase loop. That is to say, the phase may operate one another as follows, and it is a typical pattern to organize the conflicting phases in a particular order, e.g., $G_1 \rightarrow G_2 \rightarrow G_3 \rightarrow G_4 \rightarrow G_1$. Generally, the inter-green interval or red clearance time represented by the barrier is used to separate the phase for different flow directions.

FPS is a common phase control mode and applied widely to current signal control systems. For fixed-time control system, the phase sequence operates in the way of FPS mode, and the phase splits are pre-determined with constant values. For some actuated or adaptive signal control systems, FPS mode is used to organize the movements whereas the phase splits are varied in real-time with different values, according to detected traffic information. Thus, adaptive signal control adopts FPS mode as a common way and its control decision is to either extend or terminate current green phase.

2.2.2 Variable phase sequence

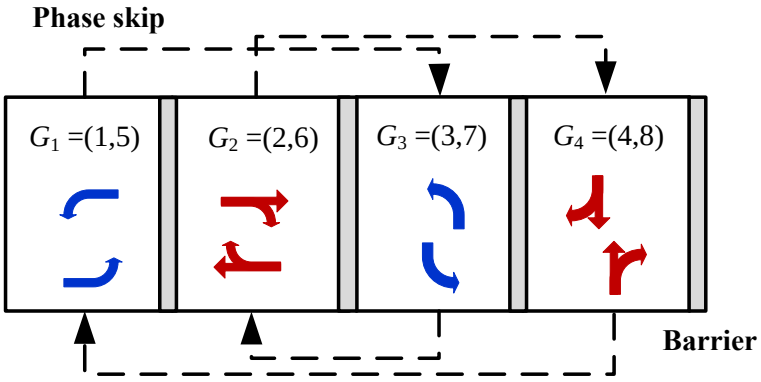


Figure 2.3: Illustration of variable phase sequence (VPS) mode

In VPS mode, the combination of signal phase chosen is the same with FPS in (2.1). Diversely, the phases in VPS may operate one after another in uncertain and unordered way. For example, in Fig. 2.3, the phase sequence operates as $G_1 \rightarrow G_3 \rightarrow G_2 \rightarrow G_4 \rightarrow G_1$. The barriers of red clearance time are used to separate each phase in time.

With VPS mode, actually, signal controller makes phase skip to another one considering the performances, such as queue length, vehicle waiting time, from other phases. We can see that both FPS and VPS can be applied to adaptive signal control system for changing traffic conditions. With FPS mode, controller focuses on the phase split itself being either extend or terminated. However, with VPS mode, controller not only considers the duration of phase split, but also takes the possibility of other phases into account, namely the phase sequence optimization.

As VPS mode supports multiple decisions for signal phases control, the action space of VPS is larger than FPS. That is why previous research in literature focuses less on the control methods with VPS. Recently, due to artificial intelligence techniques with advantages of intelligent computation, the state-action space increased by VPS is not hard to overcome. Importantly, from the reviews of literature, some research especially about RL learning uses VPS mode [47, 75, 93, 105, 111]. VPS mode may promote the performance of adaptive traffic signal control.

2.2.3 Adaptive phase sequence

Lane	1	2	3	4	5	6	7	8
1		0	0	1	1	1	0	0
2	0		0	0	1	1	1	0
3	0	0		0	0	1	1	1
4	1	0	0		0	0	1	1
5	1	1	0	0		0	0	1
6	1	1	1	0	0		0	0
7	0	1	1	1	0	0		0
8	0	0	1	1	1	0	0	

Figure 2.4: Matrix of adaptive phase sequence (APS) mode

In APS mode, both of the phase sequence and the components (lanes) of flow combination are varied. In other words, the possibilities of phase choice are more than four-phase mechanism. In detail, there are many but limited additional traffic flow combinations which can also avoid flow conflicts, e.g., lane 1 can be combined with one of lanes 4, 5, and 6. The possible combinations are selected, the performances may be different. We can list all the combination possibilities (in total 12 only considering the upper triangular matrix as symmetry), as shown in Fig. 2.4. Note that for the relationship (combination coefficient) between two lanes, assign 1 if they are non-conflicting, or assign 0 otherwise. Therefore, one possible phase sequence may operate like (1, 5) $\rightarrow$ (1, 6) $\rightarrow$ (2, 6) $\rightarrow$ (3, 8) $\rightarrow$ (4, 7) $\rightarrow$ . . . . It shows clearly in Fig. 2.5.

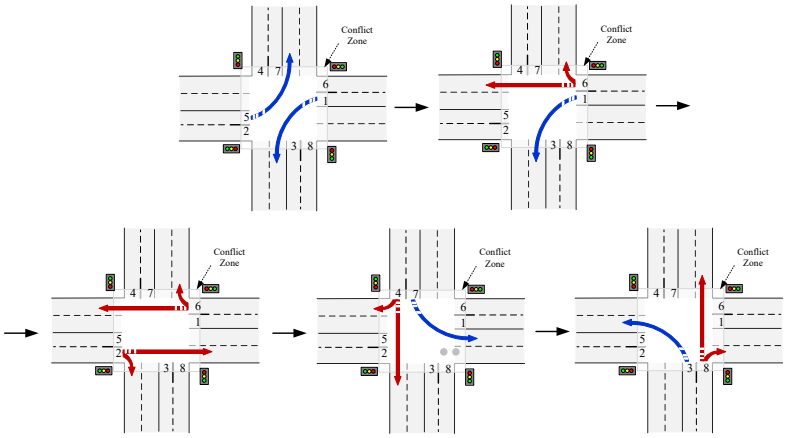


Figure 2.5: Example of APS mode

APS contains all the possible flow combinations without any sequence required. It is firstly investigated in our algorithm, and will be implemented in the later case study.

From the discussion above, we can find that the adaptive capability of FPS, VPS, or APS is strengthened one by one. For adaptive traffic signal control, normally, signal phase duration is not fixed, but phase sequence is still cyclic, it is the way of FPS mode. While, the next alternative phase may not work in a particular phase order. According to different traffic conditions, phase sequence can work in an acyclic way, like the VPS mode. Moreover, in the case of APS, the combination grouped by non-conflicting flows can also be selected properly as long as other non-conflicting flow combinations exist, and phases have no sequence required as well.

It is well-known that intelligent vehicle technology is progressing very rapidly. Recently, autonomous traffic management at un-signalized intersections interests many researchers. Vehicles can be guided using wireless communication to pass through the intersection automatically. Although APS mode is based on the concept of signal phase organization, it offers a highly adaptive way to schedule vehicles passing (or turning) through the conflict zone. Even without regard to signalized infrastructures, it can extend to some situations of Vehicle-to-Vehicle and Vehicle-to-Infrastructure communication. In some degree, it almost approaches the autonomous traffic management system, with both considerations of the safety for flows passing and the efficiency of adaptive control.

2.3 System modeling at isolated intersection

Normally, traffic signal control system at intersection can be seen as a stochastic discrete event system. The discrete intervals can be represented by the stages decomposed in the optimization dynamic process. Traffic signal control problem can be processed by a multi-stage decision making procedure in discrete-time. It is very useful and efficient to solve a complex problem, especially for adaptive traffic signal control problem. For example, the widespread adaptive signal systems, such as DYPIC [15], PRODYN [16], and OPAC [13] are all based on the multi-stage decision making. Recently, traffic signal control model based on the framework of discrete-time Markov Decision Process (MDP) attracts much attention. Because MDP can describe stochastic traffic environment and the based model is usually solved by DP and related schemes, especially RL and ADP techniques. We will introduce related knowledge of MDP and use MDP for signal control system modeling at isolated intersection.

2.3.1 Markov Decision Process

As for an MDP (only discrete-time MDP in the thesis), we mean a stochastic process $\{Y_t\}$ that takes values in a state set which is governed by a control sequence $\{Z_t\}$, and satisfies the following controlled Markov property:

$$\begin{aligned} &P(Y_{t+1} = s_{t+1} | Y_t = s_t, Z_t = a_t, Y_{t-1} = s_{t-1}, Z_{t-1} = a_{t-1}, \ldots, Y_0 = s_0, Z_0 = a_0) \\ &= P(Y_{t+1} = s_{t+1} | Y_t = s_t, Z_t = a_t) = p(s_t, a_t, s_{t+1}). \end{aligned} \tag{2.4}$$

The environment of the decision problem we discuss is described by a finite MDP [88].

Definition 1 *A finite MDP is a tuple* $< S, A, p, R >$ *where* S *is the finite discrete set of environment states,* A *is the finite set of actions,* $p : S \times A \times S \to [0, 1]$ *is the state transition probability function, and* $R : S \times A \times S \to \mathbb{R}$ *is the reward function.*

The state $s_t \in S$ describes the environment at each time t. The controller can choose the state at each time by taking actions $a_t \in A$. As a result of the action a_t, the environment changes its state from s_t to some $s_{t+1} \in S$, according to the state transition probabilities given by p which is represented as $p(s_t, a_t, s_{t+1})$. The controller receives immediate scalar reward $r_t \in \mathbb{R}$ according to the reward function $R : r_t = R(s_t, a_t, s_{t+1})$.

For deterministic models, the transition probability function is replaced by a transfer function $\sigma(s_t, a_t)$ simply expressed as $\sigma : S \times A \rightarrow S$. It follows that the reward is completely determined by the current state and action: $r_t = R(s_t, a_t)$, $R : S \times A \rightarrow \mathbb{R}$.

Given an initial state s_0 and a sequence of decisions a_t, the optimization problem is to minimize (or maximize depending on the problem) the expected total reward, which is expressed as

$$\min_{a_t \in A} E\left\{\sum_{t=0}^{T-1} \gamma^t r_t | s_0 = s\right\}, \tag{2.5}$$

where T is the horizon, γ $(0 < \gamma < 1)$ is the discount factor.

2.3.2 Model definitions of characteristic in MDP

2.3.2.1 Model assumption

Traffic signal control problem can be formulated by using the framework of MDP. Assume that one interval is from t to $t + 1$. At first, the following principal assumptions are given.

(1) The indications of traffic signals are formulated in discrete-time and divided into unit intervals. The size of one interval is 2 seconds.

(2) Queue lengths are calculated at the end of each temporal interval, and signals may only be changed at the boundary between intervals.

(3) Signal phases are composed of effective greens and reds only, thus excluding amber intervals.

(4) Each phase contains at least mandatory intervals including inter-green interval and minimum green interval, during which no signal switching is admissible. The extension of green signal is one interval per step.

(5) There is no lost time for vehicle receiving green signal.

(6) The discharge rate (saturation flow) on each lane is one vehicle per interval. This rate is equivalent to 1800 vehicles per hour.

2.3.2.2 Model framework

From the definition of MDP in 2.3.1, the problem formulation requires the characterization of state, action, transition probability, and objective criterion of reward (cost) function. Traffic signal control problem can be well described as follows.

Traffic state

At time t, for an isolated signalized intersection having total N lanes, traffic state can be expressed by $s_t = (k_t, x_t)$, $s_t \in S$, where k_t is the vehicle state vector and x_t is the signal state vector. For each lane n ($n = 1, \ldots, N$), traffic state is expressed as $s_t(n) = (k_t(n), x_t(n))$ and we have

$$\begin{aligned} s_t &= (s_t(1), s_t(2), \ldots, s_t(N))^{\mathrm{T}}, \\ k_t &= (k_t(1), k_t(2), \ldots, k_t(N))^{\mathrm{T}}, \\ x_t &= (x_t(1), x_t(2), \ldots, x_t(N))^{\mathrm{T}}. \end{aligned} \tag{2.6}$$

We define the vehicle state $k_t(n)$ on each lane n by the actual number of queuing vehicles. Assume that the maximum capacity of lane n is L_n, thus

$$0 \leq k_t(n) \leq L_n. \tag{2.7}$$

The signal state $x_t(n)$ on each lane n is either green or red indication. We define $x_t(n)$ to be a binary variable satisfying

$$x_t(n) = \begin{cases} 1, & \text{if signal is green on lane } n \\ 0, & \text{if signal is red on lane } n. \end{cases} \tag{2.8}$$

Traffic action

The decision or action of the signal controller is $a_t = (a_t(1), a_t(2), \ldots, a_t(N))^{\mathrm{T}}$, $a_t \in A$. In adaptive traffic signal control system, the definition of action at time t on lane n is to switch the current green phase to next one or unchanged, namely extending the current green phase. We define $a_t(n)$ to be binary variable, which is expressed as

$$a_t(n) = \begin{cases} 1, & \text{for signal switch on lane } n \\ 0, & \text{unchanged on lane } n. \end{cases} \tag{2.9}$$

In addition, the mandatory minimum green interval $g_{\min}$ and inter-green interval g_{int} are executed for safety. During the mandatory intervals, we have $a_t(n) = 0$.

State transition

Traffic signal control problem is a stochastic discrete event for decision making. Traffic state transition probability $p(s_t, a_t, s_{t+1})$ is generated when state s_t transforms to s_{t+1} by taking action a_t. Since traffic movements are independent, the transition probability of the whole intersection is given by

$$p(s_t, a_t, s_{t+1}) = \prod_{n=1}^{N} p_n(s_t(n), a_t(n), s_{t+1}(n)). \tag{2.10}$$

Since $s_t(n) = (k_t(n), x_t(n))$, for simplicity, we use $p_n(k_t(n), a_t(n), k_{t+1}(n))$ to express the transition probability of vehicle state changing from $k_t(n)$ to $k_{t+1}(n)$ in the condition of the signal state $x_t(n)$, and it depends on random arrivals and the chosen action $a_t(n)$ for departures. Let q_n be the constant probability of vehicle arriving during one interval on lane n. Thus, $1 - q_n$ is the probability of no arrival. According to the stochastic control problem with a finite state space, the transition probabilities are defined as follows. If taking action $a_t(n)$, traffic flow on lane n receives green signal during the coming interval, the transition probabilities on lane n are given by

$$\begin{cases} p_n(k_t(n), a_t(n), k_{t+1}(n) - 1) = 1 - q_n; & (0 < k_t(n) \leq L_n, k_t(n) \in \mathbb{N}), \\ p_n(k_t(n), a_t(n), k_{t+1}(n)) = q_n; & (0 < k_t(n) \leq L_n, k_t(n) \in \mathbb{N}), \\ p_n(k_t(n), a_t(n), k_{t+1}(n)) = 1; & (k_t(n) = 0). \end{cases} \tag{2.11}$$

Otherwise, the action $a_t(n)$ implies red signal switch. Then,

$$\begin{cases} p_n(k_t(n), a_t(n), k_{t+1}(n)) = 1 - q_n; & (0 \leq k_t(n) < L_n, k_t(n) \in \mathbb{N}), \\ p_n(k_t(n), a_t(n), k_{t+1}(n) + 1) = q_n; & (0 \leq k_t(n) < L_n, k_t(n) \in \mathbb{N}), \\ p_n(k_t(n), a_t(n), k_{t+1}(n)) = 1; & (k_t(n) = L_n). \end{cases} \tag{2.12}$$

Normally, the probability model describing the information process is hard to be obtained. Thus, we can use the power of computer to generate random observations, which satisfy a specific distribution. The process is generally referred as Monte Carlo sampling. Deterministic state transitions with the stochastic arrival information $w_t = (w_t(1), w_t(2), \ldots, w_t(N))^{\mathrm{T}}$ using Monte Carlo method can factually describe traffic environment in a simulation way. Actually, there are differences between a distribution model and a sample model. Given a starting state and action, a distribution model generates all possible transition weighted by their probabilities of occurring, and a sample

model produces a possible model. In many applications it is much easier to obtain sample models than distribution models [58].

We use transfer functions instead of transition probabilities to describe the deterministic state transitions in system. Once the system has made a decision on signal status at time t, the state of intersection will be changed. The transition of signal state is described as

$$x_{t+1}(n) = (x_t(n) + a_t(n)) \mathrm{mod}_2 \tag{2.13}$$

and the vehicle state represented by queue length $k_t(n)$ is transferred as

$$k_{t+1}(n) = k_t(n) + w_t(n) - y_t(n) \tag{2.14}$$

where $w_t(n)$ denotes the traffic arrivals satisfying the distribution according to traffic arrival rates. It adopts the value of either 0 or 1 vehicle/interval (veh/int). The traffic departure rate $y_t(n)$ is also a binary variable constricted by

$$y_t(n) = \begin{cases} 1, & \text{if } x_t(n) = 1 \text{ and } k_t(n) + w_t(n) \geq 1 \\ 0, & \text{otherwise.} \end{cases} \tag{2.15}$$

As we can see that the vehicle state at the next time step is determined by the system state s_t, information of future vehicle arrivals w_t, and policy decision a_t at the current step t. The state transition at each step is deterministic. However, traffic arrivals satisfy a stochastic process. Since state transitions are influenced by random arriving traffic, the process of vehicle state can be seen as a stochastic process with Markov property.

Reward function

The objective of traffic signal control is to minimize the overall average waiting time per vehicle. In the end of each interval, the total number of vehicles can be calculated easily. In a rational way, the one-step transition reward r is measured by the sum of queue lengths at the next time $t + 1$ (or in a simple way, the sum of maximums of two queue lengths measured in the combination), which is defined as

$$r_t = \sum_{n=1}^{N} k_{t+1}(n). \tag{2.16}$$

Thus the objective of optimization for traffic signal control at intersection can be determined by expected total reward function in (2.5).

2.4 System modeling at traffic network

For traffic network control in a simulation way, normally, a simulation model is required for network time-varying loading. A microscopic way for network loading model is considered here. Vehicle moves on the network with properties of varying speed, position, and direction. In particular, a new vehicle-following model is emphasized. On the other hand, the traffic signal control mechanism is based on the isolated intersection control model discussed in Section 2.3. Moreover, the coordination between intersections is taken into account by using the idea of tunable state control.

2.4.1 Network loading model in micro-simulation system

2.4.1.1 Network representations

A classical type of 5-intersection network is studied, as shown in Fig. 2.6. In this typical network system, essential elements are the intersections, links, lanes, individual vehicles, and signal controllers. System elements have some properties that are essential to construct dynamic traffic models. The network contains several intersections connected with links. The link consists of two lanes where vehicles are involved. The signal controllers at intersections send the right-of-way for vehicles passing through (or turning) the conflict zones.

Intersection

For each intersection, eight movements are depicted (assume that the right-turn movement is integrated into the straight one sharing the same signal). In the small boxes represented by $S1, S2,$ etc., traffic demand for system simulation is generated for each lane of the link. The input random traffic data satisfies Bernoulli 0-1 distribution in a discrete-time procedure. The traffic signal controller located at each intersection coordinates with neighbors and controls the different directions of traffic flows.

Links

The link has two lanes referring to the left-turn lane and the lane of straight forward combined with the right-turn. There are three kinds of links described in the network. These are called entrance link, inside link, and exit link. The entrance link does not have

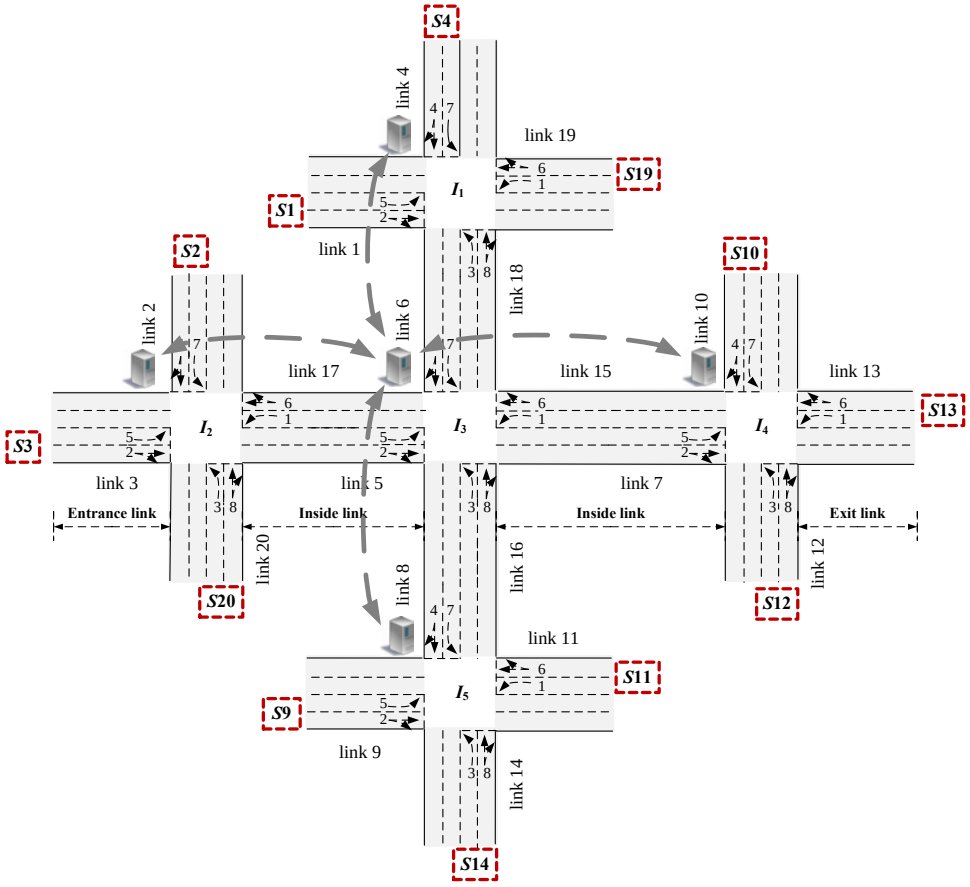


Figure 2.6: 5-intersection traffic network

any upstream link belonging to the network. Traffic data sources are all positioned there. The inside link is between two intersections. Individual vehicles move on the inside links after evacuating the intersection. The exit link is the one that vehicles output from the network.

Lane choice

The lanes are discretized into unit places with equal length, as shown in Fig. 2.7. Note that the length of each unit place equals to the minimum head-head distance in queue. Thus, the place is occupied only by one vehicle or empty. From the beginning of the link to the end of the link, places are numbered from 1 to maximum length. The intersection zone (node) is the buffer place set by $L + 1$.

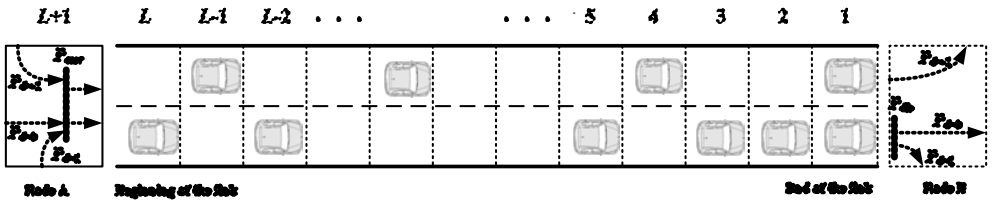


Figure 2.7: Inside link

The lane choice in our model is carried out by proportions at nodes of the network. Three merging and diverging movements are indicated in node A and B, respectively. Let $P_{d=-1}, P_{d=0}$, and $P_{d=1}$ be the proportions of left-turn, straight forward, and right-turn, respectively. In node B, we define

$$\begin{aligned} P_{d=-1} &= P^1_{mer}, \\ P_{d=0} &= P^2_{mer} P^1_{div}, \\ P_{d=1} &= P^2_{mer} P^2_{div}, \end{aligned} \tag{2.17}$$

where the merging proportions $P_{mer} = [P^1_{mer}, P^2_{mer}]$ are two random distributions to the left-turn and straight (right-turn) lanes after vehicles leaving A, and the diverging proportions $P_{div} = [P^1_{div}, P^2_{div}]$ are two random distributions assigned to the directions of straight forward and right-turn before vehicles entering B.

2.4.1.2 Vehicle-following model

In a microscopic way, the very popular network loading model about vehicle-following mechanism is based on cellular automata (CA) theory. From a theoretical point of view, four main ingredients play an important role in cellular automata models [45]. (1) *The physical environment* is the underlying structure consisting of a discrete lattice of cells. (2) Each cell can be in a certain *state*, where typically an integer represents the number of distinct states. (3) For each cell, define a *neighborhood* that locally determines the evolution of the cell. (4) *A local transition rule* acts upon a cell and its direct neighborhood, such that the cell's state changes from one discrete time step to another (i.e., the system's iterations).

In a traffic problem, CA-based model can update the dynamic system in parallel for all the behaviors of individual vehicles. It presents all vehicles moving on the network in a discrete-time procedure. Research has shown that CA-based model can yield realistic

behavior. One of the most popular CA-based model for vehicle-following is the Nagel-Schreckenberg (NaSch) model [42].

NaSch model

The NaSch model was originally defined on a single-lane road. The road is subdivided into cells, which can be either empty or occupied by one vehicle. Every vehicle has a non-negative integer velocity. For the update of the road, the following four steps are performed simultaneously for all vehicles:

- **Acceleration**: if the velocity v of a vehicle is lower than $v_{\max}$, and if the distance to the next vehicle ahead is larger than $v + 1$, the velocity is advanced by one.
- **Slowing down**: if a vehicle at place a looks ahead the next vehicle at place $a + b$ with $b \leq v$, it reduces the velocity to $b - 1$.
- **Randomization**: with probability ρ, the velocity of each vehicle (if $v > 0$), is decreased by one.
- **Vehicle motion**: each vehicle is advanced v places.

The randomization takes into account that individual driving behaviors for different vehicles result in non-deterministic dynamics of vehicle motions in reality.

A new vehicle-following model

Before studying the vehicle-following model, general notations and definitions are given, and all variables assume integral values on lane n. Let:

- L_l be the length of link l, defined by the total number of unite places;
- $p_{i,t}$ be the position of vehicle i in the unit place at time t, $p_{i,t} \in [0, L_l+1]$, especially $p_{i,t}$ equals 0 or $L_l + 1$ when vehicle is in the conflict zone after leaving the stop line or before entering the lanes of downstream link, respectively;
- $v_{i,t}$ be the velocity (place/int) of vehicle i, $v_{i,t} \in [0, v_{\max}]$, especially $v_{i,t} = 1$ when vehicle is in the conflict zone;
- $\Delta p_{i,j,t}$ be the total empty places between the adjacent vehicles i and j;

- $\Delta v_{i,j,t}$ be the difference of velocities between the adjacent vehicles i and j;
- k_t be the queue length (veh).

In NaSch freeway model, traffic lane is divided into cells of equal size and each vehicle can move with an integer velocity. Vehicle velocity has properties of acceleration, slowing down, and randomization. The underlying traffic model moves the individual vehicles on the discrete sites of lane. The new position as well as the velocity of each vehicle is updated during the time interval. That is to say, in a discrete-time procedure, all vehicles move in parallel according to their current positions and velocities. In the study of the urban traffic network, the link distance between two adjacent intersections is not so long as a freeway. Thus, we just consider the acceleration and deceleration of vehicle velocity. The randomization, which presents the probability of velocity depending on human behaviors or external varying conditions, is not taken into account in our case. For simplicity, vehicles between lanes in the link are independent and the first-in-first-out (FIFO) rule is accepted in the model. In the case of the traffic network system, a new vehicle-following model will be discussed.

Note that places on lane are indexed from L_l to 1 for the entering place to the approaching place at intersection. Vehicle moving on lane is related to the position and velocity, and those of the vehicle ahead. The relations about the position and velocity between post-vehicle i and pre-vehicle j can be expressed as

$$\begin{aligned} &\Delta p_{i,j,t} = p_{i,t} - p_{j,t} - 1, \quad 0 \leq \Delta p_{i,j,t} \leq L_l; \\ &\Delta v_{i,j,t} = v_{i,t} - v_{j,t}, \qquad\quad 0 \leq \Delta v_{i,j,t} \leq v_{\max}. \end{aligned} \tag{2.18}$$

We know that the post-vehicle can accelerate, decelerate, and move with constant velocity, according to the distance and velocity of the pre-vehicle. In the vehicle-following model, two basic procedures are considered in order. Firstly, the post-vehicle i accelerates or decelerates the same velocity with the pre-vehicle j simultaneously. After that, the additional $\Delta v_{i,j,t}$ is considered into the post-vehicle i. Note that post-vehicle i always moves in the maximum relative velocity $\Delta v_{i,j,t}$ under the security distance. Meanwhile, it satisfies $v_{i,t} \geq v_{j,t}$. For safety, $\Delta v_{i,j,t}$ is the required value that can be uniformly reduced from maximum to zero depending on the distance $\Delta p_{i,j,t}$. In order to obtain $\Delta v_{i,j,t}$ and update the states of post-vehicle i, there are total four cases discussed as follows in the conditions of queue length k_t and the position of pre-vehicle j.

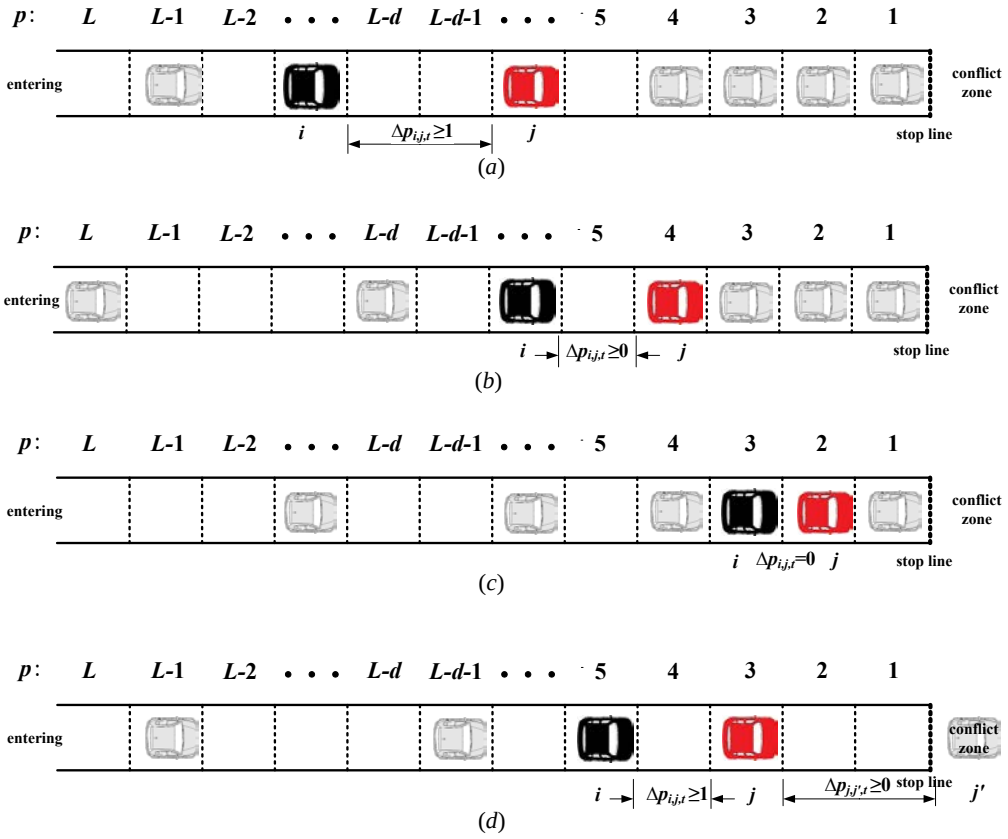


Figure 2.8: Cases in vehicle-following model: (a) Case 1, (b) Case 2, (c) Case 3, and (d) Case 4

Case 1: $k_t \neq 0$ and $p_{j,t} > k_t$. It indicates that there exists a queue length and the pre-vehicle j is not in the queue. Thus, the post-vehicle i is also not in the queue. Assuming the distance $\Delta p_{i,j,t}$ ensures that the $\Delta v_{i,j,t}$ can be reduced from maximum to zero uniformly, as shown in Fig. 2.8(a). Therefore, the minimum $\Delta p_{i,j,t}$ can be determined during $\Delta v_{i,j,t} + 1$ steps. Moreover, in order to move vehicles under the security distance (set to be one place in this paper) when $\Delta v_{i,j,t} = 0$, the distance $\Delta p_{i,j,t}$ should be increased by one. Thus, the restriction can be expressed as

$$\Delta p_{i,j,t} \geq \frac{(1 + \Delta v_{i,j,t})\Delta v_{i,j,t}}{2} + 1. \tag{2.19}$$

Case 2: $k_t \neq 0$ and $p_{j,t} = k_t$. In this case, the pre-vehicle j is the last vehicle in the queue and post-vehicle i is not yet added, as shown in Fig. 2.8(b). As the same with Case 1, the maximum relative velocity $\Delta v_{i,j,t}$ can be determined by $\Delta p_{i,j,t}$. The only difference is that the security distance is zero. It means when $\Delta p_{i,j,t} = 0$, post-vehicle i just arrivals

at the end of queue length. Thus, we have,

$$\Delta p_{i,j,t} \geq \frac{(1+\Delta v_{i,j,t})\Delta v_{i,j,t}}{2}. \tag{2.20}$$

Case 3: $k_t \neq 0$ and $p_{j,t} < k_t$. In this case, the pre-vehicle j as well as the post-vehicle i is in the queue, as shown in Fig. 2.8(c). We assume that, if green signal is accepted for lane n, all the vehicles in this queue have velocity one; otherwise zero. Thus, we can conclude that if $\Delta v_{i,j,t} = 0$ and $\Delta p_{i,j,t} = 0$, vehicles i and j in the queue indicate the same velocities being of either 1 or 0. In this case,

$$\Delta p_{i,j,t} = \Delta v_{i,j,t} = 0. \tag{2.21}$$

Case 4: $k_t = 0$. In this case, the relation of $\Delta p_{i,j,t}$ and $\Delta v_{i,j,t}$ between the pre-vehicle j and post-vehicle i is the same with Case 1. However, in Case 4, there is no vehicle in the queue. Therefore, we should define the limited velocity of the first pre-vehicle j. It should be guaranteed that the first pre-vehicle j at least can uniformly decelerate the velocity to zero before the stop-line when receiving red signal. Imagine that a virtual vehicle j' locates in front of the place '1' with velocity 0 in red signal or 1 otherwise, as shown in Fig. 2.8(d). According to Case 2, the relative velocity $\Delta v_{j,j',t}$ can be determined as well. It satisfies

$$\Delta p_{j,j',t} \geq \frac{(1+\Delta v_{j,j',t})\Delta v_{j,j',t}}{2}. \tag{2.22}$$

Solving (2.22), the maximum $\Delta v_{j,j',t}$ can be obtained by

$$\Delta v_{j,j',t} = \lfloor \frac{\sqrt{8\Delta p_{j,j',t}+1}-1}{2} \rfloor, \tag{2.23}$$

where the function $y = \lfloor x \rfloor$ is defined as that y is the maximum integer not larger than x. Thus, for next time step, the velocity and position of the first pre-vehicle j can be respectively written as

$$\begin{aligned} v_{j,t+1} &= \min(v_{j',t} + \Delta v_{j,j',t}, v_{\max}), \\ p_{j,t+1} &= \max(p_{j,t} - v_{j,t+1}, 0) \end{aligned} \tag{2.24}$$

where $v_{j',t} = 0$ if receiving red signal and $v_{j',t} = 1$ otherwise.

In conclusion, the relative velocity $\Delta v_{i,j,t}$ can be obtained by solving (2.19), (2.20), and (2.21). That is,

$$\Delta v_{i,j,t} = \begin{cases} \lfloor \frac{\sqrt{8(\Delta p_{i,j,t}-1)+1}-1}{2} \rfloor, & \text{if } p_{j,t} > k_t \geq 0 \text{ and } \Delta p_{i,j,t} \geq 1 \\ \lfloor \frac{\sqrt{8\Delta p_{i,j,t}+1}-1}{2} \rfloor, & \text{if } k_t \neq 0, p_{j,t} \leq k_t \text{ and } \Delta p_{i,j,t} \geq 0. \end{cases} \tag{2.25}$$

According to the Case 4 and $\Delta v_{i,j,t}$ in (2.25), the $v_{i,t}$ and $p_{i,t}$ of post-vehicle i can be updated eventually by

$$\begin{aligned} v_{i,t+1} &= \min(v_{j,t} + \Delta v_{i,j,t}, v_{\max}), \\ p_{i,t+1} &= \max(p_{i,t} - v_{i,t+1}, 0). \end{aligned} \tag{2.26}$$

In total, the vehicle-following model is based on (2.24) and (2.26). On each lane, all the vehicles from the head one to the last have the ergodic process to update the velocity and position in each time step. The process is extended to the network. We use $p_{i,t+1}$ to judge that either vehicle i at next time $t+1$ will leave the stop line or join the queue length. Therefore, the traffic departure and arrival can be determined in the model.

2.4.2 Coordinated signal control model at network

2.4.2.1 Multiagent MDP

If the isolated intersection can be formulated by a single agent MDP framework, the network case can be modeled in its extension called multiagent MDP. The generalization of multiagent MDP is defined as follows.

Definition 2 *A finite multiagent MDP is a tuple* $< S, A, p, R >$ *where* S *is the finite discrete set of environment states and* $S = S_1 \times \cdots \times S_M$ *with* M *agents,* A *is the finite set of joint actions and* $A = A_1 \times \cdots \times A_M$, $p : S \times A \times S \to [0, 1]$ *is the state transition probability function, and* $R : S \times A \times S \to \mathbb{R}$ *is the reward function.*

Being similar definition to a single agent MDP, in multiagent case, the state $\boldsymbol{s}_t = (s_t^1, s_t^2, \ldots, s_t^M)$, $\boldsymbol{s}_t \in S$, describes the environment at each time t. The controller can choose the state at each time by taking joint actions $\boldsymbol{a}_t = (a_t^1, a_t^2, \ldots, a_t^M)$, $\boldsymbol{a}_t \in A$. As a result of the action $\boldsymbol{a}_t$, the environment changes its state from $\boldsymbol{s}_t$ to some $\boldsymbol{s}_{t+1} \in S$, according to the state transition probabilities given by p which is represented as $p(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1})$. The controller receives immediate a scalar reward $r_t \in \mathbb{R}$ according to the reward function $R : r_t = R(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1})$. For deterministic models, the transition probability function p is simply expressed as $p : S \times A \to S$. It follows that the reward is completely determined by the current state and action: $r_t = R(\boldsymbol{s}_t, \boldsymbol{a}_t)$, $R : S \times A \to \mathbb{R}$.

Given the initial state $\boldsymbol{s}_0$ and a sequence of decision $\boldsymbol{a}_t \in A$, $(t = 0, 1, 2, \ldots, T-1)$, the objective is to minimize the discount expected total reward

$$\min_{\boldsymbol{a}_t \in A} E\left\{\sum_{t=0}^{T-1} \gamma^t r_t | \boldsymbol{s}_0 = \boldsymbol{s}\right\} \tag{2.27}$$

where γ $(0 < \gamma < 1)$ is the discount factor.

It is known that transition probability function is hard to receive in complex problem. Usually, Monte Carlo sampling is applied to multiagent MDP. So that the stochastic problem is formulated by deterministic models using observed stochastic distribution data referring to traffic arrivals.

2.4.2.2 Tunable state control for coordination

For traffic network control, an individual intersection can self-organize by using local information or coordinate its action by using the information from neighbors, just like the communication between agents. Generally, Markov decision process (MDP) is regarded as the mathematic foundation for RL and ADP. Here, the tunable state and joint action between two adjacent intersections are discussed when the coordination at network is considered.

By using the basic assumptions and notations at isolated intersection control, some definitions of variables for network control are given as follows. Let:

- $\boldsymbol{k}_t$ be the vehicle queue length matrix with dimension $N \times M$ that $\boldsymbol{k}_t = (k_t^m, m = 1, \ldots, M)$ and $k_t^m = (k_t^m(n), n = 1, \ldots, N)^T$, where M is the total number of intersections at network and N is the total number of lanes at intersection;
- $\tilde{\boldsymbol{k}}_t$ be the total vehicles occupied on lane with dimension $N \times M$;
- $\boldsymbol{x}_t$ be the system signal state with dimension $N \times M$, and assign the element $x_t^m(n)$=1 to signal green and assign $x_t^m(n)$=0 to red on lane n at intersection m;
- $\boldsymbol{a}_t$ be the system action with dimension $N \times M$ and assign the element $a_t^m(n)$=1 to switch signal and assign $a_t^m(n)$=0 to signal unchanged on lane n at intersection m;
- $\boldsymbol{w}_t$ be the traffic arrival information for queue length with dimension $N \times M$;
- $\tilde{\boldsymbol{w}}_t$ be the traffic arrival information for the lane with dimension $N \times M$;

- y be the traffic departure rate (veh/int) on lanes, assuming all lanes have the same value 1 veh/int, which is equivalent to 1800 veh/h.

It is found that, in the case of short and fine planning stages, the influence from the intersection to the other one cannot appear immediately. In other words, the vehicle platoon generated from upstream to arrive at downstream makes a delayed influence on the decision making of the local intersection. In the view of independent agent system, only the local information approaching intersection is normally used. We use vehicle queue length to act as a local state. In a global view, the adjacent intersection affects the local one implicitly in some future time. However, in adaptive traffic signal control, it is hard to determine and synchronize this effect. The influence only happens on the lane of inside link where the vehicle states, including the queuing one and moving one, are changed by the joint action of two adjacent intersections. On the other hand, vehicles in the link are normally viewed as the stability performance of the network. Thus, we use the total number of vehicles on lane to act as a coordinated state. Rather than processing the local state and the coordinated state separately, such as an agent hierarchical structure referring to a local agent and a supervisor agent, our system state is the integration of these two states with tunable weights. In this way, it can not only reduce the computation complexity caused by too many agents, but also overcome the negative effects from the single aspect of independent agent control or coordinated agent control at network. In addition, the joint action is only considered in the inside link. Especially for the entrance link, state transitions depend on the data generator and the actions of local intersection. Here, we discuss the definition of tunable system state in the view of intersection and analysis the dynamic state transition in the view of traffic lane.

At local intersection u, the local state is expressed as $\hat{s}_t^u = (k_t^u, x_t^u)$, which will be transfered to $\hat{s}_{t+1}^u$ with action a_t^u. As for the coordinated state, which refers to local intersection u and adjacent intersection v, it is expressed as $\hat{s}_t^{u,v} = (\tilde{k}_t^u, x_t^u, x_t^v)$ and will be transfered to $\hat{s}_{t+1}^{u,v}$ with action a_t^u and a_t^v. According to $\hat{s}_t^u$ and $\hat{s}_t^{u,v}$, the tunable state at intersection u can be written as

$$s_t^u = (ek_t^u + (1-e)\tilde{k}_t^u, x_t^u, x_t^v), v \in \Gamma(u) \tag{2.28}$$

where e $(0 \leq e \leq 1)$ is the tunable parameter, and $\Gamma(u)$ is the neighbor set of u. Then, the integration system state at network is $\boldsymbol{s}_t = (s_t^1, s_t^2, ..., s_t^M)$, where M is the total number of intersections.

It is easy to calculate the dynamic transition of binary signal state $\boldsymbol{x}_t$ according to the binary action $\boldsymbol{a}_t$. For the dynamic state transitions of $\boldsymbol{k}_t$ and $\tilde{\boldsymbol{k}}_t$, more details are given. Considering the lane n in inside link between local intersection u and adjacent intersection v, the transitions of queuing vehicles and all vehicles on lane n are respectively expressed as follows

$$\begin{aligned} k_{t+1}^u(n) &= k_t^u(n) + w_t^u(n) - g_t^u(n), \\ \tilde{k}_{t+1}^u(n) &= \tilde{k}_t^u(n) + \tilde{w}_t^u(n) - \tilde{g}_t^u(n), \end{aligned} \tag{2.29}$$

where for vehicles in queuing, the arrival $w_t^u(n)$ and departure $g_t^u(n)$ are given by

$$w_t^u(n) = \begin{cases} 1, & \text{if } p_{i,t+1} = k_t^u(n) + 1 \\ 0, & \text{if } p_{i,t+1} > k_t^u(n) + 1 \end{cases} \tag{2.30}$$

$$g_t^u(n) = \begin{cases} 1, & \text{if } x_t^u(n) = 1 \text{ and } k_t^u(n) + w_t^u(n) \geq y \\ 0, & \text{otherwise} \end{cases} \tag{2.31}$$

and for vehicles on lane, the arrival $\tilde{w}_t^u(n)$ and departure $\tilde{g}_t^u(n)$ are given by

$$\tilde{w}_t^u(n) = \begin{cases} 1, & \text{if } g_t^v(n_d) = 1 \text{ and } X_1 = 1, X_2 = 1 \\ 0, & \text{otherwise} \end{cases} \tag{2.32}$$

$$\tilde{g}_t^u(n) = g_t^u(n). \tag{2.33}$$

Some notes in (2.32) are given in detail. For the traffic flows before entering the link, assign $d = -1, d = 0$, and $d = 1$ to the left, straight forward, and right directions, respectively. n_d is the traffic lane in the adjacent intersection v releasing vehicles to lane n. After vehicle entering the link, vehicle directions are randomly distributed again as the proportion predetermined. In detail, according to the direction proportions allocated by (2.17) in lane choice mechanism, the judgments of X_1, X_2 are binary variables in the following conditions. $X_1 = 1$ means that the direction of vehicle on lane n_d points at the entering link, and $X_1 = 0$ otherwise (it is possible for straight and right-turn sharing one lane). Meanwhile, in the link, $X_2 = 1$ is for the selection of lane n of being left-turn or straight-right lane, and $X_2 = 0$ for the alternate. See Fig. 2.9.

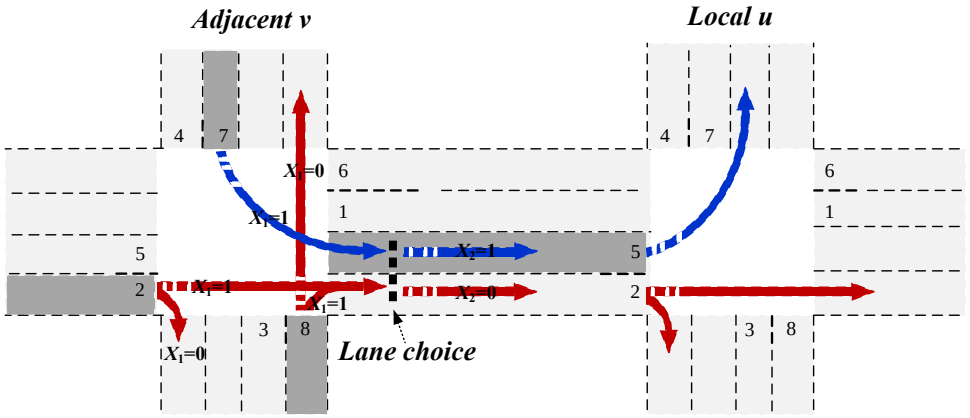


Figure 2.9: Explanation of X_1, X_2 in the case that $n = 5$ $(n \in u)$ and $n_d = 7, 2, 8$ $(d = -1, 0, 1.\ n_d \in v)$

In addition, by reason of different capacities of links, we unify the $\tilde{k}^u_{t+1}(n)$ to be multiplied by $L_{\min}/L_l$.

From the discussion above, we can see the difference between the independent and coordinated intersection control in this study. The coordinated intersections use the arriving information not only on the queue (independent) but also on the lane. Thus, the system state is affected by the joint action of adjacent intersections.

According to the definition of tunable system state for traffic network signal control, the immediate cost function r_t is defined by

$$r_t = \sum_{m=1}^{M} \sum_{n=1}^{N} (e k^m_{t+1}(n) + (1-e)\tilde{k}^m_{t+1}(n)). \tag{2.34}$$

Thus the objective of optimization for traffic signal control at network can be determined by expected total reward function in (2.27).

2.5 Summary

This chapter mainly makes the dynamic modeling for the adaptive traffic signal control system. Three parts are presented. Firstly, beside the introduction of the fixed and variable phase sequence mode, we especially propose the adaptive phase sequence (APS) mode for traffic flow organization. APS has total 12 phase possibilities, one of which is waited for the proper choice by phase optimization, according to traffic conditions in real time.

It is suggested that the organization of APS will be more adaptive than FPS and VPS. Secondly, the signal control model of an isolated intersection is constructed based on the framework of MDP in discrete time. The characteristics including traffic state, traffic action, state transition, and reward function are formulated. Two formulations of state transition are proposed. One is the stochastic state transition with probability transition function. The other one is the determined state transition which works by the way of Monte Carol sampling in simulation. Lastly, considering the modeling at network case, the network loading model and multiagent based signal control model are presented. In network loading, a new vehicle-follow model is proposed as the underlying traffic model for updating system. The signal control at network is based on the multiagent MDP framework. For traffic network coordination, the tunable system state with components of queue length and total number of vehicles on lane, is proposed for this control mechanism.

CHAPTER 3

METHODS STUDIED BASED ON DYNAMIC PROGRAMMING

3.1 Introduction

To solve MDP problem, there are two types of classical DP based methods. One is the backward DP algorithms [29], such as value iteration algorithm and policy iteration algorithm. The other one is the forward search algorithms, such as A* [122] and real-time DP algorithm [123]. We try to use two DP related algorithms to solve traffic signal control problems and discuss the feasibilities of these DP based algorithms.

The first algorithm we proposed is based on a backward DP. It is the value iteration algorithm with stochastic traffic state transitions. The objective is to obtain an optimal stationary policy when state value converges. The second algorithm is a forward search algorithm based on A*. It works with deterministic state transitions under a decision tree and obtains the optimal policy in this shortest path problem. These two algorithms are applied to stochastic states system and deterministic states system, and the related works are published in [124, 56], respectively. The difference between stochastic state and deterministic state is that the former considers the state transition probability function while the latter uses the deterministic state transitions in a Monte Carlo simulation. Whatever,

they both belong to stochastic traffic problems. We have already discussed this in Section 2.3, Chapter 2.

The proposed algorithms can realize the adaptive traffic signal control in some cases. However, some drawbacks still appear as the limitations of DP. We will present these algorithms in detail and a case study of each algorithm is discussed. Based on related DP methods and their applications, then, we will abstract some limitations of DP.

3.2 Backward DP algorithm for control and analysis

3.2.1 DP introduction

For a control problem defined on time series, DP can decompose the problem into stages, which correspond to successive discrete epochs on time series, as shown in Fig. 3.1.

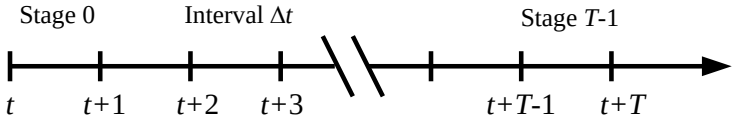


Figure 3.1: Time series and stages

Recall the MDP definition mentioned in Section 2.3.1. Solving a control problem modeled as MDP is equivalent to finding an optimal policy π^* (a mapping from states to actions) to minimize the value function of each state. With initial state s_0 following the optimal policy π^* during horizon T, the optimal value function is defined by

$$J(s_0) = \min_{a_t \in A} E\left\{\sum_{t=0}^{T-1} \gamma^t r_t\right\}. \tag{3.1}$$

The DP solution by following Bellman's equation recursively computes (3.1). That is

$$J(s_t) = \min_{a_t \in A} E\left\{r_t + \gamma J(s_{t+1})\right\}, \text{for } t = 0, 1, \ldots, T-1, \tag{3.2}$$

where decision a_t is selected from a finite set of A at each time step t, and the expectation operator is taken in respect to the probability in state transition from s_t to s_{t+1} with decision a_t.

The optimal deterministic action a_t^* at each time t can be greedily calculated by the minimum value $J(s_t)$. Thus, we have

$$\pi^*(s_t) = a_t^* = \arg\min_{a_t \in A} E\left\{r_t + \gamma J(s_{t+1})\right\}, \tag{3.3}$$

where argmin means the argument of the minimum. It returns the action that minimizes the value of state.

Finite and infinite DP

A finite DP problem is said to have a finite horizon T if the value function J accumulates over a finite number of steps, which can be expressed as

$$J(s_0) = \min_{a_t \in A} E\left\{J(s_T) + \sum_{t=0}^{T-1} \gamma^t r_t\right\}. \tag{3.4}$$

Often we simply use $J(s_T) = 0$, because we are primarily interested in what to do now, given by a_0, or in projected activities over some horizon $t = 0, 1, \ldots T-1$. Problems of this sort often bear interest of achieving optimization over a specific horizon. A good example of such is the shortest path problem.

Similarly, an infinite DP problem is said to have an infinite horizon T if the value function J accumulates over an infinite number of steps, which can be expressed as

$$J(s_0) = \min_{a_t \in A} E\left\{\sum_{t=0}^{\infty} \gamma^t r_t\right\}. \tag{3.5}$$

The infinite horizon problem is of particular interest to understand steady-state properties in Markov process. At steady-state, state transition probabilities become time invariant, and the value of J converges. Solving the infinite horizon problem, it requires an iteration algorithm that leads to convergence in values of J and a stopping criterion that specifies the region of convergence. There are two common iteration algorithms: value iteration and policy iteration. The two algorithms are discussed later.

Stochastic and deterministic DP

A stochastic (or probabilistic) DP problem is that the state is subsequently transfered with probability by the state and decision at the current step, as shown in Fig. 3.2. The state

transition probability function is $p(s_t, a_t, s_{t+1})$, $p : S \times A \times S \to [0, 1]$. Since S is finite state space, the Bellman's equation in (3.2) for a stochastic problem is explicitly rewritten as

$$J(s_t) = \min_{a_t \in A} \sum_{s_{t+1} \in S} p(s_t, a_t, s_{t+1}) \left[r_t + \gamma J(s_{t+1}) \right]. \tag{3.6}$$

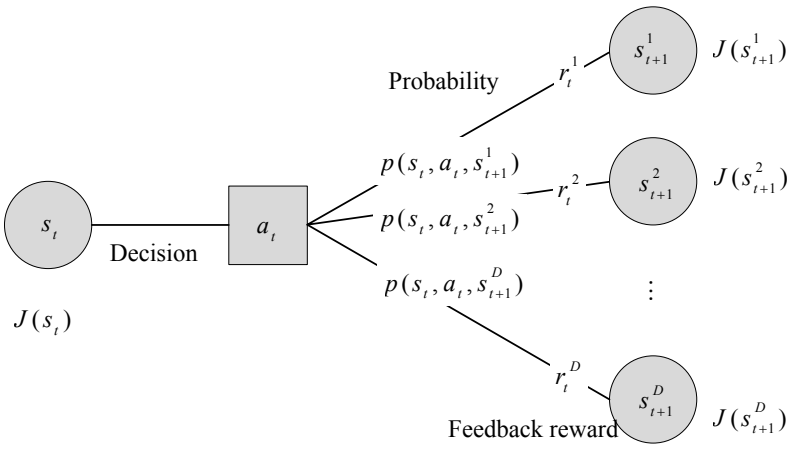


Figure 3.2: A sample of state transition in stochastic dynamic programming

A deterministic DP problem is that the state at the next step is completely determined by the state and decision at the current step. Standard formulation takes this into account by using transfer function $\sigma(s_t, a_t)$ instead of a transition probability to specify the next state. That is to say, in MDP the transition probability function is defined by

$$p(s_t, a_t, s_{t+1}) = \begin{cases} 1, & \text{if } \sigma(s_t, a_t) = s_{t+1} \\ 0, & \text{if } \sigma(s_t, a_t) \neq s_{t+1}. \end{cases} \tag{3.7}$$

The Bellman's equation in (3.2) for a deterministic problem can be rewritten as

$$J(s_t) = \min_{a_t \in A} \left\{ r_t + \gamma J(s_{t+1}) \right\}. \tag{3.8}$$

In a deterministic problem, sometimes the exogenous stochastic information w_t adds to the system so that the transfer function is $\sigma(s_t, a_t, w_t)$. In this case, the process of s_t can be seen as a stochastic process with Markov property, i.e., a Markov process. This usually appears in Monte Carlo simulation. We may rewrite (3.8) as

$$J(s_t) = \min_{a_t \in A} E_{w_t} \left\{ r_t + \gamma J(s_{t+1}) \right\} \tag{3.9}$$

where w_t is the exogenous information or noise. In traffic model, it is the random traffic arrival information.

Value iteration and policy iteration

In an infinite horizon DP problem, the steady-state in value convergence can be obtained by iteration algorithms. We can think of a steady-state problem as one without the time dimension. In stochastic DP formulation, the steady-state optimality equations can be expressed as

$$J(s) = \min_{a \in A} \sum_{s' \in S} p(s, a, s') \left[r + \gamma J(s') \right]. \tag{3.10}$$

Correspondingly, a policy is called stationary if it does not change over time in steady-state. Normally an infinite horizon DP problem is to determine an optimal stationary deterministic policy $\pi^*(s)$, $\forall s \in S$ in (3.10), which refers to the MDP optimal policy in (3.3) under the optimality criterion of expected total discounted reward. Value iteration and policy iteration are the two common DP algorithms to achieve the MDP optimal policy.

Value iteration It is the most widely used algorithm in DP. It involves iteratively estimating the value function. At each iteration the estimate of value function determines which decisions we will make and as a result defines a policy. The basic version of value iteration algorithm is given in Fig. 3.3.

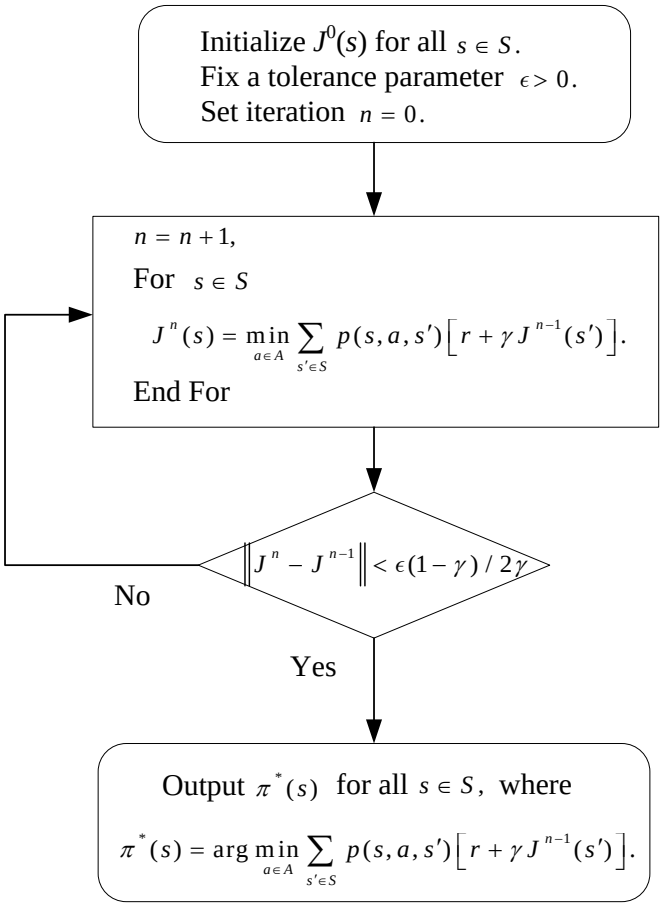


Figure 3.3: Value iteration of dynamic programming

Policy iteration It has two iterative processes, i.e., policy evaluation and policy improvement. Given a certain policy π, policy evaluation tries to approximate the values of each state under this policy. The values of each state are the inputs to policy improvement process. The purpose of policy improvement process is to adjust the policy according to new state values. The basic version of policy iteration algorithm is given in Fig. 3.4.

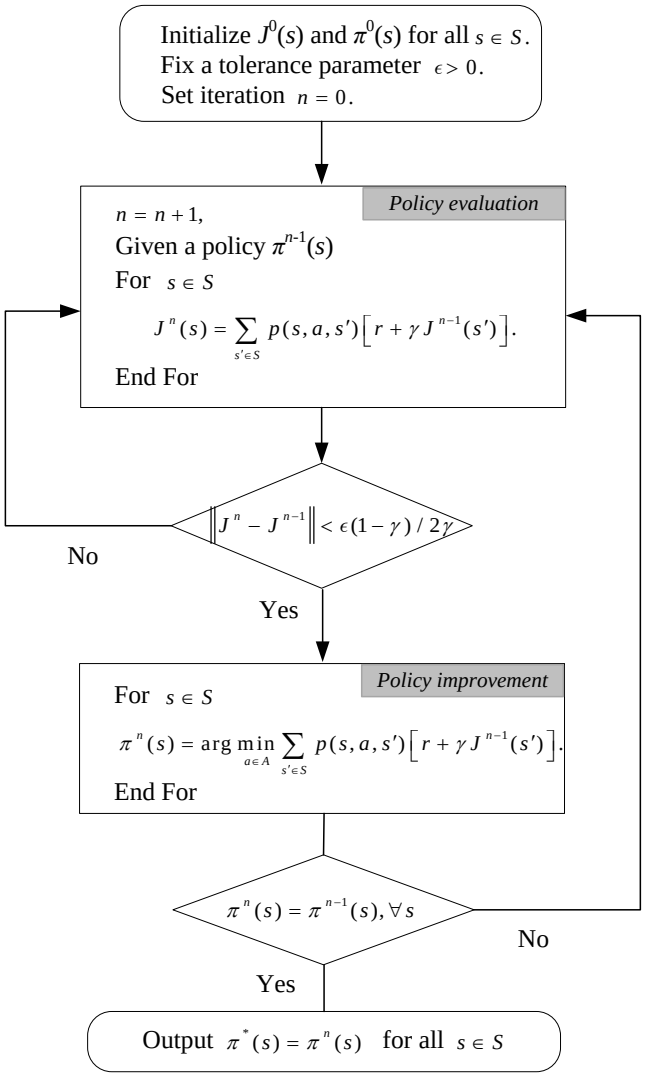


Figure 3.4: Policy iteration of dynamic programming

By comparing value iteration and policy iteration, it can be seen that both policy evaluation and policy improvement need to visit each state multiple times and are computationally inefficient. On the contrary, value iteration method effectively integrates policy evaluation and policy improvement and has better computational efficiency. Next, we will study a simple traffic signal control problem by using value iteration algorithm.

3.2.2 Value iteration algorithm for stochastic states system

Based on the MDP formulation of traffic signal control problem and the conventional DP algorithm, we will investigate a control method for a simple two-phase signalized intersection by using value iteration algorithm.

Consider a simple two-phase isolated signal intersection, as shown in Fig. 3.5. For sake of readability, the turn-left and turn-right movements are not presented.

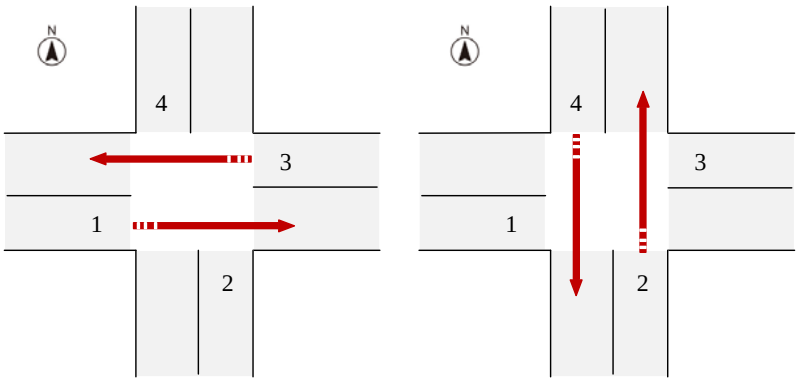


Figure 3.5: A simple two-phase intersection

The intersection controlled by two-phase signal provides green signal to the E-W and N-S approaches alternately, as shown in Fig. 3.6. To avoid interference between antagonistic movements, the inter-green interval g_{int} (all red) is necessary, set by $g_{\text{int}} = 1$ int. Moreover, the minimum green time $g_{\min}$ for each phase is predetermined and set by $g_{\min} = 3$ ints. The incremental interval each step is $\Delta t = 2s$. Other model assumptions can be seen in Section 2.3.2.

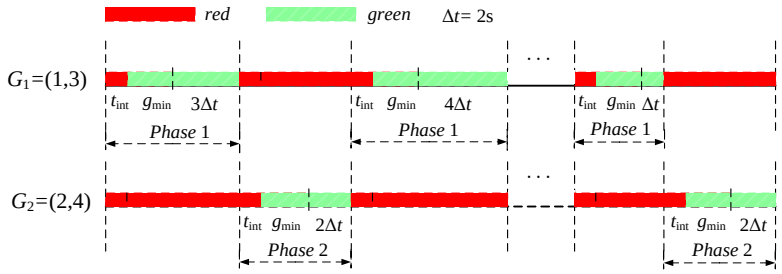


Figure 3.6: Example of two-phase traffic signal planning

In DP algorithm, the obtained state values need to traverse all state space and these values are stored in a look-up table. We propose an idea of decomposition MDP that

makes probability transition matrix into several small matrices related to two signal rules. One rule is the fixed phase sequence (Phase 1 and 2) and the other rule is the naturally constant sequence of signal indications (e.g., in each phase signals indicate as sequence of all-red, minimum green, green extension, and red). Thus, the decomposition method only calculates the relative states owning the available probabilities between the small matrices or in its local matrix, and avoids to calculate all the transition probabilities of the states (the properties of signal status and the number of vehicles) in one tremendous transition matrix. This idea refers to the following traffic state transition model.

At first, some notations are defined as follows.

- G_c, is the c^{th} $(c = 1, 2)$ traffic flow combination;
- G_1R_2, the green signal for G_1 and the red for G_2;
- R_1G_2, the red signal for G_1 and the green for G_2;
- R_1R_2, the red signals for the whole intersection;
- $G_{1\min}R_2$, minimum green for G_1 and the red for G_2;
- $R_1G_{2\min}$, minimum green for G_2 and the red for G_1;
- a_{ij}, the action to switch the phase from G_i to G_j, where $i, j = 1, 2$;
- J_ζ, the value set with the same dimensionality of vehicle states under each signal state, $\zeta = 1, 2, ..., 6$.

The traffic state transition model is shown in Fig. 3.7. In this model, the transition states are organized by two layers in the structure. The first layer contains the six traffic signal states which are described by circles. The second layer that is covered by the first one, consists of the same structures of vehicle states under each signal state. Thus, the state transition should be considered in two aspects, namely the signal state transition and the vehicle state transition.

It is obviously that there are two big loops among the six signal states. One is the state transition which is marked by the solid arrow lines and the other is the value transmission marked by the dashed arrow lines. In state transition loop, the vehicle state is changed under the guide of signal state transition. Moreover, the transfered vehicle states under the current signal are just relevant to the successors in the following signal state. In value

transmission loop, the value of the current state is backward of the next state, and it acts as the evaluation for next step according to the value function. In simplified way, the unique and deterministic action between two signal states is not presented.

Notice that two small loops are attached to signal state G_1R_2 and R_1G_2, respectively. The signal states in these cases have alternative choice to extend the intervals during current green phase or switch signal to another one. Actually, all signal states are transformed sequentially between two phase groups. By iteratively calculating the value of each state in time sequence, stationary optimal policy can be obtained.

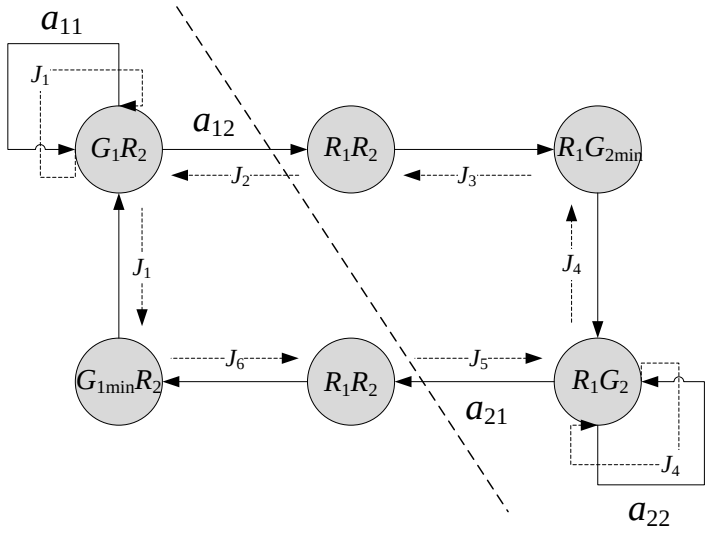


Figure 3.7: Traffic state transition model in decomposition MDP

By using the MDP model definition and DP algorithm, the traffic signal control problem can be solved. The basic characteristics of MDP for the modeling are introduced in Section 2.3.2. In the model, probability transition matrix is generally hard to find. In order to obtain the matrix, it is not necessary to search all the elements, i.e., probabilities in the matrix. Only relevant vehicle state transitions are marked, available probabilities can be obtained. In the traffic state transition model, both of the signal state layer and the vehicle state layer make sequential transitions, and the former is illustrated evidently in Fig. 3.7. In the vehicle state layer, all ordered states own the same properties of each traffic signal. Once state transitions are determined, the probabilities from s to s' can be calculated by transition probability function defined in (2.11) and (2.12) and then they

Algorithm 1 Value iteration algorithm for traffic signal control based on decomposition of MDP

Initialization of states $s = (k, x)$, $s \in S$, value function $J^0(s) = \mathbf{0}$;
Set iteration $n = 0$;
repeat
    $n \leftarrow n + 1$;
    for all traffic signal states $x_\zeta, (\zeta = 1, 2, \ldots, 6)$ **do** ▷ Transition sequence of x_ζ is based on the state transition model in Fig. 3.7;
        allocate all vehicle states k_ζ to x_ζ;
        for all state $s \in S$ and action $a \in A$ in x_ζ **do**
            calculate $J_\zeta^n(s) = \min\limits_{a \in A} \sum\limits_{s' \in S} p(s, a, s') \left[r + \gamma J_\zeta^{n-1}(s') \right]$;
        end for
    end for
    $J^n \leftarrow ((J_1^n), (J_2^n), \ldots, (J_6^n))$;
until $\|J^n - J^{n-1}\| < (1 - \gamma)/2\gamma$ is satisfied;
For all $s \in S$ in x_ζ, calculate the stationary optimal policy according to $\pi_\zeta^*(s) = \arg\min\limits_{a \in A} \sum\limits_{s' \in S} p(s, a, s') \left[r + \gamma J_\zeta^{n-1}(s') \right]$;
$\pi^*(s) \leftarrow (\pi_1^*(s), \pi_2^*(s), \ldots, \pi_6^*(s))$.

are recorded into the transition matrix. Based on the value iteration algorithm of DP, the traffic signal control optimal policy can be obtained by Algorithm 1.

3.2.3 Case study and analysis

The proposed algorithm is implemented to the case of two-phase isolated intersection illustrated in Fig. 3.5. The traffic demand data is generated by the uniform random data 0 and 1, using Inverse Transformation Method (ITM) which meets a Bernoulli probability distribution. Meanwhile, the "heaviest" flow rate q_n in the traffic combination G_c determines the contribution to the overall workload, which is called relative traffic load ρ and defined by

$$\rho = \sum_c \max_{n \in G_c} \{q_n\}, \tag{3.11}$$

where n is the traffic flow that $n = 1$, 2, 3, and 4 in the case.

Table 3.1: Simulation results of average traffic delays

Relative traffic load		$\rho = 0.2$		$\rho = 0.4$		$\rho = 0.6$		$\rho = 0.8$	
Traffic volume by ITM (veh/h)		(181, 163, 187, 172)		(368, 362, 383, 346)		(526, 569, 524, 521)		(717, 708, 725, 725)	
FC		4.06	0	5.68	0	8.36	0	17.64	0
ADC	(delay/s & reduction/%)	3.72	-8.4	5.17	-9.0	7.38	-11.7	16.30	-7.6
MDP		3.26	-19.7	4.64	-18.3	6.07	-27.4	12.85	-27.2

The performance measures are average traffic delay at whole intersection and vehicle queue length in each traffic flow combination. In simulation, we compare the proposed algorithm by MDP with the fixed-time control (FC) and actuated control (ADC).

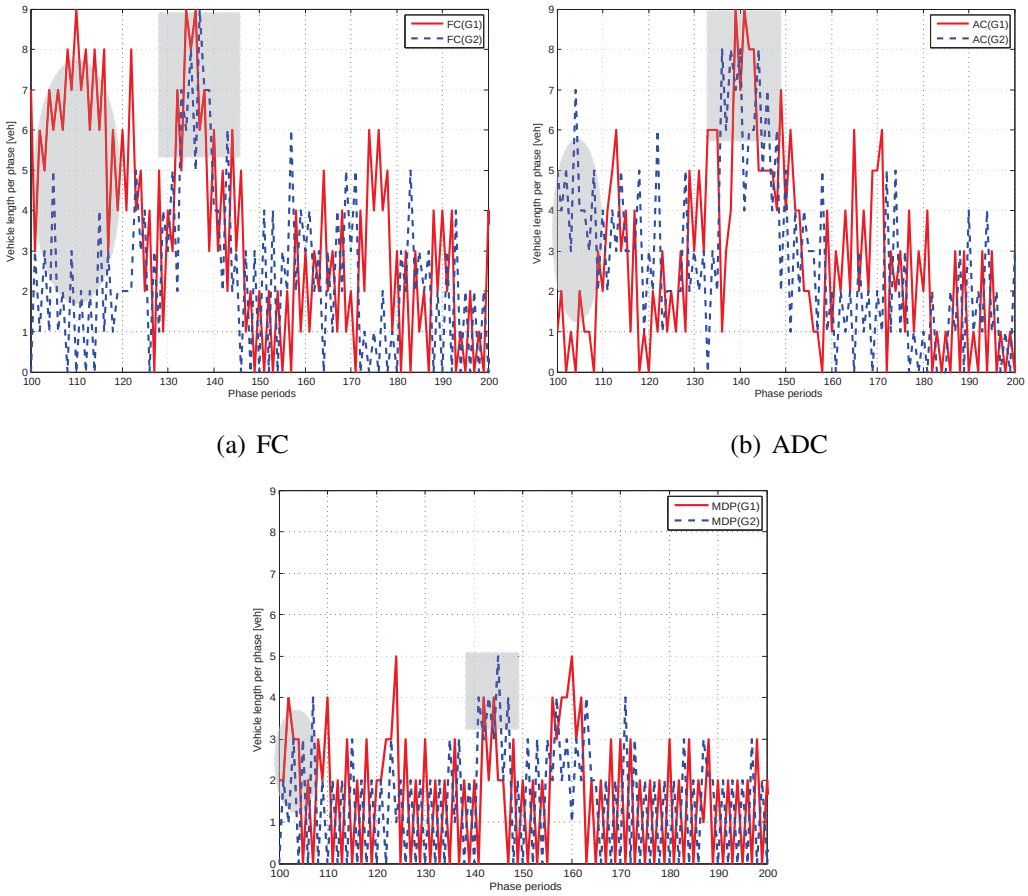
In Tab. 3.1, the comparisons of average traffic delay in these three control methods are given. From the traffic scenarios with different relative traffic loads, obviously, we see that MDP outperforms FC and ADC with less delays. Especially in high traffic load (ρ=0.8), MDP has about 27.2 % reductions from FC. It is much better than ADC, which has only about 7.6% reductions from FC.

In Fig. 3.8, the comparisons of vehicle length (represented by the maximum one in traffic flow combinations) in these three control methods are given. Obviously, MDP can control the vehicle length efficiently in each phase period. The shadow areas in figures indicate that the vehicle length in MDP is relatively less and more stable than those in FC and ADC.

However, the proposed algorithm based on MDP is offline. When the steady-state is obtained in the value convergence, we use Monte-Carlo sampling of the arrival information to calculate the system state, then the corresponding optimal control decision in the stationary policy is implemented. The iteration convergence often takes much time, especially in large state space. In our case of two-phase signal control with 6 signal state divisions, assuming the largest vehicle length of each lane is $L = 19$, the state space is 6×20^4, and the computation time for convergence is list in Tab. 3.2. Moreover, when traffic arrival rate changes, the steady-state is different and needs to be calculated again. Therefore, this algorithm is impractical for real-time traffic signal control problem. Next, we will present another algorithm, which uses the forward search algorithm based on DP and can solve the mentioned problems to realize the control for a more complex case in real-time.

Table 3.2: Computation of convergence in value iteration algorithm for solving MDP

Relative traffic load	ρ=0.2	ρ=0.4	ρ=0.6	ρ=0.8
Iterations	62	75	88	95
Computation time (h)	2.05	2.52	2.97	3.23



(a) FC

(b) ADC

(c) MDP

Figure 3.8: Vehicle length comparisons in three control methods ($\rho = 0.6$)

3.3 Forward search algorithm for control and analysis

3.3.1 Forward search A* introduction

Without evaluating the complete state space in conventional backward DP algorithm, some algorithms use the knowledge of start state to focus computation on just those states

that are reachable from the start state by following an optimal policy. Subsequently, the forward search (heuristic search DP) algorithms based on state accessibility were proposed, such as A* [122, 125], LAO* [126], real-time DP[123]. The previous study of these algorithms are proved that they can improve computational efficiency and save much computation time, which seems possible to be implemented for real-time traffic signal control. In the fundamental work of the thesis, we propose a new adaptive traffic signal control algorithm based on the forward search A* algorithm.

Thus, in this part, we briefly introduce the classical A* algorithm, which is the basis for the forward search traffic signal control algorithm presented in the next part.

The A* algorithm was originally presented by Hart et al. [122]. It was designed to solve the shortest path problem between an origin and a destination. As A* traverses the graph, it builds up a tree of partial paths. The leaves of this tree (called the open set or fringe) are stored in a priority queue that orders the leaf nodes by a cost function, which combines a heuristic estimate of the cost to reach a goal and the distance traveled from the initial node. Specifically, in each node $\hat{n}$ the cost function $f(\hat{n})$ is

$$f(\hat{n}) = g(\hat{n}) + h(\hat{n}) \tag{3.12}$$

where $g(\hat{n})$ is the known cost from initial node to $\hat{n}$, and $h(\hat{n})$ is a heuristic estimate of the cost from node $\hat{n}$ to any goal node. For the algorithm to find the actual shortest path, the heuristic function must be admissible, meaning that it never overestimates the actual cost to get to the nearest goal node. The heuristic function is problem-specific and must be provided by the user of the algorithm. Sometimes, it is difficult to find a good $h(\hat{n})$. If the heuristic function $h(\hat{n}) = 0$ for all nodes $\hat{n}$, then A* is essentially the same as the Dijkstra's shortest path algorithm.

From [127] by Nils Nilsson, we list the pseudocode of A* algorithm as follows:

Pseudocode

1. Create a search graph $\mathcal{G}$, consisting solely of the start node $\hat{n}_0$. Put $\hat{n}_0$ on a list called OPEN.

2. Create a list called CLOSED that is initially empty.

3. If OPEN is empty, exit with failure.

4. Select the first node on OPEN, remove it from OPEN, and put it on CLOSED. Call this node $\hat{n}$.

5. If $\hat{n}$ is a goal node, exit successfully with the solution obtained by tracing a path along the pointers from $\hat{n}$ to $\hat{n}_0$ in $\mathcal{G}$. (The pointers define a search tree and are established in Step 7.)

6. Expand node $\hat{n}$, generating the set, $\mathcal{M}$, of its successors that are not already ancestors of $\hat{n}$ in $\mathcal{G}$. Install these members of $\mathcal{M}$ as successors of $\hat{n}$ in $\mathcal{G}$.

7. Establish a pointer to $\hat{n}$ from each of those members of $\mathcal{M}$ that were not already in $\mathcal{G}$ (i.e., not already on either OPEN or CLOSED). Add these members of $\mathcal{M}$ to OPEN. For each member, $\hat{n}'$, of $\mathcal{M}$ that was already on OPEN or CLOSED, redirect its pointer to $\hat{n}$ if the best path to $\hat{n}'$ found so far is through $\hat{n}$. For each member of $\mathcal{M}$ already on CLOSED, redirect the pointers of each of its descendants in $\mathcal{G}$ so that they point backward along the best paths found so far to these descendants.

8. Reorder the list OPEN in order of increasing f values. (Ties among minimal f values are resolved in favor of the deepest node in the search tree.)

9. Go to Step 3.

In the pseudocode above, the part of step 7 is to find the shortest path by using pointers. Step 7 is often not implemented. Some of these pointers will ultimately be redirected in any case as the search progress. In Section 3.3.2, our proposed forward search algorithm for traffic signal control will give a labeled position method to search backward along the best paths.

In order to understand well about the solution procedure of A* algorithm, we introduce a simple example of shortest path problem.

Example Given a weighted, directed graph $\mathcal{G}$, find the shortest path from the start node V0 to the destination node V5 by passing any other nodes V1, V2, V3, and V4.

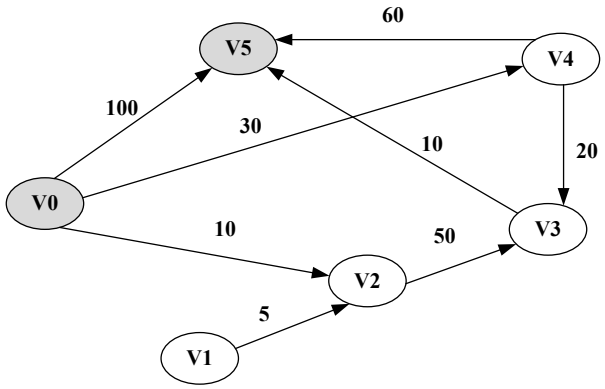


Figure 3.9: An example of weighted directed graph

OPEN	**(V0, 0)**
CLOSED	NULL
OPEN	**(V2, 10)** (V4, 30) (V5, 100)
CLOSED	(V0, 0)
OPEN	**(V4, 30)** (V3, 60) (V5, 100)
CLOSED	(V0, 0) (V2, 10)
OPEN	**(V3, 50)** (V3, 60) (V5, 90) (V5, 100)
CLOSED	(V0, 0) (V2, 10) (V4, 30)
OPEN	**(V5, 60)** (V5, 90) (V5, 100)
CLOSED	(V0, 0) (V2, 10) (V4, 30) (V3, 50)
OPEN	NULL
CLOSED	(V0, 0) (V2, 10) (V4, 30) (V3, 50) (V5, 60)

Figure 3.10: A* solution for shortest path problem

In the list, the cost value $f(\cdot)$ of each node is written with the node together. When the OPEN list is empty, the CLOSED list obtains the shortest path from V0 to any nodes in the graph.

3.3.2 Forward search algorithm for deterministic states system

In this part, we introduce an algorithm for adaptive traffic signal control problem, formulated by MDP with deterministic state transitions. This algorithm is a forward search based on A* algorithm. It is shorten by FSDP in the thesis.

In FSDP algorithm, there are three important processes. The first one is that FSDP under a decision tree has state variables constrained by the planning time where multi-step iteration of mandatory transformation is executed. The second one is the process optimization of repeated or invalid traffic states. the repeated states will be cutting off and the next states in leaf nodes will not appear any more. Those will reduce a mass of calculations of irrelevant states. Moreover, the solve-labeling procedure named labeled position rule is included in the algorithm to improve efficiency after reaching the goal state. Normally, in the planning period, it is often difficult to find the best solution in each step although the optimum value of the goal state is obtained. The labeled position method we proposed is a good way to solve this problem.

In order to verify the efficiency of FSDP, we consider the typical traffic intersection given in Fig. 2.1 and the traffic flow organization patterns by using phase sequence FPS, VPS, and APS. Notice that the application refers to the case of deterministic state transition for the traffic problem with stochastic arrivals. The basic MDP traffic model is no more expressed here and some donations are additionally given.

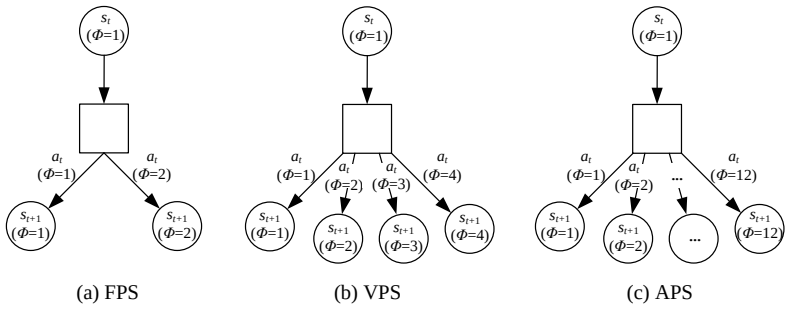
We denote Φ to the numbered phase. According to FPS, VPS, and APS modes in Section 2.2, we define the values Φ in Tab. 3.3.

Table 3.3: Phase numbered in FPS, VPS, and APS

$\Phi(\Phi = c)$	1	2	3	4	5	6	7	8	9	10	11	12
FPS(G_c)	(1,5)	(2,6)	(3,7)	(4,8)								
VPS(G_c)	(1,5)	(2,6)	(3,7)	(4,8)								
APS(G_c)	(1,4)	(1,5)	(1,6)	(2,5)	(2,6)	(2,7)	(3,6)	(3,7)	(3,8)	(4,7)	(4,8)	(5,8)

According to the definition of binary signal state variable x in (2.8), we know that each phase Φ corresponds to the x vector, e.g., $\Phi = 1$: $x = (1, 0, 0, 0, 1, 0, 0, 0)^{\mathrm{T}}$. Thus, the original definition of state $s = (k, x)$ can be rewritten as $s = (k, \Phi)$, where k is the vehicle queue length state.

In deterministic problem, the state s_t is transfered to the following state s_{t+1} definitely with decision a_t. In different traffic phase sequence mode, the next phase decision is different. Using circle as state node and rectangle as decision node, three types of these phase decisions are illustrated in Fig. 3.11.



(a) FPS (b) VPS (c) APS

Figure 3.11: Illustration of phase decision under phase modes of (a) FPS (2-connectors), (b) VPS (4-connectors), and (c) APS (12-connectors)

Assume that we can know the traffic arrival information in the near future. It includes two parts. One is that the upstream roadside sensors provide the exact information of arriving traffic of the next certain seconds. The other part is to use some predicted model to generate the estimated arrivals. The arrival information and decision making in real-time refer to the rolling horizon approach, which we will discuss in this chapter. Here, we make the knowing entire decision horizon T_p, which can roll ahead to the all simulation period. The queue lengths of each lane and the signal state (using Φ) at any stage can be obtained by using the traffic model. The optimization goal is to find an optimal control strategy consisting of a sequence of actions $\pi = \{a_0, a_1, \ldots, a_{T_p-1}\}$ that minimize the utility (cost) function. Based on the initial signal state, queue length of each lane, and future traffic arrival information, the entire decision process can be illustrated by decision tree. A simple phase sequence using FPS mode in the decision tree is shown in Fig. 3.12. Then, some discussions and conclusions are carried out surrounding FPS, and can also extend to the cases of VPS and APS.

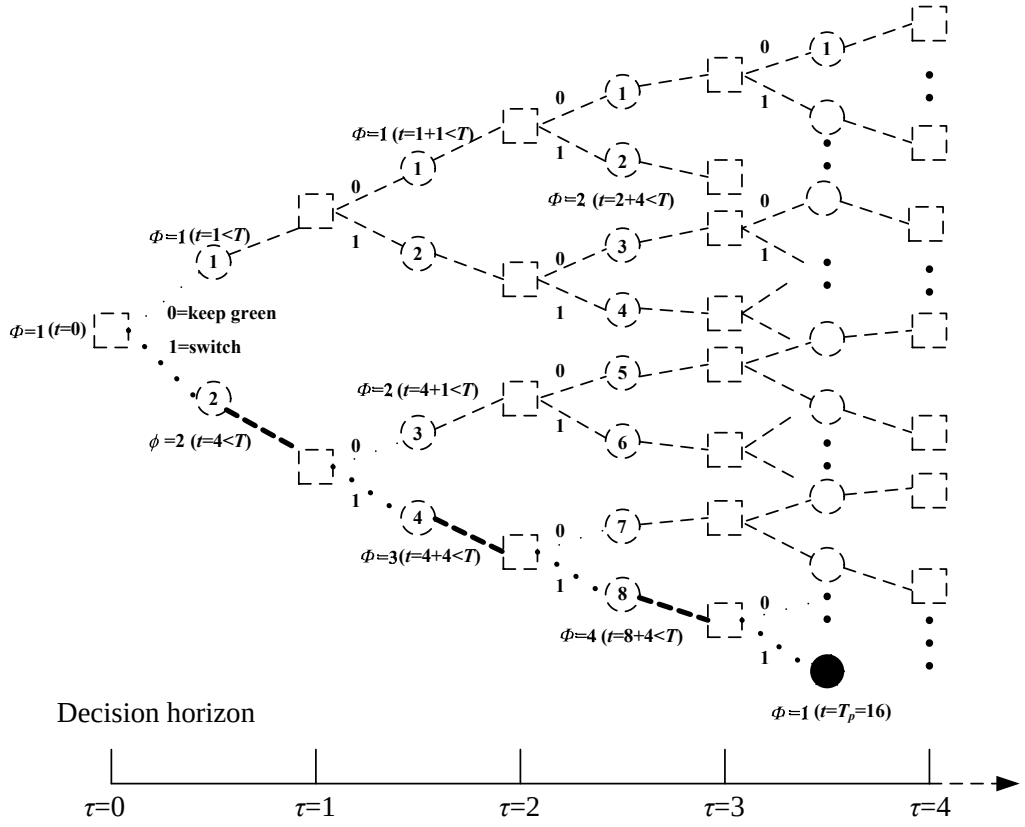


Figure 3.12: Illustration of traffic signal control under decision tree

We construct an explicit graph with FPS mode that initially consists only the start state. A tip or leaf state signed by solid black circle is said to be terminal; otherwise, it is said to be nonterminal. A nonterminal tip state can be expanded by adding to the explicit graph its outgoing b-connectors (one for each action), e.g., 2-connectors in FPS (0 for keeping signal green and 1 for switching). In our case, set the planning horizon T_p=16 intervals. The sum of inter-green and minimum green time is 4 intervals after phase changed. Thus, in the planning horizon, all phases can be changed alternately at most one time, which is shown in the bold black route. Note that when the states in mandatory intervals are omitted, the decision step τ is not the same meaning with the normal planning step t. In the following part, we use τ related to the step and t related to time.

In every step (or stage), the number of state variables will increase in geometrical progression. The whole tree in the last step, will possess 2^{16} states if the constraints for pruning are unconsidered. It will become a large scale programming problem. In our study, there are two ways to reduce the number of states. At first, when the traffic signal

is switched, the current phase adds one unless it has already been phase 4 and will return to phase 1 cyclically. This will take four mandatory intervals, including one inter-green interval and three minimum green intervals. So when the total time equals to the planning period T_p, it will not create new branch anymore. Meanwhile, the information and the evaluation value of the state will be in storage. Secondly, the same states will be merged and the best strategy of the current interval can be made based on the evaluation function. It is equivalent to cutting off the identical states owning the non-optimal cumulative evaluation values, as well as the successor states going to be produced. Note that the judgment of the values of same states must have the same time properties, namely the total planning time has already been implemented. That's because the multi-step iteration processing of the mandatory intervals makes the evaluation level different from possessions accumulated step by step. Hypothetically, the states of mandatory execution will virtually add to the queue list of the comparing states with the same steps.

A state in circle is labeled solved if it is a terminal state or if all of its subsequent states are labeled solved. Labeling procedure can improve the efficiency in search of the states which have the optimal evaluation values, because it is unnecessary to search below a solved state for other sub-optimal evaluations. State position acts as the label for each state.

With outgoing 2-connectors, there will be 2^τ states positions in decision step τ. The state position $\hat{\boldsymbol{p}}(\tau) = (\hat{p}_i(\tau), i \in [1, 2^\tau])^{\mathrm{T}}$ of step τ can be classified as the following recursion formula. Assume that each state owns its particular location no matter whether the states are the same or not.

$$\begin{cases} \hat{\boldsymbol{p}}(\tau+1) = (2\hat{\boldsymbol{p}}(\tau) - 1, 2\hat{\boldsymbol{p}}(\tau)) \\ \hat{\boldsymbol{p}}(0) = 1 \\ \tau \in [0, T_p - 1], \tau \in \mathbb{N}. \end{cases} \tag{3.13}$$

It is clear that really not all the states own their positions due to the time constraints and the repeated states. The time t should satisfy the following condition:

$$t(\tau+1) = \begin{cases} t(\tau) + (g_{\mathrm{int}} + g_{\min}), & \text{if switch signal}, t(\tau) \le T_p - (g_{\mathrm{int}} + g_{\min}) \\ t(\tau) + 1, & \text{if keep green}, t(\tau) \le T_p - 1. \end{cases} \tag{3.14}$$

When the planning is completed ($t = T_p$), the evaluation values of states in the last step will be compared ultimately, and the optimal strategy during the planning period can be confirmed. During one decision τ, the single step forward evaluation function is

defined as

$$J_t = \min_{a_t \in A} E_{w_t}\{r_t + \gamma J_{t-1}\}. \tag{3.15}$$

and the multi-step (the mandatory intervals $t_M = g_{\text{int}} + g_{\min}$) forward evaluation function is defined as

$$J_{t+t_M-1} = \min_{a_t \in A} E_{w_t} \left\{ \sum_{t'=t}^{t+t_M-1} \gamma^{t'-t} r_{t'} + \gamma^{t_M} J_{t-1} \right\}. \tag{3.16}$$

According to (3.15) and (3.16), the optimal strategy a_τ^* of the particular state in the current stage τ can be found. All the values of the available states with the properties of positions and planning time t should be compared. Actually, the states in the comparable states set at different time t own the chance to be branched, but we just choose the minimum one until the goal state ($t = T_p$) is reached. This seems like the A* algorithm. Differently, with the multi-step iteration as the assumption in the case, our method reduces the branching possibilities and reaches the goal state fast. Moreover, we just retain the state of being the optimal accumulative value among the same states at time t. During the multi-step iteration, we should declare that, the states (exclude the start and the end step states) transferred in mandatory intervals are ignored. They are not participated in the process optimization. As for the reduction of the same states, it has no influence on generating the successor states and the planning. Totally speaking, when the whole planning T_p is completed, the policy namely the set of strategies a_τ^* is an optimum policy to reach the final state. For obtaining that, two important properties are given below.

Property 1: Suppose that $\mathcal{L}$ denotes the number of optimal trajectory l^* determined by the optimal cost value J during the planning horizon T_p. There exists $\mathcal{L} \geq 1$.

Proof: The trajectory $l_i(\tau)$ denotes the i^{th} position set of successor states during τ steps ($\tau \leq T - 1$). The $s(\hat{p}_j(\tau), l_i(\tau))$ denotes the traffic state in position $\hat{p}_j(\tau)$ of trajectory $l_i(\tau)$, where $j = 1, 2, \ldots, N_j$ and $i = 1, 2, \ldots, N_i$.

It is easy to know that the optimal state $s(\hat{p}^*(\tau), l^*(\tau))$ and the corresponding position $\hat{p}^*(\tau)$ exist uniquely in each step, according to (3.15) and (3.16) and the pruning constraints in the decision tree. The $\hat{p}^*(\tau)$ belonging to l^* will be discussed in the following two cases.

Case 1: there exists $J\left(\hat{p}_j^*(\tau-1)\right) = J\left(\hat{p}_{\tilde{j}}^*(\tau-1)\right), (j \neq \tilde{j})$, surely know that $s(\hat{p}_j^*(\tau-1), l^*(\tau-1)) \neq s(\hat{p}_{\tilde{j}}^*(\tau-1), l^*(\tau-1))$. Suppose that, with different actions taken to reach the next same states $s(\hat{p}_j^*(\tau), l^*(\tau)) = s(\hat{p}_{\tilde{j}}^*(\tau), l^*(\tau))$, the same one-step rewards are obtained. Eliminate either of two positions (assume this one to be $\hat{p}_{\tilde{j}}^*(\tau)$).

The remaining position $\hat{p}_j^*(\tau)$ is determined by different actions, see Fig. 3.13(a). Consequently, $\hat{p}_j^*(\tau)$ simultaneously belongs to different $l^*(\tau)$, i.e., $l_i^*(\tau)$ and $l_{\tilde{i}}^*(\tau)$,$(i \neq \tilde{i})$. Thus, $\mathcal{L} \geq 1$ is satisfied.

Case 2: there exists $J\left(\hat{p}_j^*(\tau-1)\right) \neq J\left(\hat{p}_{\tilde{j}}^*(\tau-1)\right), (j \neq \tilde{j})$ of different traffic states. Suppose that, with different actions taken to reach the next same states, we have $s(\hat{p}_j^*(\tau), l^*(\tau)) = s(\hat{p}_{\tilde{j}}^*(\tau), l^*(\tau))$, the same one-step rewards are obtained. Reserve the position (assume this one to be $\hat{p}_j^*(\tau)$) which can obtain the optimal cost value, and remove the other one $\hat{p}_{\tilde{j}}^*(\tau)$. Thus, the remaining position $\hat{p}_j^*(\tau)$ is determined by the optimal action. Nevertheless, there may exist $J\left(\hat{p}_j^*(\tau)\right) = J\left(\hat{p}_{\tilde{j}'}^*(\tau)\right), (j \neq \tilde{j}')$, and two different optimal states and the corresponding positions exist, see Fig. 3.13(b). Consequently, $\hat{p}_j^*(\tau)$ and $\hat{p}_{\tilde{j}'}^*(\tau)$ belong to $l_i^*(\tau)$ and $l_{\tilde{i}'}^*(\tau)$ $(i \neq \tilde{i}')$, respectively. Thus, $\mathcal{L} \geq 1$ is satisfied. Actually, in each step, the position belongs to at least one trajectory. Therefore, $N_j \leq N_i$. ■

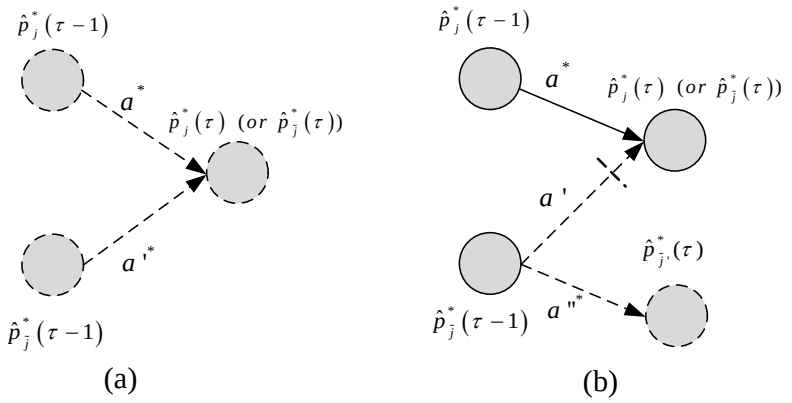


Figure 3.13: Explanation of different cases of the optimal positions (the dash-arrow in (a) means existing the chosen action to the same states, but for simplicity, we don't choose it for next planning any more; the dash-arrow with vertical bar in (b) means not existing the chosen action to the same state, but it may still reach to other state determined by the optimal action.)

As in the proof above, the optimal trajectory is likely not unique. But during the planning horizon, the performances are almost the same due to the same evaluation values. In this global optimization, just one of the optimal trajectories l^* is adopted. Based on this assumption, we have the following property.

Property 2: The optimal state $s(p^*(\tau), l^*(\tau))$ determined by one of the optimal trajectories $l^*(\tau)$ has the position $p^*(\tau)$ which is exclusive as well as the position $p^*(\tau-1)$ of its previous node that is related to the particular previous state.

Proof: The chosen optimal trajectory $l^*(\tau)$ is determined by the optimal position $\hat{p}^*(\tau) \in \hat{\boldsymbol{p}}(\tau)$. According to the rule of position labeled arrangement, after the optimal position $\hat{p}^*(\tau)$ in the planning horizon is found, the position of previous state $\hat{p}^*(\tau-1)$ can be written as:

$$\hat{p}^*(\tau-1) = \begin{cases} \hat{p}^*(\tau)/2, & \text{if } \hat{p}^*(\tau)\text{mod}_2 = 0 \\ (\hat{p}^*(\tau)+1)/2, & \text{if } \hat{p}^*(\tau)\text{mod}_2 = 1. \end{cases} \tag{3.17}$$

Obviously, when the position $\hat{p}^*(\tau)$ we choose is unique (other optimal position has been eliminated in the rule), the position of previous state $\hat{p}^*(\tau-1)$ is exclusive, and the optimal trajectory l^* chosen is uniquely determined. ■

Corresponding to this proof, the action also can be derived backward as

$$u^*(\tau) = \begin{cases} 1, & \text{if } \hat{p}^*(\tau)\text{mod}_2 = 0 \\ 0, & \text{if } \hat{p}^*(\tau)\text{mod}_2 = 1. \end{cases} \tag{3.18}$$

Consequently, the optimal phase can also be obtained successively while the action $u^*(\tau)$ is occurred, namely the phase is unchanged as $u^*(\tau) = 0$ and plus one as $u^*(\tau) = 1$.

Actually, after we obtain all the optimal position during planning horizon T_p, with the initial phase setting $\Phi(\hat{p}(0))$, the optimal phase can be found as follows:

$$\Phi(\hat{p}^*(\tau+1)) = \begin{cases} \Phi(\hat{p}^*(\tau)), & \text{if } \hat{p}^*(\tau+1) = 2\hat{p}^*(\tau) - 1 \\ \Phi(\hat{p}^*(\tau)) + 1, & \text{if } \hat{p}^*(\tau+1) = 2\hat{p}^*(\tau). \end{cases} \tag{3.19}$$

When the phase spills, it will return to phase 1 cyclically.

For example, the optimal position (may not be unique) of the goal state is supposed to be $\hat{p}^*(7) = 12$. According to (3.17) and (3.18), we can get the plan in Fig. 3.14. The dash arrows between the optimal positions represent the backward search based on the rule. After all the optimal positions are found, the optimal policy and the green phase of traffic states will be planned regularly.

With outgoing b-connectors (e.g., VPS mode with outgoing 4-connectors and APS with outgoing 12-connectors), the basic principle in FPS can extend to a normal way. Thus, (3.13) can be developed as

$$\begin{cases} \hat{\boldsymbol{p}}(\tau+1) = (b\hat{\boldsymbol{p}}(\tau) - (b-1), b\hat{\boldsymbol{p}}(\tau) - (b-2), \ldots, b\hat{\boldsymbol{p}}(\tau)) \\ \hat{\boldsymbol{p}}(0) = 1 \\ \tau \in [0, T_p - 1], \tau \in \mathbb{N}. \end{cases} \tag{3.20}$$

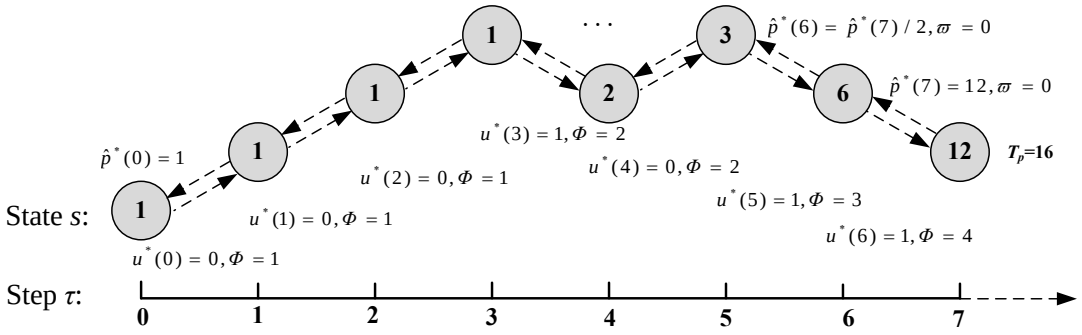


Figure 3.14: Example of fixed phase sequence based on optimal position (assume initial $\Phi = 1$ and $\varpi = \hat{p}^*(\tau) \mathrm{mod}_2$)

Once the optimal policy is found, the position of previous state $\hat{p}^*(\tau - 1)$ can be calculated inversely. Thus, (3.17) can be normalized as

$$\hat{p}^*(\tau-1) = \begin{cases} \hat{p}^*(\tau)/b, & \text{if } \hat{p}^*(\tau)\mathrm{mod}_b = 0 \\ (\hat{p}^*(\tau)+(b-1))/b, & \text{if } \hat{p}^*(\tau)\mathrm{mod}_b = 1 \\ \ldots, & \ldots \\ (\hat{p}^*(\tau)+1)/b, & \text{if } \hat{p}^*(\tau)\mathrm{mod}_b = b-1. \end{cases} \tag{3.21}$$

After we obtain all the optimal position during planning horizon T_p, with the initial phase setting $\Phi(\hat{p}(0))$, the optimal phase in (3.19) can extend as follows:

$$\Phi(\hat{p}^*(\tau+1)) = \begin{cases} \Phi(\hat{p}^*(\tau)), & \text{if } \hat{p}^*(\tau+1) = bp^*(\tau) - (b-1) \\ \Phi(\hat{p}^*(\tau)) + 1, & \text{if } \hat{p}^*(\tau+1) = bp^*(\tau) - (b-2) \\ \ldots & \ldots \\ \Phi(\hat{p}^*(\tau)) + b - 1, & \text{if } \hat{p}^*(\tau+1) = bp^*(\tau). \end{cases} \tag{3.22}$$

When the phase spills, it will return to the corresponding phase calculated according to

$$\Phi'(\hat{p}^*(\tau+1)) = \begin{cases} \Phi(\hat{p}^*(\tau+1)), & \text{if } \Phi(\hat{p}^*(\tau+1)) \le \Phi_m \\ \Phi(\hat{p}^*(\tau+1))\mathrm{mod}_{\Phi_m}, & \text{if } \Phi(\hat{p}^*(\tau+1)) > \Phi_m \end{cases} \tag{3.23}$$

where Φ_m is the maximum phase in each mode.

Rolling horizon approach

DP assures a global optimum solution, and it requires complete knowledge of arrivals over entire control period. In real-time traffic control problem, a rolling planning horizon

approach is commonly used with the new available information from detectors. The signal planning horizon to make the optimal policy consists of two parts: the 'head' and the 'tail'. Every interval in the 'head' of the horizon can receive the real-time available data from detectors, whilst the intervals in the 'tail' of the horizon are supplied with the predicted data obtained by a variety of approaches. Fixed-tail comes very close to the optimal and represents a feasible and very promising approach to real-time control [19]. In our model, the fixed-tail will be used. Accordingly, the 'tail' consists of a fixed flow related to the moving average of the flow rate.

It is noteworthy that the optimal policy is calculated for the entire horizon, but only the head period of the newly generated plan is implemented due to the actual data. When the head period expires and new arrival information becomes available, the process rolls forward and repeats itself as shown in Fig. 3.15. The process optimization to find the optimal or near optimal policy which is implemented for the next time horizon should be done in real-time, at least before new information becomes available. Actually, rolling period t_r can be set to various values, just satisfying less than the 'head' period.

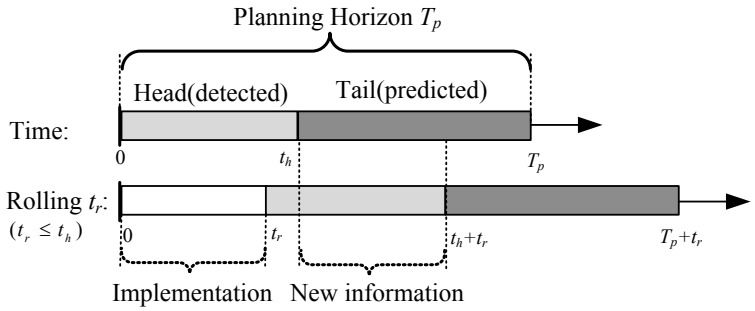


Figure 3.15: Rolling horizon approach

In the thesis, the traffic model is about the time-varying function of the state transfers which are executed by taking the optimal policy in planning horizon. During the horizon, we make a plan by using the detected data of the future vehicle arrivals and predicted data obtained by the estimation based on the arrival rates. The rolling horizon approach makes the controller to implement the strategies of the 'head' (or less then 'head') intervals of the horizon. As long as the running time of the optimization is less than the period of rolling forward, new arrival information can be planned during the next rolling horizon

successively. Hence, traffic states will be transferred continuously by taking the optimal policy. In the simulation section, we will show that the most time consumed is much less than the implemented intervals of the 'head' period in practice. The real-time traffic control could be guaranteed.

In summary, the procedure of our approach with phase sequence control mode is shown in Algorithm 2.

Algorithm 2 FSDP with phase sequence mode (b=2, 4, 12) signal control algorithm

choose an initial state s_0, set decision step $\tau = 0$, planning time $t = 0$;
while $t \leq T_p$ **do**
    compare the cumulative evaluation values $J(s)$ for all available states $s \in \mathcal{S}$; choose the optimal value $J(s^*)$ and its corresponding position $\hat{p}^*(\tau)$ of step τ;
    search $\hat{\boldsymbol{p}}(\tau + 1)$ by using (3.20) under $\hat{p}^*(\tau)$;
    $\tau = \tau + 1$;
    for all $\hat{p}_i(\tau) \in \hat{\boldsymbol{p}}(\tau)$ **do**
        if mod($\hat{p}_i(\tau)$,b)=1; **then**
            $t = t + 1$ ▷ case of signal unchanged, one more green extention
            calculate the transfered state and its value by solving (3.15) under satisfying (3.14);
        else
            $t = t + 4$ ▷ case of signal switched, 4 more mandatory intervals
            calculate the transfered state and its value by solving (3.16) under satisfying (3.14);
        end if
    end for
    add the new transfered states into the states set $\mathcal{S}$;
    search and compare the values of same states, then record the position of the optimal one, whereas other positions to be null;
    set the position of previous branched state to be null until the goal state ($t = T_p$) appears;
    set states with positions being null to be unavailable;
end while
according to (3.21), (3.22) and (3.23), the optimal strategy of each step is achieved.

In the next section, we discuss the application of the FSDP algorithm in numerical experiments.

3.3.3 Case study and analysis

The proposed algorithm is implemented to the cases of kinds of phase control modes at a typical isolated intersection illustrated in Fig. 2.1. Here, we use the traffic flow ratio $\hat{\rho}$ instead of relative traffic load ρ in (3.11). $\hat{\rho}$ is more precise than ρ to represent the traffic load, which is defined as follows based on the conventional 4-phase combinations.

$$\hat{\rho}=\frac{V}{S_a}=\frac{1800\sum_{c}^{4}\sum_{t}^{T}\max_{n\in G_c}\{k_t(n)\}}{T\cdot S_a} \tag{3.24}$$

where T is the entire simulation horizon; V is the sum of maximum volume of each traffic flow combination; S_a is the saturation flow per lane, i.e., 1800 veh/h; $n=1,2,\ldots,8$.

The performance measures are the average traffic delay at whole intersection and the vehicle queue length in each traffic flow combination. The algorithm efficiency is also important. In simulation, we compare the proposed algorithm (FSDP) and its rolling version (one step ahead FSDP-R) with the optimal fixed-time control method and adaptive control method Q-learning. All the experiments are implemented in MATLAB 64-bits with 3.8 GiB, Intel® Core i5 CPU 750, 2.67GHz $\times$ 4.

The traffic scenarios with symmetric and asymmetric arrival flows are simulated, as shown in Tab. 3.4. The entire simulation time is $T=1600$ intervals.

Table 3.4: Traffic scenarios of asymmetric and symmetric average flow rates

	Flow rate (veh/int)	Traffic volume by ITM (veh/h)	Ratio $\hat{\rho}$
Traffic Scenario A	(0.10,0.20,0.10,0.20)	(356,732,363,735)	0.6262
Traffic Scenario B	(0.20,0.20,0.20,0.20)	(746,720,739,721)	0.8231

The results of average traffic delay are shown in Tab. 3.5. It is clearly shown that as a whole our proposed algorithm FSDP (FSDP-R) has a good performance to reduce the traffic delays by comparing with Optimal FC and Q-learning methods. Notably, the phase sequence modes play an important role in these delay reductions. In Traffic Scenario B, using Q-learning the results of APS mode have about 55% delay reductions from FPS and VPS, and using FSDP the results of APS mode have about 68% and 72% improvements from FPS and VPS, respectively. Thus, on the other hand, it indicates that APS mode has a great potential to adaptive traffic signal control. Meanwhile, VPS mode is better than FPS mode. However, in practical, with rolling horizon approach (one step rolling ahead in our case), FSDP will take much computation time and be hard to operate in real-time.

Table 3.5: Results of average traffic delay (s) in Scenario A and B

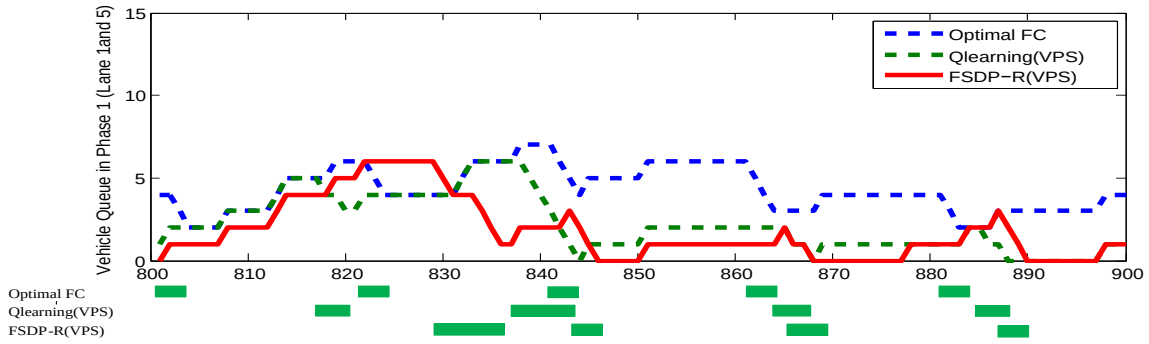
Traffic Scenario	A			B		
Phase sequence	FPS	VPS	APS	FPS	VPS	APS
Optimal FC [1]	24.63			50.04		
Q-learning	20.08	18.43	12.30	49.35	48.88	22.35
FSDP	18.55	14.65	8.78	45.38	40.48	12.65
FSDP-R [2]	14.16	12.20	-	37.88	36.64	-

[1] In FC, only operate in FPS.
[2] In FSDP-R with APS, results are not obtained as computation complexity.

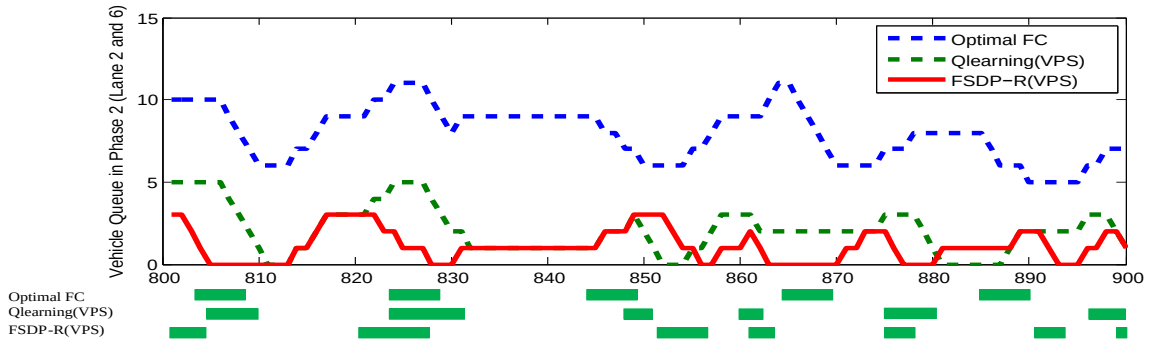
Table 3.6: Comparisons of run time (s)

Traffic Scenario	A			B		
Phase sequence	FPS	VPS	APS	FPS	VPS	APS
Optimal FC	0.1			0.1		
Q-learning	197.1	368.2	824.3	198.4	370.6	830.2
FSDP	4.4	16.1	369.2	10.5	74.4	1010.7
FSDP-R	55.1	243.6	-	125.6	1139.5	-

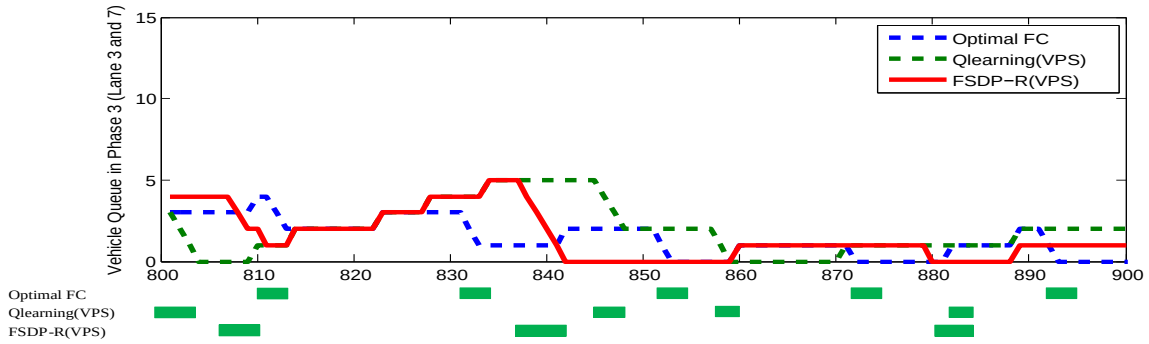
The simulation results of vehicle queues (the maximum queue length in each combination) and the duration of green phase are shown in Fig. 3.16, where the traffic demand in Scenario A is chosen. For comparison, FSDP-R(VPS) performers the best than optimal FC and Q-learning. FSDP-R(VPS) and Q-learning(VPS) can adjust the green duration and sequences intelligently according to the changing traffic arrivals, especially in FSDP-R(VPS) method. In the phase 1 and phase 3 with flow rate 0.10 veh/int, the total green phase duration of FSDP-R(VPS) and Q-learning(VPS) are both less than optimal FC, the less appearances as well. On the other hand, in phase 2 and phase 4 with high flow rates 0.20 veh/int, the total green phase duration of FSDP-R(VPS) and Q-learning(VPS) are both more than optimal FC and with more appearances.



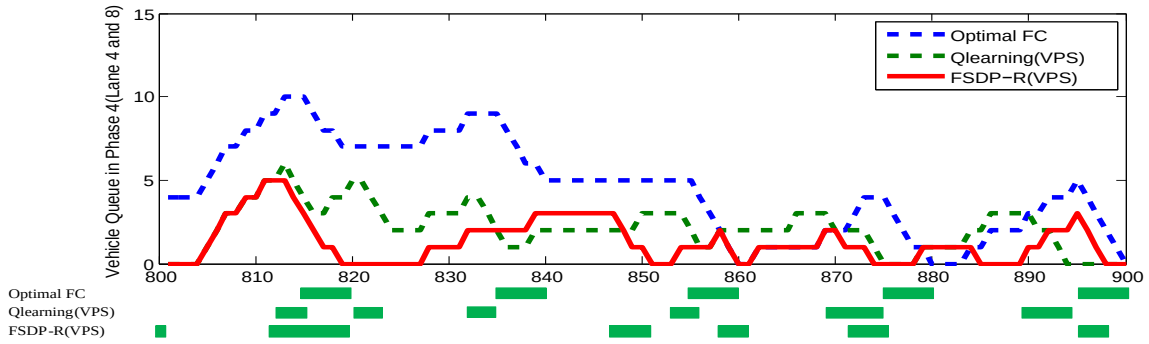
(a) The evolution of queue (Y-axis) in Phase 1 (Lane 1 and 5) between time step 800 and 900 (X-axis)



(b) The evolution of queue (Y-axis) in Phase 2 (Lane 2 and 6) between time step 800 and 900 (X-axis)



(c) The evolution of queue (Y-axis) in Phase 3 (Lane 3 and 7) between time step 800 and 900 (X-axis)



(d) The evolution of queue (Y-axis) in Phase 4 (Lane 4 and 8) between time step 800 and 900 (X-axis)

Figure 3.16: Comparisons of queue evolutions and signal sequences among the Optimal FC, Q-learning(VPS), and FSDP-R(VPS)

Details about the computation run time in seconds by different methods are given in Tab. 3.6. Obviously, the run time in FSDP with APS is taken much more than it with FPS and VPS. It could be explained by that the APS has 12-connectors (phase possibilities) so that the state space in decision tree possesses huge representations. The chance of same states get decreased as signal states increased. In this condition, the algorithm can not omit the branches largely so that it occupies much memory and time in computation. It ensures that in FSDP-R with APS, the computation is impractical for the traffic signal control although the traffic delay may be more reduced. However, from the comparisons of performance measures and time cost (e.g., $74.4/(1600 \times 2) = 0.02$ s/step), FSDP is after all a good choice in real-time adaptive signal control with conventional 4-phase modes of FPS and VPS.

3.4 Limitation of DP based algorithms

Despite the simple form exhibited by Bellman's equation and the global optimality it guarantees, DP is often of little practical value. A problem formulated in DP usually has the difficulty of computational requirement for finding optimal solution. Powell in his study [87] refers to the computation problem as "three curses of dimensionality", which is related to the dimensions of state space, information space (space of random noise), and decision space. Bellman's equation uses the state variable, information variable, and decision variable to calculate the value function and make optimal decision. The computation complexity is in exponential order to the size of each variable space. For example, there is an isolated intersection with 8 lanes in 4-phase mechanism. Each lane can occupy 19 at most vehicles, the arrival information is either 0 or 1 for each lane and the signal decision for each lane is either 0 remaining green or 1 switching the current status. Thus, the computation order is $4 \times 20^8 \times 2^8 \times 2^8$. The memory requirement for a look-up table approximation of value function J is "tremendously high" and impossible for a controller to find optimal solutions in real-time.

In addition, another difficulty of DP is that a complete set of information for the whole problem is required. For operations in real-time, we usually do not have complete information a priori. This makes DP problem in a short term planning, which effects the performance in whole horizon. Research has been found that some approaches, such as

rolling horizon approach, model predictive control, can weaken this effects. But, it will cost additional time especially in DP problems.

To verify the DP issues above in the application of traffic signal control, we proposed two DP based algorithms for practical investigation and understood the difficulties of DP in depth. Also, some valuable information was discovered.

In our early research, an optimal policy can be found using DP algorithms such as value iteration and policy iteration. Both of them are backward search algorithms to obtain the global optimum policy whereas having some disadvantages of evaluating the complete state space. In MDP, the transition probability $p(s_t, a_t, s_{t+1})$ is the probability of state s_t transforming to next state s_{t+1} by taking action a_t, and the reward function $R_t(s_t, a_t, s_{t+1})$ is the immediate effect of action a_t. By using traditional DP algorithms to solve Bellman's equation, the knowledge of transition probability and reward function are required. It is very hard to obtain the possibility matrix in the whole state space. Moreover, state values are stored in a look-up table, where the dimension of the table is equal to the dimension of the state space. We have to loop over the entire state-action space to evaluate the optimal decision so that the optimal stationary deterministic policy can be obtained in the convergence of iteration. In our case study, the value iterative algorithm works for a simple traffic signal control problem. Although the controller appears satisfied performance, this algorithm is offline with impractical results due to the drawbacks of DP mentioned above.

In order to bring principles of DP to real-time control, there are two immediate tasks: first, reducing the dimensionality of control problem; second, accumulating limited sensor information progressively to improve knowledge of underlying control process.

Therefore, in the second type algorithm, we try to investigate a forward DP using the search tree. The planning problem can be formulated by the shortest path problem from the predefined original state to the goal state (or termination of time horizon). By introducing Monte-Carlo method, the state transition can be completed in a deterministic way. The problem can still be a stochastic one when it processes the independent extraneous random information (noise). Meanwhile, rolling horizon approach is applied to the algorithm in order to guarantee the most use of limited receiving information. In FSDP algorithm, although we use the accessible states represented in a graph without evaluating all the states in traditional DP, limitations of the planning horizon time (T_p) and the increased action space (b-connectors) still exist. These limitations largely reduce the com-

putation efficiency, which impairs the capability for practical operations, such as the case of FSDP with APS mode for adaptive traffic signal control. In detail, assuming the tree is completed, when we have $T_p = 16$ with APS mode that $b = 12$, the last stage has 12^{16} states. In the proposed algorithm with time constraints and same states brunches omitted, this huge state space will not appear, but still too large to compute.

Fortunately, a great number of studies are investigating learning systems based on DP for solving stochastic optimal control problems, arguing that DP provides the appropriate basis for compiling planning results into reactive strategies for real-time control, as well as for learning such strategies when applied to the tasks involving uncertainty. Actually, heuristic function such as in A* is usually hard to find. Therefore, learning technique to estimate this heuristic information should be reasonably noticed. Approximate dynamic programming (ADP) based on this becomes a good candidate to address the limitation of DP. We will introduce ADP approach in the next chapter and propose an algorithm using ADP with the machine learning technique, i.e., recursive least-squares temporal difference learning (RLS-TD(λ)), for real-time adaptive traffic signal control.

3.5 Summary

In this chapter, we discussed two proposed DP based algorithms for adaptive traffic signal control. Then, some limitations of DP were discovered and concluded in the perspective of methodology.

For a simple 2-phase intersection problem, the value iteration algorithm is implemented by using stochastic state transition model. The idea of decomposition of traffic states is proposed for the construction of probability transition matrix, which is usually hard to receive but is really done in our case. The optimal stationary policy is obtained by calculating the value iteratively for convergence. By the case study, this method performs better than traditional fixed-time and actuated control. Unfortunately, it is an offline operation and costs much time for convergence. Even in changing traffic conditions, the optimal stationary policy should be computed repeatedly.

Consequently, we developed a forward search DP algorithm based on A*. This algorithm tries to save computation time and is applied to the deterministic state transition. The application of FSDP is in a typical intersection problem. Importantly, the achievement of FSDP algorithm is about the successful application of real-time adaptive signal

control. Additionally, rolling horizon approach can be implemented into the algorithms for realistic application with limited traffic arrival information in real-time. But the phase optimization limits in FPS (fixed phase sequence) and VPS (variable phase sequence) modes, which are conventional 4-phase mechanisms. In extended work, this algorithm appears computation burden for the proposed concept of APS (adaptive phase sequence) mode due to the states largely increased. It is found that FSDP and its rolling version operate well on the performances of reductions of average traffic delay and vehicle queue length. As long as not with too small rolling steps ahead (such as one step), FSDP could guarantee the real-time operations in FPS and VPS.

As we can see that many improvements are achieved by using APS mode in FSDP. Whereas, DP has the limitations of computation in large state space and incomplete information in real-time. We try to use another potential approach, namely approximate dynamic programming (ADP) to address these difficulties. The ADP with a particular learning method that we prepare to use will be introduced in the next chapter, which is the final approach for real-time adaptive traffic signal control at isolated intersection and whole network.

CHAPTER 4

APPROXIMATE DYNAMIC PROGRAMMING WITH RLS-TD(λ) LEARNING ALGORITHM

4.1 Introduction

Approximate dynamic programming is a derivative of DP. The formulation of ADP is firstly proposed by Werbos [89] and developed by many researchers. In general, there are three distinctive features of ADP approach, regardless of the specific applications [19]. First, ADP aims to significantly reduce computational requirement by using approximation techniques estimating the true value function in Bellman's equation. Second, ADP adopts a forward process rather than the backward recursive calculation in conventional DP algorithm. In forward process, value function is calculated only upon visiting the actual state rather than the entire state space. The state transition only happens after exogenous information is observed and a decision is taken. Third, the adaptation of approximation using machine learning technique is employed in ADP. The parameters in adaptation are tuned incrementally and finally converge to optimal ones.

In principle, ADP should be able to approximate the solution of any problem in control or planning which can be formulated as an optimization problem [128]. For traffic signal

control problem formulated in dynamic modeling system, research has been found that ADP has a potential to solve the computational requirement which DP cannot tackle.

In this chapter, we try to make ADP practical for the real-time implementation of adaptive traffic signal control system. The system process is stochastic because it is influenced by exogenous and random vehicle arrivals. The state transition here is deterministic and the system receives real-time information of future arrivals before evaluating a decision. In the following context, we will introduce some fundamentals of ADP and machine learning. Online algorithms for the traffic problems will be designed. This chapter is organized as follows.

Firstly, we introduce approximation structures of ADP, which often adopts the estimated function of neural network or linear function. Secondly, the normal learning techniques, such as gradient descent, temporal difference (TD(λ)), for updating approximation are presented. Then, we will suggest to integrate recursive least-squares temporal difference learning (RLS-TD(λ)) and related scheme to ADP approach. Lastly, we propose algorithms for isolated intersection signal controller and traffic network control based on the previous model constructions.

4.2 Structure of ADP

In ADP, we define a continuous approximation function $\tilde{J}(\cdot,\theta)$: $S \times \mathbb{R}^K \rightarrow \mathbb{R}$ to replace the exact cost-to-go function $J(\cdot)$: $S \rightarrow \mathbb{R}$, where θ is a K-dimensional parameter vector of $\tilde{J}$, and S is state space. At each discrete temporal interval t, we can calculate a greedy decision $a_t^*(s_t)$ by using $\tilde{J}$. That is,

$$a_t^*(s_t) = \arg\min_{a_t \in A} E_{w_t}\left\{r_t + \gamma \tilde{J}(s_{t+1}, \theta_t)\right\}. \tag{4.1}$$

Meanwhile, the objective value function in (3.9) can be observed by

$$\hat{J}(s_t) = \min_{a_t \in A} E_{w_t}\left\{r_t + \gamma \tilde{J}(s_{t+1}, \theta_t)\right\}, \tag{4.2}$$

where $\hat{J}(s_t)$ is the observed value of the current state according to the environment.

To approximate the cost-to-go function, one usually tries to choose a parameter vector θ so as to minimize some error metric between the function $J(\cdot)$ and $\tilde{J}(\cdot,\theta)$. A common objective for updating approximation function is to find

$$\theta^* = \arg\min_{\theta \in \mathbb{R}^K} \left\| J - \tilde{J} \right\|, \tag{4.3}$$

by calculating correction increment $\Delta\theta$ and to update estimation through

$$\theta_{t+1} = \theta_t + \Delta\theta_t \tag{4.4}$$

where $\|\cdot\|$ is the Euclidean norm, and the correction increment $\Delta\theta$ is usually obtained from machine learning process.

Convergence of θ to θ^* by using an incremental process expressed by (4.4) is guaranteed if specific approximation structures and learning techniques are used [129].

As we can see from (4.2) and (4.4), instead of computing the optimal J which is huge-dimensional, we compute the low-dimensional parameter vector θ. By using machine learning technique, ADP makes the estimated value of being the current state to approach the true value function in Bellman's equation. Thus, it can avoid the tremendous computation in DP and just operates forward in time to calculate the estimated value of the visiting actual state. Next, we will discuss ADP approximation function and learning techniques in detail. Once appropriate approximation architecture is established, learning techniques can then be applied to update parameters of the approximation function progressively.

An approximation function is also called an approximator in this thesis. There are two common groups for continuous functions. One is the non-linear approximator, such as neural networks, and the other one is the linear approximator. Other forms of being not continuous are aggregation, look-up table, etc. Here, we mainly introduce the continuous approximators and their gradient descent rules for updating parameters.

4.2.1 Neural networks approximation

ADP architecture employed by neural networks has a rich literature. The early research about this refers to Neuro-Dynamic Programming (NDP)[89, 86] and its related schemes, such as Adaptive Critic Designs (ACDs) [130]. The extensive research in NDP can also be categorized as [91] (1) heuristic dynamic programming (HDP); (2) dual heuristic dynamic programming (DHP); and (3) globalized dual heuristic dynamic programming (GDHP). Variations from these three basic design paradigms are also available, such as action dependent (AD) versions of the above architectures. AD refers to the fact that the action value is an additional input to the critic network. Action dependent variants from the original three paradigms will be denoted with an abbreviation of "AD" in front of their specific architecture, such as ADHDP. Noticeably, the ADP with neural networks is usually analyzed in continuous-time problems.

Normally in NDP, there are three modules, namely the model module, action module, and critic module. Considering an online learning algorithm using ADHDP, the neural networks in ADP just refer to action network and critic network. The action network is used to generate the control value and the critic network is used to evaluate the efficiency of the control value generated by the action network. The outputs of both networks will be adjusted through tuning the weights in them in order to make the equation of the principle of optimality more balanced. The structure of ADHDP [131] can be described in Fig. 4.1.

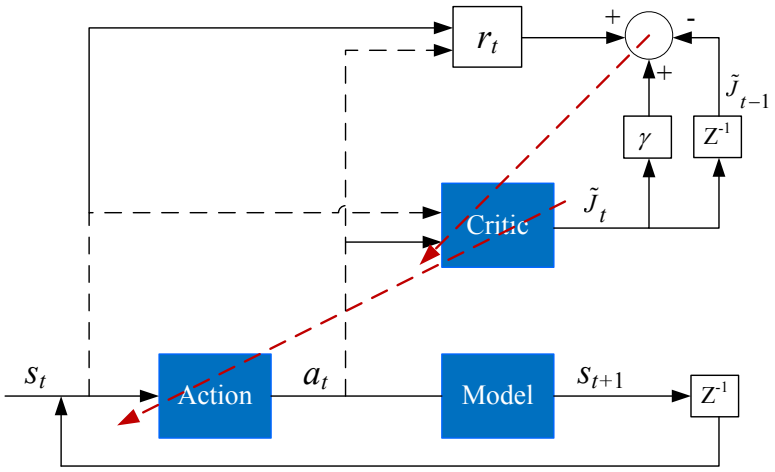


Figure 4.1: The schematic diagram of the ADHDP framework (the solid lines represent signal flow, while the dashed lines are the paths for parameter tuning)

The structure of the action network and the critic network is shown in Fig. 4.2. The basic idea in NDP (or ACDs) is to adapt the weights of critic network to make the approximation function satisfy the modified Bellman's equation in (4.2). The design of critic network and its training are very important. Control value generated from action network is adjusted through the tuning weights in action network, according to the feedback value of critic network. Based on [91], non-linear three layered (one hidden layer) feed-forward neural networks with hyperbolic tangent activation function

$$Th(y) = \frac{1 - \exp(-y)}{1 + \exp(-y)} \tag{4.5}$$

is considered. The structure and tuning weights of critic network and action network are presented as follows.

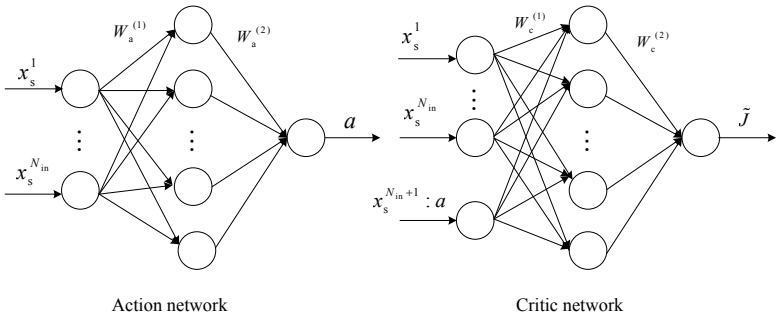


Figure 4.2: The structure of the action network and critic network

Critic network

In the critic network, the estimated value $\tilde{J}_t$ is

$$\tilde{J}_t = \sum_{i=1}^{N_{\text{ch}}} W_{\text{c}}^{(2)i}(t) g_{\text{c}}^i(t), \tag{4.6}$$

$$g_{\text{c}}^i(t) = Th(h_{\text{c}}^i(t)),\ i = 1, 2, ..., N_{\text{ch}}, \tag{4.7}$$

$$h_{\text{c}}^i(t) = \sum_{j=1}^{N_{\text{in}}+1} W_{\text{c}}^{(1)i,j}(t) x_{\text{s}}^j(t),\ i = 1, 2, ..., N_{\text{ch}}, \tag{4.8}$$

where h_{c}^i is the ith hidden node input the critic network; g_{c}^i is corresponding output of the hidden node; $W_{\text{c}}^{(1)i,j}$ is the generic input weight of the critic network to be learned and $W_{\text{c}}^{(2)i}$ is the generic output weight; N_{ch} is the number of neurons in the hidden layer and $N_{\text{in}} + 1$ is the number of inputs into the critic network including the action a_t from the action network; x_{s}^j is the jth state (or action when $j = N_{\text{in}} + 1$) input for critic network.

The critic network is used to provide $\tilde{J}_t$ as an approximation of J_t at time t. The prediction error is defined as

$$e_{\text{c}}(t) = r_t + \gamma \tilde{J}_t - \tilde{J}_{t-1} \tag{4.9}$$

and the critic network is trained by minimizing the objective function

$$E_{\text{c}}(t) = \frac{1}{2} e_{\text{c}}^2(t). \tag{4.10}$$

The weight update rule for the critic network using a gradient descent adaptation is given by

$$\mathbf{W}_{\text{c}}(t+1) = \mathbf{W}_{\text{c}}(t) + \Delta \mathbf{W}_{\text{c}}(t) \tag{4.11}$$

$$\Delta \mathbf{W}_{\mathrm{c}}(t) = l_{\mathrm{c}}(t)(-\frac{\partial E_{\mathrm{c}}(t)}{\partial \mathbf{W}_{\mathrm{c}}(t)}) \tag{4.12}$$

$$\frac{\partial E_{\mathrm{c}}(t)}{\partial \mathbf{W}_{\mathrm{c}}(t)} = \frac{\partial E_{\mathrm{c}}(t)}{\partial \tilde{J}_t} \frac{\partial \tilde{J}_t}{\partial \mathbf{W}_{\mathrm{c}}(t)} \tag{4.13}$$

where $l_{\mathrm{c}}(t) > 0$ is the learning rate of the critic network at time t, which usually decreases with time to a small value, and $\mathbf{W}_{\mathrm{c}}$ is the weight vector in critic network.

Action network

In the action network, the output control (action) value a_t can be derived from

$$a_t = Th(\nu(t)), \tag{4.14}$$

$$\nu(t) = \sum_{i=1}^{N_{\mathrm{ah}}} W_{\mathrm{a}}^{(2)i}(t) g_{\mathrm{a}}^{i}(t), \tag{4.15}$$

$$g_{\mathrm{a}}^{i}(t) = Th(h_{\mathrm{a}}^{i}(t)),\ i = 1, 2, ..., N_{\mathrm{ah}}, \tag{4.16}$$

$$h_{\mathrm{a}}^{i}(t) = \sum_{j=1}^{N_{\mathrm{in}}} W_{\mathrm{a}}^{(1)i,j}(t) x_{\mathrm{s}}^{j}(t),\ i = 1, 2, ..., N_{\mathrm{ah}}, \tag{4.17}$$

where h_{a}^{i} is the ith hidden node input the action network; g_{a}^{i} is corresponding output of the hidden node; $W_{\mathrm{a}}^{(1)i,j}$ is the generic input weight of the action network to be learned and $W_{\mathrm{a}}^{(2)i}$ is the generic output weight; N_{ah} is the number of neurons in the hidden layer and N_{in} is the number of inputs into the action network.

The principle in adapting the action network is to indirectly back propagate the error between the desired ultimate objective, denoted by U_t, and the approximate function $\tilde{J}_t$ from the critic network.

The weight updating in the action network can be formulated as follows. Let

$$e_{\mathrm{a}}(t) = \tilde{J}_t - U_t \tag{4.18}$$

and the action network is trained by minimizing the objective function

$$E_{\mathrm{a}}(t) = \frac{1}{2} e_{\mathrm{a}}^{2}(t). \tag{4.19}$$

The weight update rule for the action network is given by using a gradient descent adaptation

$$\mathbf{W}_{\mathrm{a}}(t+1) = \mathbf{W}_{\mathrm{a}}(t) + \Delta\mathbf{W}_{\mathrm{a}}(t) \tag{4.20}$$

$$\Delta\mathbf{W}_{\mathrm{a}}(t) = l_{\mathrm{a}}(t)(-\frac{\partial E_{\mathrm{a}}(t)}{\partial \mathbf{W}_{\mathrm{a}}(t)}) \tag{4.21}$$

$$\frac{\partial E_{\mathrm{a}}(t)}{\partial \mathbf{W}_{\mathrm{a}}(t)} = \frac{\partial E_{\mathrm{a}}(t)}{\partial \tilde{J}_t}\frac{\partial \tilde{J}_t}{\partial a_t}\frac{\partial a_t}{\partial \mathbf{W}_{\mathrm{a}}(t)} \tag{4.22}$$

where $l_{\mathrm{a}}(t) > 0$ is the learning rate of the action network at time t, which usually decreases with time to a small value, and $\mathbf{W}_{\mathrm{a}}$ is the weight vector in action network.

From the structure of neural networks in ADP and the corresponding gradient descent rules for adaptation, this ADP proposes a feasible and effective solution for reinforcement learning problem with continuous states and actions, avoiding the " curse of dimensionality" problem in DP. However, there are still some problems as follows [131]. Firstly, the poor choice of initial values of network weights may lead to poor effects. Sometimes we need do some offline training by simulation for initial weights setting. Secondly, limitation is the desired ultimate objective U setting. Usually, the reward is set to 0 for encouragement and -1 for punishment, and the total return is zero if the action is an optimal one. But in complex cases, a continuous reward $\tilde{J}$ (not be 0) would be a better choice. Thus, the large discrepancy on U might lead to a large training error in action network. In a word, despite computation advantages in ADP, the neural network approximation appears some difficulties in network training, especially for a complex and discrete time problem. Next, we will introduce another approximation architecture, namely the linear function approximation which is easy to train and implement.

4.2.2 linear function approximation

The literatures on ADP with linear approximator are relatively rich and well proved, because the linear function is simpler for both implementation and training than the neural network approximator mentioned above. More importantly, the linear case has a broad application for practical discrete-time problems with discrete space. In supervised learning, such as the neural network training, targets are the corresponding exact outputs. While in unsupervised learning, we do not have the source of exact outputs. Reinforcement learn-

ing technique allows a system to learn from its own interactions with the process, which belongs to unsupervised learning and is well applied to linear approximator application.

A linear approximator can be expressed as

$$\tilde{J}(s,\theta) = \sum_{i=1}^{K} \theta(i)\phi^i(s) \tag{4.23}$$

where $\theta(i)$ is the parameter and ϕ^i is a mapping function defined on state $s \in S$. The function ϕ^i refers to the feature-extraction function (or basis function) that maps state to valued feature vector. Each parameter $\theta(i)$ refers to the associated weight.

Using vector expression, (4.23) can also be written as

$$\tilde{J}(s,\theta) = \theta^{\mathrm{T}}\phi(s) \tag{4.24}$$

or

$$\tilde{J}(\theta) = \Phi\theta \tag{4.25}$$

where $\theta = (\theta(1), \theta(2), \ldots, \theta(K))^{\mathrm{T}}$ is a column parameter vector, and $\phi(s) = (\phi^1(s), \phi^2(s), \ldots, \phi^K(s))^{\mathrm{T}}$ is the column vector of feature functions, and Φ is viewed as an $|S| \times K$ matrix whose ith column is equal to ϕ^i.

Based on feature-extraction function, the architecture of linear function approximation is illustrated in Fig. 4.3.

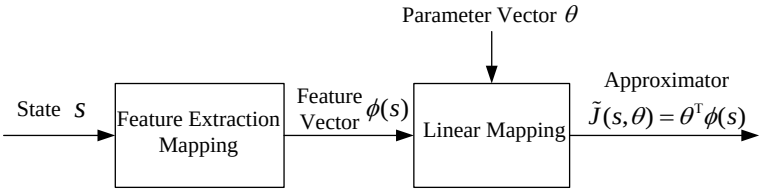


Figure 4.3: Linear feature-based architecture

To analysis the adaptation of parameter vector θ, as usual, the state value function J^π is estimated from experience generated using policy π. It means that we use the action determined by policy π to choose the next state that we visit. This is known as on-policy learning. The true value of policy J^π is defined as

$$J^\pi(s) = E\left\{\sum_{t=0}^{\infty} \gamma^t r_t(s, A^\pi(s))\right\} \tag{4.26}$$

where $A^{\pi}(s)$ is the fixed policy on state s.

It turns out that if we use on-policy learning, the linear approximator $\tilde{J}$ with parameter vector θ is used to approximate true value function J^{π} under the selected policy π. The function approximation should be fitted to minimize the function

$$\min_{\theta} \sum_{s} d_s^{\pi} (J^{\pi} - \tilde{J})^2 \tag{4.27}$$

where d_s^{π} is the steady-state probability of being in state s while following policy π.

The θ can be optimized by using method in supervised learning. However, the function approximator has limited state resources and limited solutions. To address this, we use the observed output $\hat{J}_t$ in (4.2) under the greedy policy solved in (4.1), and we can approximate it using $\tilde{J}$ with adaptation parameter learned in unsupervised learning (reinforcement learning). Thus, in order to find the best value of θ, we can do by

$$\min_{\theta} \frac{1}{2} E(\hat{J} - \tilde{J})^2 \tag{4.28}$$

where constant $\frac{1}{2}$ is just to be added when doing the derivative of quadratic function.

The expected value refers to a stochastic distribution, which can be solved on observed examples. An ideal goal in (4.28) would be to find a global optimum θ^*. Reaching this is sometimes possible for simple function approximators such as linear ones rather than the complex ones such as neural networks. In addition, in linear case there is only one optimum θ^*. Thus, any method guaranteed to converge to or near a local optimum is automatically guaranteed to converge to or near the global optimum [58]. It is natural to apply a gradient descent rule for the adaptation step,

$$\begin{aligned} \theta_{t+1} &= \theta_t + \Delta\theta_t \\ &= \theta_t - \eta_t(\tfrac{1}{2}\nabla_{\theta_t}(\hat{J}(s_t) - \tilde{J}(s_t, \theta_t))^2) \\ &= \theta_t + \eta_t(\hat{J}(s_t) - \tilde{J}(s_t, \theta_t))\nabla_{\theta_t}\tilde{J}(s_t, \theta_t) \end{aligned} \tag{4.29}$$

where $\eta_t > 0$ is the step-size learning parameter that satisfies the following conditions for convergence.

$$\sum_{t=0}^{\infty} \eta_t = \infty \; ; \; \sum_{t=0}^{\infty} {\eta_t}^2 < \infty. \tag{4.30}$$

Since $\tilde{J}(s,\theta) = \sum_{i=1}^{K} \theta(i)\phi^i(s) = \theta^{\mathrm{T}}\phi(s)$, the gradient with respect to θ_t is given by

$$\nabla_{\theta_t}\tilde{J}(s_t,\theta_t) = \begin{pmatrix} \frac{\partial \tilde{J}(s_t,\theta_t)}{\theta_t(1)} \\ \frac{\partial \tilde{J}(s_t,\theta_t)}{\theta_t(2)} \\ \vdots \\ \frac{\partial \tilde{J}(s_t,\theta_t)}{\theta_t(K)} \end{pmatrix} = \begin{pmatrix} \phi^1(s_t) \\ \phi^2(s_t) \\ \vdots \\ \phi^K(s_t) \end{pmatrix} = \phi(s_t). \tag{4.31}$$

Thus, the updating equation (4.29) is given by

$$\theta_{t+1} = \theta_t + \eta_t(\hat{J}(s_t) - \tilde{J}(s_t,\theta_t))\phi(s_t). \tag{4.32}$$

As the linear approximator is easy to be completed in complex problem, we will focus on this linear form and search some superior learning methods instead of the gradient descent one. In the next section, we will introduce a popular learning technique for updating parameter vector in linear approximator. The learning method refers to temporal difference learning and its developed version combining with recursive least-squares method.

4.3 RLS-TD(λ) for linear function approximation

4.3.1 Multi-step temporal difference (TD(λ)) learning

Suppose that we observe a sequence of states s_t based on simulation implementation with random information w_t, i.e., $(s_0, a_0, w_1, s_1, a_1, w_2, \ldots, s_T, a_T, w_{T+1})$. The temporal difference δ_t (also called TD error) is defined corresponding to the transition from s_t to s_{t+1} by

$$\delta_t = r_t + \gamma\tilde{J}(s_{t+1},\theta_t) - \tilde{J}(s_t,\theta_t). \tag{4.33}$$

In linear case $\tilde{J}(s_t,\theta_t) = \theta_t^{\mathrm{T}}\phi(s_t)$, we simply use $\phi_t'\theta_t$ ('$\prime$' means the transpose) to substitute the approximation $\tilde{J}$ in (4.33). Thus, the TD error at time t can be rewritten as

$$\delta_t = r_t - (\phi_t - \gamma\phi_{t+1})'\theta_t. \tag{4.34}$$

Then, for $t = 0, 1, \ldots, T$, the multi-step TD (also called TD(λ)) learning method updates θ_t according to the formula

$$\theta_{t+1} = \theta_t + \eta_t\delta_t \sum_{k=0}^{t} (\gamma\lambda)^{t-k}\nabla_{\theta_t}\tilde{J}(s_t,\theta_t) \tag{4.35}$$

where θ_0 is initialized to some arbitrary vector, η_t is a sequence of scalar step-size satisfying (4.30), λ is a parameter in [0,1] , and the gradient $\nabla_{\theta_t}\tilde{J}(s_t,\theta_t)$ is the vector of partial derivatives with respect to the components of θ_t.

In the case of linear function approximation, a more convenient representation of TD(λ) is obtained by defining a sequence of eligibility vectors z_t,

$$\begin{aligned} z_t &= \sum_{k=0}^{t} (\gamma\lambda)^{t-k} \nabla_{\theta_t}\tilde{J}(s_t,\theta_t) \\ &= \sum_{k=0}^{t} (\gamma\lambda)^{t-k} \phi_t. \end{aligned} \tag{4.36}$$

With this notation, (4.35) can be rewritten as

$$\theta_{t+1} = \theta_t + \eta_t \delta_t z_t \tag{4.37}$$

and the eligibility vectors can be updated recursively according to

$$z_{t+1} = \gamma\lambda z_t + \phi_{t+1} \tag{4.38}$$

initialized with $z_t = \mathbf{0}$.

In the study of Tsitsiklis & Van Roy [98], the above linear TD(λ) algorithm is proved to converge with probability 1 under certain assumptions and the limit of convergence θ^* is also derived, which is satisfied the following equation,

$$E_0[A(X_t)]\theta^* - E_0[b(X_t)] = 0 \tag{4.39}$$

where $X_t = (s_t, s_{t+1}, z_t)$ $(t = 0, 1, \dots)$, is a Markov process, $E_0[\cdot]$ is the expectation with respect to the unique invariant distribution of X_t, or called "steady-state" expectation, and $A(X_t)$ and $b(X_t)$ are matrix and vector valued functions, respectively, which are defined as

$$A(X_t) = z_t(\phi_t - \gamma\phi_{t+1})' \tag{4.40}$$

$$b(X_t) = z_t r_t. \tag{4.41}$$

Especially for TD(0),

$$A(X_t) = \phi_t(\phi_t - \gamma\phi_{t+1})' \tag{4.42}$$

$$b(X_t) = \phi_t r_t. \tag{4.43}$$

Using $A = E_0[A(X_t)]$ and $b = E_0[b(X_t)]$, according to (4.39), θ^* can be solved by

$$\theta^* = A^{-1}b \tag{4.44}$$

where A is invertible.

For special case $\lambda = 0$, TD(0) is an equivalent to single-step TD algorithm where only the most recent observation matters to calculate the value function as well as the update of approximation. The details of proof of TD(λ) convergence is excellently given in [98]. We omit it to avoid repetition.

In order to improve the efficiency of linear TD(λ) algorithm, recursive least-squares method in the next section is used for the linear TD(λ) learning.

4.3.2 RLS-TD(λ)

One of the most appealing features of linear regression is the ease with which models can be updated recursively. Using recursive least-squares method and TD(λ), the linear function approximation could be updated recursively. Bradtke and Barto in [99] firstly proposed Least-Squares TD (LS-TD) and its recursive version RLS-TD in the linear regression. Then, in the work of Boyan and Xu [100, 132], they proposed that the LS-TD(λ) and RLS-TD(λ) can be viewed as the extension of LS-TD and RLS-TD from λ=0 to general $0 \leq \lambda \leq 1$, respectively. As mentioned in [100], LS-TD(λ) offers several significant advantages. At first, least-square algorithms would be expected to converge with fewer training samples. Secondly, TD(λ)'s convergence can be slowed dramatically by a poor choice of the step-size parameters but LS-TD(λ) eliminates these parameters. Thirdly, performance of TD(λ) is sensitive to the initial estimate value while LS-TD(λ) does not rely on an arbitrary initial estimate. The RLS-TD(λ) has the similar advantages. Moreover, RLS-TD(λ) has an advantage in computation and is more suitable for online learning than LS-TD(λ).

Before this, we introduce firstly the least-squares TD (LS-TD(0)) and recursive least-squares TD (RLS-TD(0)) learning suggested in [99].

In linear function approximation problem, let us study the simple linear regression form,

$$\psi_t = \theta^{\mathrm{T}}\phi_t + \upsilon_t \tag{4.45}$$

where ψ_t and ϕ_t are measured quantities and θ is to be determined. The variable υ_t is the equation error and it is natural to select θ so that the variance of υ_t is minimized, i.e., to

find

$$\min_{\theta} O(\theta) \tag{4.46}$$

where

$$\begin{aligned} O(\theta) &= \frac{1}{2} E v_t^2 \\ &= \frac{1}{2} E(\psi_t - \phi_t{}'\theta)^2. \end{aligned} \tag{4.47}$$

The function $O(\theta)$ is quadratic in θ, therefore (4.46) can be found by solving

$$\left[-\frac{d}{d\theta} O(\theta) \right]^{\mathrm{T}} = E\phi_t(\psi_t - \phi_t{}'\theta) = 0. \tag{4.48}$$

Eq. (4.48) cannot be solved exactly, since the probability distribution of (ψ_t, ϕ_t) is not known and the expectation cannot be evaluated. One way around this would be to replace expectation with sample means, i.e., Ev_t^2 could be approximated by $(1/t)\sum_{i=1}^{t} v_i^2$, and it brings us the least-squares method. Thus, we prefer to write the criterion function as

$$O(\theta) = \frac{1}{t} \sum_{i=1}^{t} (\psi_i - \phi_i{}'\theta)^2. \tag{4.49}$$

In unsupervised learning, there is not output source ψ_t. The temporal difference in (4.34) can be applied to sample the errors. In LS-TD(0), the least-squares approximation to θ^* at time t ($t \geq 1$) is the vector θ_t that minimizes the quadratic objective function

$$O(\theta_t) = \frac{1}{t} \sum_{i=1}^{t} \left(r_i - (\phi_i - \gamma\phi_{i+1})'\theta_t \right)^2. \tag{4.50}$$

By employing the instrumental variables approach [133], the LS-TD(0) solution of (4.50) give us the tth estimate to θ^*. That is,

$$\begin{aligned} \theta_t &= \left(\frac{1}{t} \sum_{i=1}^{t} \phi_i(\phi_i - \gamma\phi_{i+1})' \right)^{-1} \left(\frac{1}{t} \sum_{i=1}^{t} \phi_i r_i \right) \\ &= \left(\sum_{i=1}^{t} \phi_i(\phi_i - \gamma\phi_{i+1})' \right)^{-1} \left(\sum_{i=1}^{t} \phi_i r_i \right) \\ &= \left(\sum_{i=1}^{t} A(X_t) \right)^{-1} \left(\sum_{i=1}^{t} b(X_t) \right) \end{aligned} \tag{4.51}$$

where ϕ_i is the instrumental variable chosen to be uncorrelated with the input and output noise.

Using LS-TD(0) builds explicit estimates of the A matrix and b vector expressed by expectation in (4.44). We use the estimations $\tilde{A}$ and $\tilde{b}$, which are expressed as follows:

$$\tilde{A}_t = \sum_{i=1}^{t} A(X_i) = \sum_{i=1}^{t} \phi_i(\phi_i - \gamma\phi_{i+1})' \tag{4.52}$$

$$\tilde{b}_t = \sum_{i=1}^{t} b(X_i) = \sum_{i=1}^{t} \phi_i r_i. \tag{4.53}$$

By inserting the expressions of (4.52) and (4.53) into (4.51) we obtain

$$\theta_t = \tilde{A}_t^{-1}\tilde{b}_t. \tag{4.54}$$

After κ independent trajectories have been observed, $\tilde{A}_t$ is an unbiased estimate of κA and $\tilde{b}_t$ is an unbiased estimate of κb. Thus, θ^* can be estimated as $\tilde{A}_t^{-1}\tilde{b}_t$.

LS-TD(0) method requires the computation of a matrix inverse at each time step. Thus, recursive least-squares technique is used to derive a modified algorithm, namely RLS-TD(0), to decrease the computational complexity of LS-TD(0). The weight update rules of RLS-TD(0) are as follows

$$\tilde{\delta}_t = r_t - (\phi_t - \gamma\phi_{t+1})'\theta_{t-1} \tag{4.55}$$

$$P_t = P_{t-1} - \frac{P_{t-1}\phi_t(\phi_t - \gamma\phi_{t+1})'P_{t-1}}{1 + (\phi_t - \gamma\phi_{t+1})'P_{t-1}\phi_t} \tag{4.56}$$

$$\theta_t = \theta_{t-1} + \frac{P_{t-1}}{1 + (\phi_t - \gamma\phi_{t+1})'P_{t-1}\phi_t}\tilde{\delta}_t\phi_t. \tag{4.57}$$

The detailed derivation of RLS-TD(0) can be found in Appendix A1 when $\lambda = 0$ and $z_t = \phi_t$.

Notice that (4.57) looks like the TD(0) learning rule for function approximations that are linear in the parameters, except that the scalar step-size parameter is replaced by a gain matrix. To use the RLS-TD(0) method, users must specify θ_0 and P_0. P_0 is typically to use $P_0 = \epsilon I$, where I is the identity matrix and ϵ is some small positive constant.

According to [132], RLS-TD(λ) can be viewed as the extension of RLS-TD with $0 \leq \lambda \leq 1$. Firstly, we consider the LS-TD(λ) algorithm.

Instead of performing TD(λ) based on (4.39), LS-TD(λ) builds explicit estimates of the A matrix and b vector. We use the estimations $\tilde{A}$ and $\tilde{b}$, which are expressed as follows:

$$\tilde{A}_t = \sum_{i=1}^{t} A(X_i) = \sum_{i=1}^{t} z_i(\phi_i - \gamma\phi_{i+1})' \tag{4.58}$$

$$\tilde{b}_t = \sum_{i=1}^{t} b(X_i) = \sum_{i=1}^{t} z_i r_i. \tag{4.59}$$

Thus, θ^* can be estimated as $\tilde{A}_t^{-1}\tilde{b}_t$.

By making use of recursive least-squares methods so that the computational burden of LS-TD(λ) can be reduced. The weight update rules of standard RLS-TD(λ) are given by

$$\tilde{\delta}_t = r_t - (\phi_t - \gamma\phi_{t+1})'\theta_{t-1} \tag{4.60}$$

$$P_t = P_{t-1} - \frac{P_{t-1} z_t (\phi_t - \gamma\phi_{t+1})' P_{t-1}}{1 + (\phi_t - \gamma\phi_{t+1})' P_{t-1} z_t} \tag{4.61}$$

$$\theta_t = \theta_{t-1} + \frac{P_{t-1}}{1 + (\phi_t - \gamma\phi_{t+1})' P_{t-1} z_t} \tilde{\delta}_t z_t. \tag{4.62}$$

The detailed derivation of RLS-TD(λ) can be found in Appendix A1.

Under the similar assumptions of TD(λ) convergence proof in [98], RLS-TD(λ) is proved to converge with probability one. The proof refers to Appendix A2.

4.3.3 Learning with multi-step value iteration

During the mandatory intervals considered in our study, the multi-step t_M planning is adopted by

$$J(s_t) = \min_{a_t \in A(s_t)} E_{w_k} \left\{ \sum_{k=t}^{t+t_M-1} \gamma^{k-t} r_k + \gamma^{t_M} J(s_{t+t_M}) \right\}. \tag{4.63}$$

With function approximation $\tilde{J}(\cdot, \theta)$, (4.63) can be rewritten as

$$\hat{J}(s_t) = \min_{a_t \in A(s_t)} E_{w_k} \left\{ \sum_{k=t}^{t+t_M-1} \gamma^{k-t} r_k + \gamma^{t_M} \tilde{J}(s_{t+t_M}, \theta_t) \right\}. \tag{4.64}$$

And the controller aims to find the implemented action greedily by

$$a_t^* = \arg \min_{a_t \in A(s_t)} E_{w_k} \left\{ \sum_{k=t}^{t+t_M-1} \gamma^{k-t} r_k + \gamma^{t_M} \tilde{J}(s_{t+t_M}, \theta_t) \right\}. \tag{4.65}$$

According to(4.60), (4.61), and (4.62) in RLS-TD(λ), the multi-step planning of RLS-TD(λ) can be expressed as

$$\tilde{\delta}_t = \sum_{k=t}^{t+t_M-1} \gamma^{k-t} r_k - (\phi_t - \gamma^{t_M}\phi_{t+t_M})'\theta_{t-1} \tag{4.66}$$

$$P_t = P_{t-1} - \frac{P_{t-1} z_t (\phi_t - \gamma^{t_M}\phi_{t+t_M})' P_{t-1}}{1 + (\phi_t - \gamma^{t_M}\phi_{t+t_M})' P_{t-1} z_t} \tag{4.67}$$

$$\theta_t = \theta_{t-1} + \frac{P_{t-1}}{1 + (\phi_t - \gamma^{t_M}\phi_{t+t_M})' P_{t-1} z_t} \tilde{\delta}_t z_t. \tag{4.68}$$

The deviation of multi-step planning of RLS-TD(λ) can be found in Appendix A3.

4.4 Algorithm for adaptive traffic signal control

4.4.1 Algorithm for isolated intersection

For traffic signal control problem, the decision making of controller and the information of its surroundings are two important issues, especially when the controller is viewed as an agent, which has interaction with the environment over time and adjusts its behavior to receive better rewards. Based on the traffic dynamic model presented in Section 2, we specify some control variables in detail for the adaptive traffic signal control application implemented by ADP approach and the related RLS-TD(λ).

Specifying traffic control variables

There are three aspects of declared knowledges in traffic signal control at isolated intersection, where we prepare to use ADP with RLS-TD(λ) for the solution. The first one is the deterministic state transition in traffic dynamic model, which assures to be more facile than the stochastic state transition in practical problem solving by reinforcement learning. The second, it refers to the choice of traffic control decisions by implementing the three kinds of signal phase modes, namely FPS, VPS, and APS. And the last one, we should make specific definitions of traffic control variables in the particular learning method, i.e., ADP with RLS-TD(λ) learning. Details about these are discussed as follows.

Deterministic dynamic state As a sample model can generate random arrival information, it is usually to use the deterministic state transition rather than the stochastic state transition in the distributed model, which is weighed by probabilities when current state is transferred to next one with action taken. In many applications, it usually shows that the deterministic state transition by samples satisfying a certain distribution is easier to implement in practice. It is investigated in our case study in Section 3.3.

Control decisions As we mentioned before, the signal control at isolated intersection can adopt the three kinds of phase sequence modes, namely the FPS, VPS, and APS mode. We want to integrate these three modes into the ADP approach. Recall the system state $s_t = (k_t, x_t)$, we know that the elements in the signal state vector x_t are binary values. For example, the current signal status is that signals of line 1 and lane 5 are in green and others are in red, i.e., $x_t = (1, 0, 0, 0, 1, 0, 0, 0)^{\mathrm{T}}$. In FPS mode, the phase sequence is fixed, the action space $A_{\mathrm{FPS}}(s_t)$ has only 2 possible action vectors, namely $A_{\mathrm{FPS}}(s_t) = \{a_t^0, a_t^1\}$. The action of signal unchanged is $a_t^0 = (0, 0, 0, 0, 0, 0, 0, 0)^{\mathrm{T}}$ and the action of signal switching is $a_t^1 = (1, 1, 0, 0, 1, 1, 0, 0)^{\mathrm{T}}$ for next particular phase. The action a_t^1 indicates that signals on line 1 and lane 5 will change from being in green to red, and on lane 2 and lane 6 they do contrarily. As calculated by (2.13), the next changed signal state is $x_{t+1} = (x_t + a_t^1) \bmod_2 = (0, 1, 0, 0, 0, 1, 0, 0)^{\mathrm{T}}$. Similarly, in VPS mode, the action space $A_{\mathrm{VPS}}(s_t)$ has 4 possible action vectors, namely $A_{\mathrm{VPS}}(s_t) = \{a_t^0, a_t^1, a_t^2, a_t^3\}$. Except the a_t^0, all of the other actions can make a change sequentially from the current signal to another one. For example, if the current phase is numbered by 2, then, a_t^1, a_t^2, and a_t^3 can switch the current phase 2 to the next phase 3, 4, and 1, respectively. More complicated case is in APS mode, where there are total 12 various combinations. When the current combination has the right-of-way, the action of signal switch can be selected in the rest of 11 possibilities. By adding the unchanged action a_t^0, therefore, the action space $A_{\mathrm{APS}}(s_t)$ has 12 possible action vectors.

Control variables specified Recall the traffic dynamic system model and the suggested solution by ADP with RLS-TD(λ) learning approach, some details are expressed and redefined according to the practical control system. We will focus on the objective function of multi-step value iteration, which is defined in (4.63). The approximate solution will be obtained by solving (4.64) using linear function approximation $\tilde{J}(\cdot, \theta)$. The linear approx-

imator includes the feature-based function $\phi(s)$ on traffic state s and specified parameter vector θ. We can extract the features of state s expressed by the state (queuing vehicles and signal state) on each lane n of an intersection, namely $s_t(n) = (k_t(n), x_t(n))$. Thus, the linear approximator can be expressed as

$$\tilde{J}(s_t, \theta_t) = \sum_{n=1}^{N} \theta_t(n)' \phi_t(k_t(n), x_t(n)) = \theta_t' \phi_t \tag{4.69}$$

where $\theta_t = (\theta_t(n), n = 1, 2, \ldots, N)^{\mathrm{T}}$, and $\phi_t = (\phi_t(k_t(n), x_t(n)), n = 1, 2, \ldots, N)^{\mathrm{T}}$, N is the total number of lanes of intersection. We define $\theta_t(n)$ as

$$\theta_t(n) = (\theta_t^g(n), \theta_t^r(n))^{\mathrm{T}} \tag{4.70}$$

and assign $\theta_t^g(n)$ to queue length variable $k_t(n)$ if lane n receives green signal, or assign $\theta_t^r(n)$ otherwise. $\phi_t(k_t(n), x_t(n))$ is defined by

$$\phi_t(k_t(n), x_t(n)) = \begin{cases} (k_t(n), 0)^{\mathrm{T}}, & \text{if } x_t(n) = 1 \text{ (signal green)} \\ (0, k_t(n))^{\mathrm{T}}, & \text{if } x_t(n) = 0 \text{ (signal red).} \end{cases} \tag{4.71}$$

Thus, to verify the gradient $\nabla_{\theta_t} \tilde{J}(s_t, \theta_t)$ with respect to the components of θ_t, we have

$$\begin{aligned} \nabla_{\theta_t} \tilde{J}(s_t, \theta_t) &= \begin{pmatrix} \frac{\partial \tilde{J}(s_t, \theta_t)}{\partial \theta_t(1)} \\ \frac{\partial \tilde{J}(s_t, \theta_t)}{\partial \theta_t(2)} \\ \vdots \\ \frac{\partial \tilde{J}(s_t, \theta_t)}{\partial \theta_t(N)} \end{pmatrix} = \begin{pmatrix} \left(\frac{\partial \tilde{J}(s_t, \theta_t)}{\partial \theta_t^g(1)}, \frac{\partial \tilde{J}(s_t, \theta_t)}{\partial \theta_t^r(1)} \right)^{\mathrm{T}} \\ \left(\frac{\partial \tilde{J}(s_t, \theta_t)}{\partial \theta_t^g(2)}, \frac{\partial \tilde{J}(s_t, \theta_t)}{\partial \theta_t^r(2)} \right)^{\mathrm{T}} \\ \vdots \\ \left(\frac{\partial \tilde{J}(s_t, \theta_t)}{\partial \theta_t^g(N)}, \frac{\partial \tilde{J}(s_t, \theta_t)}{\partial \theta_t^r(N)} \right)^{\mathrm{T}} \end{pmatrix} \\ &= \begin{pmatrix} \phi_t(k_t(1), x_t(1)) \\ \phi_t(k_t(2), x_t(2)) \\ \vdots \\ \phi_t(k_t(N), x_t(N)) \end{pmatrix} = \phi_t. \end{aligned} \tag{4.72}$$

In summary, the online operation of adaptive traffic control algorithm using ADP with RLS-TD(λ) learning can be summarized in Algorithm 3.

4.4.2 Algorithm for traffic network

In a distributed system, traffic network control can be accomplished by optimizing the subsystems which are partially adjacent intersections of network. Independent network

Algorithm 3 ADP_RLS-TD(λ) for adaptive traffic signal control algorithm

choose an initial state $s_0(n) = (k_0(n), x_0(n))$, parameter $\theta_0(n) = (\theta_0^g(n), \theta_0^r(n))^{\mathrm{T}}$ for each lane n; set time $t = 0$, planning step $t_m = t_M$;
choose the action space A_{FPS} (or A_{VPS}, A_{APS});
while $t \leq T$ **do**
    if $t_m > 0$ **then**
        signal unchanged with $a_t^* = 0$;
        $t_m = \max(t_m - 1, 0)$;
    else
        for each $a_t \in A_{\mathrm{FPS}}(s_t)$ **do**
            pre-calculate and store the accumulated rewards and estimated values;
        end for
        find the optimal decision a_t^* using Eq. (4.65);
        if $a_t^* = 1$ **then**
            change signal into all-red;
            set $t_m = t_M - 1$;
        end if
    end if
    update functional parameter vector θ_t using Eqs. (4.66), (4.67), and (4.68);
    implement optimal decision a_t^* at time interval t;
    transfer system state $s_t(n)$ including signal state $x_t(n)$ and vehicle state $k_t(n)$ using Eqs. (2.13) and (2.14), respectively;
    $t = t + 1$;
end while

control and coordinated network control are given by algorithms based on the ADP with RLS-TD(λ).

4.4.2.1 Independent network control

In a distributed network, the intersection can be independently controlled. This can be viewed as the independent multi-agent system. It means that the signal controller (agent) makes decision only depend on its local information and not take other neighbor intersections into account. Thus, we just view each intersection as an independent agent and make near-optimal policy for the isolated intersection. Although in this way, it is not considered

about all intersections at network in a global way, but it operates easily and sometimes it performs well, especially an intersection needs a highly adaptive and autonomous control. For example, it is probably to implement the APS mode approaching this case.

Recall the network loading model and the traffic signal control model at multi-intersection. We can design an algorithm to find the solution of signal control for the independent intersections.

The independent network control algorithm refers to the extension of adaptive traffic signal control at isolated intersection by using ADP with RLS-TD(λ). In other words, each intersection $m = 1, 2, \ldots, M$ (M is the total number of intersections) at network can directly execute the Algorithm 3 for the self-control service.

4.4.2.2 Coordinated network control

In coordinated network control, the coordination between intersections is considered. This can be viewed as the coordinated multi-agent system. Despite the local information of each agent, the communicating information is shared mainly between adjacent intersections. The control system acts in a global view that joint action rather than local decision is taken.

Recall the definitions of tunable state of intersection m in (2.28) and the system state $\boldsymbol{s}_t = (s_t^1, s_t^2, ..., s_t^M)$. Recall the immediate cost function r_t defined in (2.34), which is based on tunable states at network. Similarly in (4.69), the feature-based function $\phi_t(\cdot)$ and the parameter $\boldsymbol{\theta}_t$ with dimension $N \times M$ are adopted to act as a linear function approximation. That is,

$$\tilde{J}(\boldsymbol{s}_t, \boldsymbol{\theta}_t) = \sum_{m=1}^{M} (\theta_t^m)^T \cdot \phi_t(s_t^m), \tag{4.73}$$

where $\theta_t^m = (\theta_t^m(n), n = 1, 2, ..., N)^T$ and $\phi_t(s_t^m) = (\phi_t(s_t^m(n)), n = 1, 2, ..., N)^T$. N is the total number of lanes at intersection and M is the total number of intersections at network. We define $\theta_t^m(n)$ as

$$\theta_t^m(n) = (\theta_t^{mg}(n), \theta_t^{mr}(n))^{\mathrm{T}} \tag{4.74}$$

and assign $\theta_t^{mg}(n)$ to tunable state $s_t^m(n)$ receiving green signal on line n at intersection m, or assign $\theta_t^{mr}(n)$ otherwise. $\phi_t(s_t^m(n))$ is defined by

$$\phi_t(s_t^m(n)) = \begin{cases} (ek_t^m(n) + (1-e)\tilde{k}_t^m(n), 0)^{\mathrm{T}}, & \text{if } x_t^m(n) = 1 \text{ (signal green)} \\ (0, ek_t^m(n) + (1-e)\tilde{k}_t^m(n))^{\mathrm{T}}, & \text{if } x_t^m(n) = 0 \text{ (signal red)} \end{cases} \tag{4.75}$$

where $e(0 \leq e \leq 1)$ is the tunable parameter. Note that the estimated value of the tunable state only refers to the parameters of local intersection for reducing the parameter dimension, but the tunable state transition is still related to the joint action between two adjacent intersections.

By using the ADP approach, the one step objective function is expressed by

$$\hat{J}(\boldsymbol{s}_t) = \min_{\boldsymbol{a}_t \in A} E\left\{r_t + \gamma \tilde{J}(\boldsymbol{s}_{t+1}, \boldsymbol{\theta}_t)\right\}, \tag{4.76}$$

the controller aims to find a sequence of joint actions greedily by

$$\boldsymbol{a}_t^* = arg \min_{\boldsymbol{a}_t \in A} E\left\{r_t + \gamma \tilde{J}(\boldsymbol{s}_{t+1}, \boldsymbol{\theta}_t)\right\}. \tag{4.77}$$

At each time step t, the joint action of all intersections at network needs to be calculated. At first, we discuss the system action space denoted by $A = \{A^1, A^2, ..., A^M\}$. Recall the system action $\boldsymbol{a}_t \in A$ and the phase definition. In 4-phase mechanism (FPS and VPS), define the signal phase $\varphi_1, \varphi_2, \varphi_3$, and φ_4 that receive green signal for lane combination {1, 5}, {2, 6}, {3, 7}, and {4, 8}, respectively. In FPS, the action space at intersection m can be defined as $A^m_{\text{FPS}} = \{a_t^m(\varphi_i), a_t^m(\varphi_{i+1})\}$ (where $i = 1, 2, 3, 4$ and if $\varphi_{i+1} > 4$, then φ_{i+1} is replaced by φ_1). In VPS, $A^m_{\text{VPS}} = \{a_t^m(\varphi_1), a_t^m(\varphi_2), a_t^m(\varphi_3), a_t^m(\varphi_4)\}$. For the proposed adaptive phase sequence (APS), the action space at intersection m can be defined as $A^m_{\text{APS}} = \{a_t^m(\varphi_1), a_t^m(\varphi_2), \ldots, a_t^m(\varphi_{12})\}$, where signal phase φ_c $(c = 1, 2, \ldots, 12)$ to receives green signal for lane combination G_c, which is defined in Tab. 3.3. For example, the VPS mode is used for intersection phase control. For the whole network, there are total 4^M possibilities of groups of joint action. In the case of Fig. 2.6, assume that the joint action at time t is $\boldsymbol{a}_t = (a_t^1(\varphi_1), a_t^2(\varphi_4), a_t^3(\varphi_2), a_t^4(\varphi_1), a_t^5(\varphi_3))$ and at time $t+1$, it changes to $\boldsymbol{a}_{t+1} = (a_{t+1}^1(\varphi_3), a_{t+1}^2(\varphi_2), a_{t+1}^3(\varphi_2), a_{t+1}^4(\varphi_3), a_{t+1}^5(\varphi_1))$, which can be illustrated by the matrices in Fig. 4.4. In addition, for security within the minimum green intervals and all red, changing traffic signal is not permissible.

$$
\begin{array}{c}
\boldsymbol{a}_t \\
\begin{pmatrix}
1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0
\end{pmatrix}
\end{array}
\rightarrow
\begin{array}{c}
\boldsymbol{a}_{t+1} \\
\begin{pmatrix}
0 & 0 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0
\end{pmatrix}
\end{array}
$$

Figure 4.4: Example of system action transition

In order to obtain the optimal $\boldsymbol{a}_t^*$, entire 4^M computations need to be traversed. With the small M, such as the network studied in our case, it is easier to exhaust the solutions to find the optimal $\boldsymbol{a}_t^*$. For a large network with big M or in APS mode with 12^M possibilities, the max-plus algorithm can be used to compute the near-optimal joint action by iterations, which could send locally optimized messages between connected nodes in the graph. This method refers to the research in [116, 118]. We just use the exhausted search to obtain $\boldsymbol{a}_t^*$.

In summary, the online coordinated network control is given in Algorithm 4.

4.5 Summary

ADP has a great advantage for the reduction of computational complexity. Rather than DP using backward recursive calculation to traverse all state space, ADP applies the function approximation and machine learning technique to simplify the calculation in Bellman's equation. In this chapter, we found that linear function approximation is easier to implement and train than neural network approximator. It has been well studied, especially in reinforcement learning. More importantly, the linear case offers enough a potential for the complexed discrete-time problem in practice, using the unsupervised reinforcement learning method, such as TD(λ). In our study, the recursive least-squares TD(λ) method is highlighted for the ADP with linear approximator. Meanwhile, we propose the formulation of multi-step iteration in RLS-TD(λ). Moreover, we design the real-time adaptive traffic signal control algorithms using ADP with RLS-TD(λ), both for the isolated in-

Algorithm 4 ADP_RLS-TD(λ) for coordinated traffic network control algorithm

choose an initial state $\boldsymbol{s}_0$, parameter $\boldsymbol{\theta}_0$; set $t = 0$;
initialize the tunable parameter e;
while $t \leq T$ **do**
    receive traffic arrival information $\boldsymbol{w}_t$ and $\tilde{\boldsymbol{w}}_t$ according to the micro-simulation dynamic model;
    use $\phi_t(\boldsymbol{s}_t)$ and $\boldsymbol{\theta}_t$ to calculate estimate value in (4.73);
    compute evaluation values in (4.76);
    find the optimal decision $\boldsymbol{a}_t^* = (a_t^m, m = 1, 2, ..., M)$ using (4.77) by exhausted search ($A_{\text{FPS}}, A_{\text{VPS}}$) or max-plus algorithm (A_{APS});
    for all intersection m **do**
        if changing traffic singal is not permissible **then**
            $\tilde{a}_t^m = \mathbf{0}$;
        else
            $\tilde{a}_t^m = a_t^m$;
        end if
    end for
    update functional parameter vector $\boldsymbol{\theta}_t = (\tilde{\theta}_t^m, m = 1, 2, ..., M)$ related to (4.66), (4.67), and (4.68);
    implement optimal decision $\tilde{\boldsymbol{a}}_t^* = (\tilde{a}_t^m, m = 1, 2, ..., M)$ at time interval t;
    transfer system state $\boldsymbol{s}_t$ including signal state $\boldsymbol{x}_t$ related to (2.13) and vehicle state $\boldsymbol{k}_t, \tilde{\boldsymbol{k}}_t$ in (2.29);
    $t = t + 1$;
end while

tersection and the whole network, which includes the independent multi-intersection and coordinated multi-intersection.

In next chapter, we will implement the proposed algorithms in numerical experiments of isolated intersection control and traffic network control. Some results by simulation will be compared and analyzed.

CHAPTER 5

APPLICATIONS AND RESULTS

5.1 Introduction

In this chapter, the ADP with linear function approximation using RLS-TD(λ) learning will be implemented to the signal control at isolated intersection and traffic network. The related algorithms are mentioned in Chapter 4. For both applications, performance measures are commonly defined. Different traffic demand scenarios are simulated for the implementation of ADP_RLS-TD(λ) comparing with methods cited from some literature.

In application to isolated intersection using the proposed algorithm, solutions including three kinds of phase sequence modes and fine planning will be considered. The phase sequence modes mentioned previously refer to the FPS, VPS, and APS. The planning solutions are normal planning with 2 s (one interval) per step and fine planning with 0.5 s (0.25 interval) per step. The evolutions of the functional parameters in approximation function will be illustrated and analyzed. In traffic scenarios, experiments are implemented by comparing the ADP_RLS-TD(λ) with other control methods with analysis of performances.

In application to traffic network using the proposed algorithm, solutions include the independent traffic network solution and the coordinated traffic network solution. They are also investigated in different phase sequence modes in 2-s solution. Before we do experiments in network case, an important part is traffic network loading, which provides traffic

environment for network. Thus, simulation results of a new vehicle-following model, which presents the vehicle behaviors in a microscopic way, will be described and analyzed. At last, we will evaluate ADP_RLS-TD(λ) compared by other control methods for traffic network control in different traffic scenarios. All experiments are implemented in MATLAB.

5.2 Application in isolated intersection

In this section, the case of signal control at isolated intersection depicted in Fig. 2.1, is simulated. Different traffic arrival rates are simulated under three patterns of traffic flow organization (or phase mode), i.e., FPS, VPS, and APS. Moreover, 2-s solution (normal) and 0.5-s solution (fine) are both considered in our algorithm ADP_RLS-TD(λ). If there is not special statement, results refer to the normal solution. Experiments are also implemented by using different methods.

5.2.1 Preparation

System settings

In the simulation of the traffic signal control system, at first, some value settings of parameters are given in Tab. 5.1. Actually, learning parameter η_t in TD(λ) is step-size scheduling as a time-varying form. From experience of some related studies, we use a constant leaning rate at η =0.001. Assumptions of the control system are mentioned in Section 2.3.2. We omit it to avoid repetition.

The other aspect of the system setting is about traffic data generation (or traffic demand in scenario). The method of arrival data generation in the case study of Chapter 3 is employed here. The random data generation is adopted by the computer simulation using Inverse Transformation Method (ITM) method, which meets a Bernoulli probability distribution. In reality, traffic data is detected from the inductive loops embedded upstream of each lane or other detecting techniques. The detected information contains traffic arrivals in the limited future time. In the simulation, traffic arrival per temporal step adopts the value of either 0 or 1, which satisfies the probability distribution. It is known that the binary variable represents one vehicle arrival during one interval or otherwise. If we want to plan the fine solution, the value of either 0 or 1 in each increment, which is the

Table 5.1: Intersection system parameter settings

Parameters	Definitions	Value settings
T	simulation period	40000 intervals [1]
N	total lanes at intersection	8
$g_{\min}$	minimum green time	3 intervals
g_{int}	inter-green (all red) time	1 interval
t_M	mandatory multi-step	4 intervals
$\theta_0^g(n)$	initial parameter to green signal	5 or 3 [2]
$\theta_0^r(n)$	initial parameter to red signal	5 or 3
γ	discount factor	0.90
η	learning rate constant	0.001
ϵ	parameter in matrix P_0	0.01
S_a	saturation (departure) flow	1 veh/int=1800 veh/h

[1] 1 interval=2 seconds.
[2] initial value 5 for Traffic Scenario A-x, B-x, and initial value 3 for Traffic Scenario C defined later.

Table 5.2: Asymmetric and symmetric average arrival rates for intersection

Traffic Scenario	Arrival rate (veh/int)	Traffic volume by ITM (veh/h)
A-1	(0.05, 0.20, 0.05, 0.20)	(188, 702, 201, 705)
A-2	(0.10, 0.20, 0.10, 0.20)	(350, 722, 365, 735)
A-3	(0.15, 0.20, 0.15, 0.20)	(527, 720, 562, 715)
B-1	(0.10, 0.10, 0.10, 0.10)	(350, 358, 342, 365)
B-2	(0.15, 0.15, 0.15, 0.15)	(560, 562, 567, 548)
B-3	(0.20, 0.20, 0.20, 0.20)	(742, 725, 740, 735)

component of one interval, randomly appears to satisfy that the sum of the values in all increments is either 0 or 1. For example, if there are 4 increments in one interval and the situation of vehicles arriving is (0, 0, 0, 0) or (0, 1, 0, 0), etc.

Different traffic arrival rates are tested in our study. Simulator generates all the traffic arrival data as the way of traffic rates in FPS or VPS combination (G_1, G_2, G_3, G_4). Asymmetric and symmetric traffic arrival rates as well as the corresponding traffic volumes are shown in Tab. 5.2. Notice that the highest traffic arrival rate in B-3 owns the intensity, calculated by the method in [134], almost 0.9 which is already close to road saturation. Traffic Scenarios A-x and B-x keep the arrival rates unchanged during the simulation. We will also investigate another scenario called Traffic Scenario C, which gives changing arrival rates in time-varying during the simulation. In Traffic Scenario C,

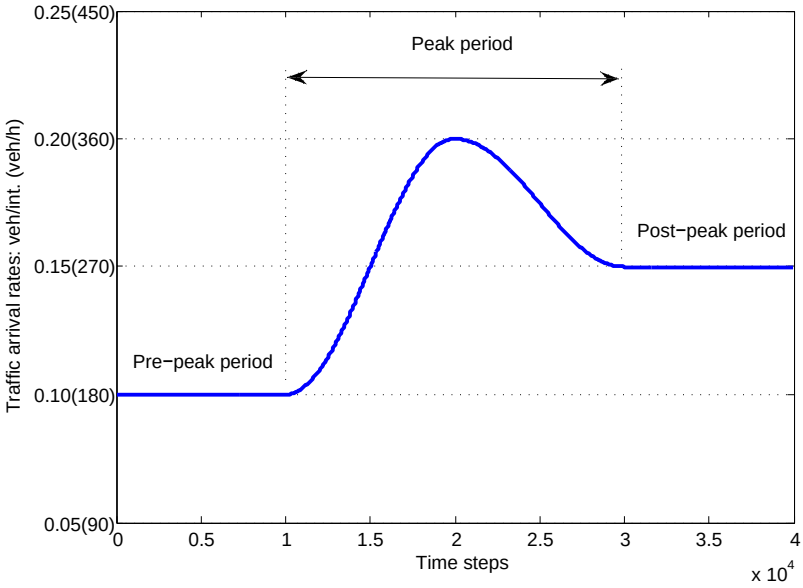


Figure 5.1: Traffic Scenario C-flow profile on average arrival rates during the simulation

symmetric arrival rates are processed smoothly ranging from 0.10 veh/int to 0.20 veh/int, as shown in Fig. 5.1.

Performance measures

In the case of the signal control at isolated intersection, performance measures are the average traffic delay and vehicle queue length. In detail, we note that

- *Average delay*. The average delay is donated by D, which is expressed in seconds and calculated according to the following form in OPAC system [134]. That is

$$D = \frac{T_{\mathrm{D}}}{T_{\mathrm{A}}} = \frac{2\sum_{n}^{N}\sum_{t}^{T} k_t(n)}{\sum_{n}^{N}\sum_{t}^{T} w_t(n)} \tag{5.1}$$

  where T_{D} (veh·int) is to measure the total vehicle-intervals (in 2-s units) which is the sum of queue lengths of all lanes N during the simulation period T; T_{A} (veh) is the total number of vehicle arrivals or vehicles released into the intersection.

- *Queue length*. The queue length is calculated by the number of waiting vehicles on lane. We also use the average of queue length at intersection when comparing performance as a whole.

Comparing methods

In order to see the characteristics of RLS-TD(λ) learning in traffic signal control, firstly, the comparisons of the properties of RLS-TD(λ) learning and TD(λ) learning under ADP approach are illustrated in a simulation way. Secondly, regarding to the performance of ADP with RLS-TD(λ) learning in FPS, VPS, and APS patterns, we compare the proposed algorithm with Webster and Haijema-MDP in the optimal FC and other algorithms in adaptive control (AC), such as Greedy algorithm, Heuristic algorithm, and Q-learning. The details about these algorithms are introduced as follows.

- *FC*. In this control method, Webster [135] and Haijema-MDP [31] algorithms are considered. In Haijema-MDP, fixed cycle time is optimized by the evaluation of MDP based on expected traffic arrival rates. This approach has a better phase time distribution than Webster's method when the asymmetric traffic volume appears at intersection.

- *Greedy algorithm*. In this method, the multi-step planning evaluation function lacks the part of heuristic information and has several steps for look forward planning. Decisions are greedily chosen by evaluating the reward function.

- *Heuristic algorithm*. This is a forward search dynamic programming algorithm, shorten by FSDP (see Chapter 3, Section 3.3), and the heuristic information is used in Bellman's equation. Because this method has a global optimization solution, computation complexity is very high by increasing the planning forward steps. Considering the detected information of the limited future time (actually, the extended information could be solved by prediction model), we set planning horizon $T_p = 16$.

- *Q-learning*. In this method, normal rules for updating value function are used.

$$Q_{t+1}(s,a) = Q_t(s,a) + \eta_t(r_t(s,a) + \gamma \min_{a' \in A} Q_t(s',a') - Q_t(s,a)), \tag{5.2}$$

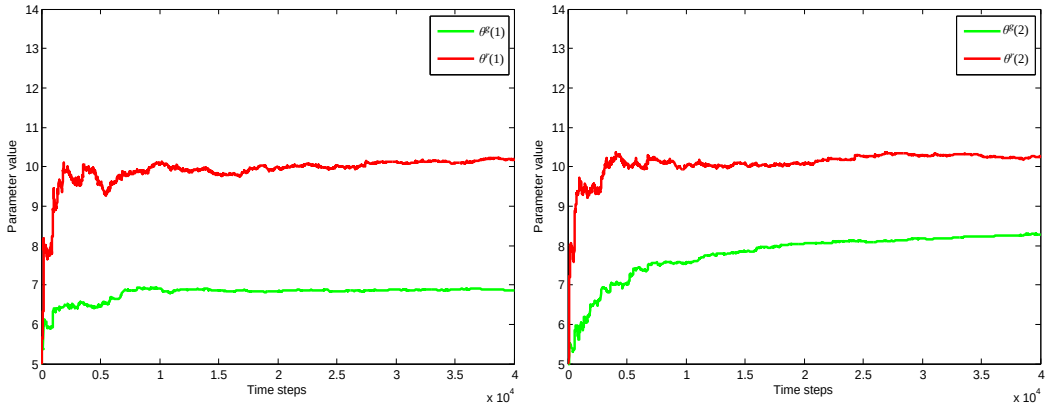
  where η_t is learning rate, s' is the transfered state from s taken action a. In this method, the entire traffic states need to be looped over because their values need to be updated overall. So that, states are reduced by choosing three density levels, i.e., low, medium, or high. Thresholds between different levels are various settings based on the traffic arrival rates.

5.2.2 Functional parameters simulation

The functional parameters in linear function approximation have some interesting properties when we do the simulation. Evolutions of the functional parameters in time-varying can help us to understand the performance of related learning algorithms. We will list and compare some results of the functional parameters in the TD(λ) and RLS-TD(λ) learning.

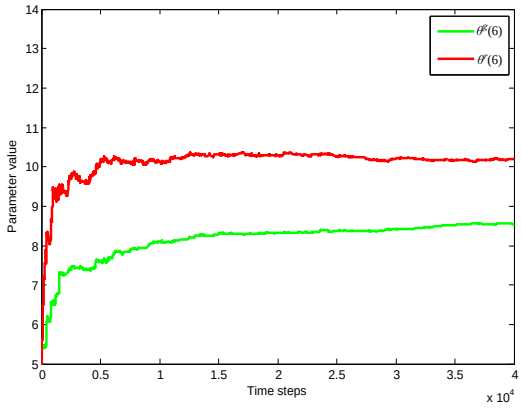
At first, let us look at the parameter properties of RLS-TD(λ) learning, illustrated in Fig. 5.2. By the samples of parameter evolutions, it is clearly shown that these parameters trend to relative steady levels after some steps. Differences appear in the parameters θ_t^g, which correspond to feature-extraction (basis) function when receiving green signal. As we can see, the values of steady level in $\theta_t^g(1)$ (Fig. 5.2(a)) and $\theta_t^g(7)$ (Fig. 5.2(d)) are smaller than those in $\theta_t^g(2)$ (Fig. 5.2(b)) and $\theta_t^g(6)$ (Fig. 5.2(c)). This could be possibly explained by that, in Traffic Scenario A-2, the arrival rates of lane 1, 2, 6, and 7 are 0.1, 0.2, 0.2, and 0.1 veh/int, respectively. Therefore, lane 1 and 7 receive less green durations than lane 2 and 6, and go faster to reach the lower values at steady level. As for the parameters θ_t^r giving almost the same values at steady levels, it indicates that waiting time for red signal on each lane is nearly the same. In other words, it is fair for vehicles waiting at the intersection.

In theory, we know that the performance of functional parameter in RLS-TD(λ) is better than the conventional TD(λ) learning referred to the literature. For the traffic signal control simulation in Traffic Scenario B-3, we compare the experiment results between these two learning techniques, as shown in Fig. 5.3. Obviously, the parameters $\theta_t^g(n)$ and $\theta_t^r(n)$ in RLS-TD(λ) obtain much less variance than those in TD(λ) (in case $\lambda = 0$, $n = 1$). Moreover, RLS-TD(λ) refers to the earlier stable trend, especially shown by the parameter $\theta_t^g(n)$. In other experiments with different parameter settings of λ and n, similar results are also obtained. It verifies that the recursive least-squares approach can speed up the convergence of TD(λ) learning process in our case.

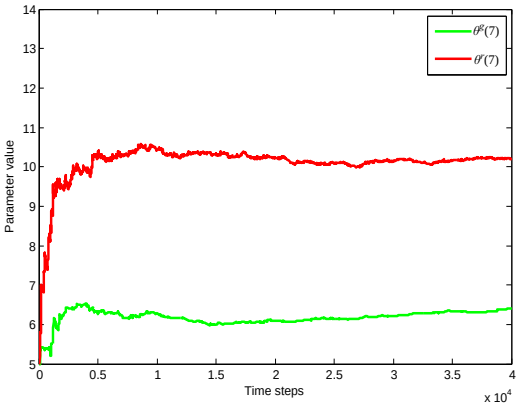


(a) $n = 1$

(b) $n = 2$



(c) $n = 6$



(d) $n = 7$

Figure 5.2: Evolutions of functional parameters by using ADP with RLS-TD(λ) learning (in Traffic Scenario A-2, APS mode with λ=0, n=1, 2, 6, and 7)

It is found that the TD(λ) learning has the relative large variances in the normal 2-s solution. Interestingly, as shown in Fig. 5.4, we find that TD(λ) reduces vibrations of parameters when the fine 0.5-s solution is implemented. Moreover, $\theta_t^g(n)$ ($n = 2$) in 0.5-s solution has the larger values than those in 2-s solution. It indicates that in the condition of traffic arrival on lane 2, it is proper to give more intervals and times in 0.5-s solution than the case of 2-s solution. Actually, it is more clear to show these phenomena in TD(λ) than in RLS-TD(λ). That is why we use TD(λ) to analyze the difference between the 2-s solution and the 0.5-s solution.

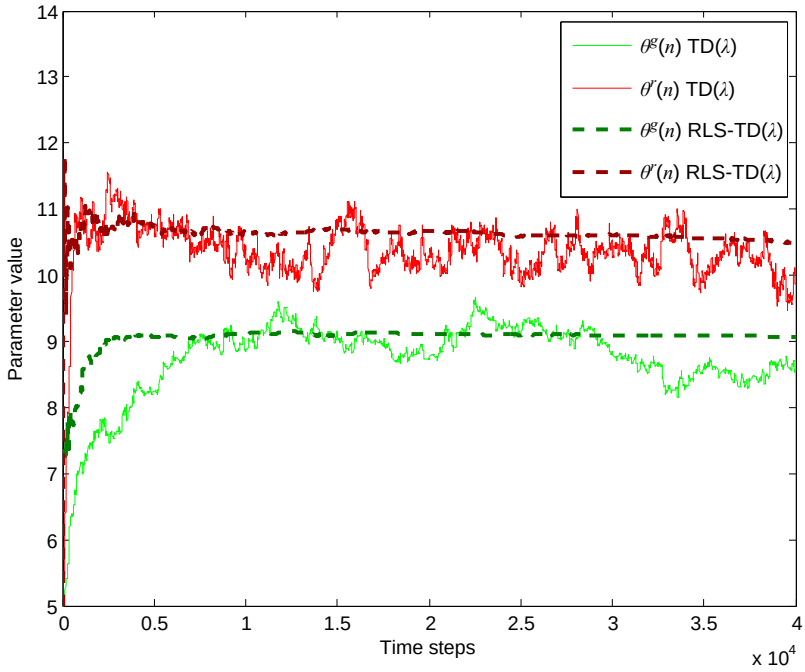


Figure 5.3: Comparisons of functional parameters by using ADP method between RLS-TD(λ) and TD(λ) learning (in Traffic Scenario B-3, FPS mode with $\lambda = 0$, n=1)

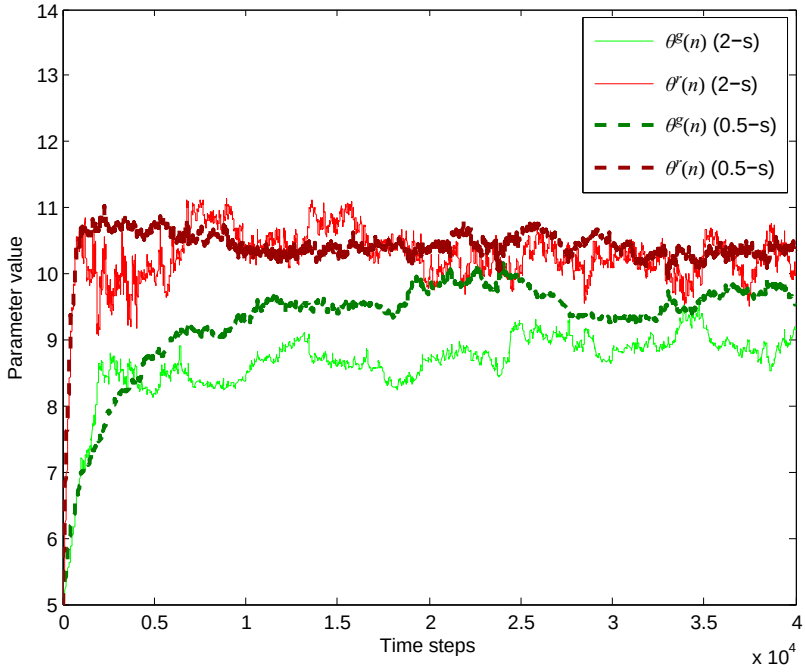
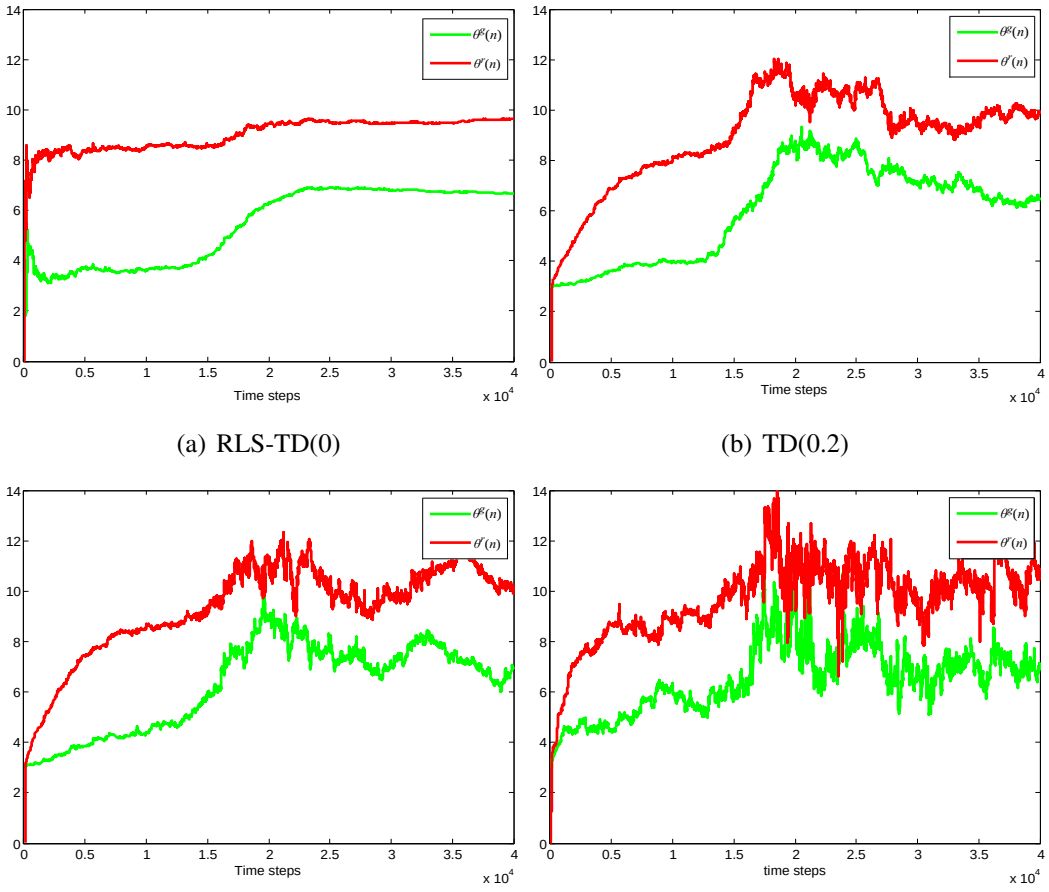


Figure 5.4: Comparisons of functional parameters by using TD(λ) learning between 2-s solution and 0.5-s solution (in Traffic Scenario B-3, VPS mode with $\lambda = 0$, n=2)

The discussion above refers to the scenarios with fixed traffic arrival rates. We also investigate the parameter properties of RLS-TD(λ) and TD(λ) in Traffic Scenario C, which owns various traffic arrival rates during the simulation. In Fig. 5.5, we can find that the parameters $\theta_t^g(n)$, $\theta_t^r(n)$ $(n = 1)$ by ADP with RLS-TD(λ) $(\lambda = 0)$ occur much less frequently than those by ADP with TD(λ), and the former are faster to reach to stable tendency, which has the same conclusion from Fig. 5.3. While TD(λ) learning updates the parameters easily effected by the current data with some vibrations. Moreover, the differences of performances appear in ADP with TD(λ) control when λ are set to be different values, see Fig. 5.5(b), Fig. 5.5(c), and Fig. 5.5(d). Obviously, setting λ to be 0.2 has less vibration than the others.

From the characteristics of parameters discussed above, we really care about influences of the ADP approaches using RLS-TD(λ) learning and TD(λ) learning for practical problem. In Fig. 5.6, control performances of average delay are compared between the RLS-TD(λ) learning and the TD(λ) learning with different settings of λ. Two typical Traffic Scenario A-2 and B-3 are implemented, respectively. It is clearly shown that the performance of RLS-TD(λ) learning is better than TD(λ) learning, by reason of the advantage for updating parameters. In the condition of higher demands in Traffic Scenario B-3, TD(1) even obtains a bad value. In Sutton's view [58], TD(λ) can be understood as one particular way of averaging n-step backups. This average contains all the n-step backups, each weighed proportional to λ^{n-1}, where $0 \leq \lambda \leq 1$. If $\lambda = 0$, then the overall backups reduce to the first component, i.e., the one-step TD backup, whereas if $\lambda = 1$, then the overall backups reduce to the last one component. We can conclude from the simulation results that the small λ, which indicates a nearby few backups mainly determining the TD error, can guarantee the satisfied performance. It could be explained by that the values of traffic states calculated in short time backups are better than many steps backups when traffic states are frequently visited.

From the analysis of parameter properties and performance of delay, RLS-TD(λ) is superior to TD(λ) learning under the ADP approach for adaptive traffic signal control. By the following experiments, performances of ADP with RLS-TD(λ) learning comparing with other control methods are discussed in the next section.



(a) RLS-TD(0)

(b) TD(0.2)

(c) TD(0.5)

(d) TD(1)

Figure 5.5: Comparisons of functional parameters by using ADP method between RLS-TD(λ) and TD(λ) learning (in Traffic Scenario C, APS mode with n=1)

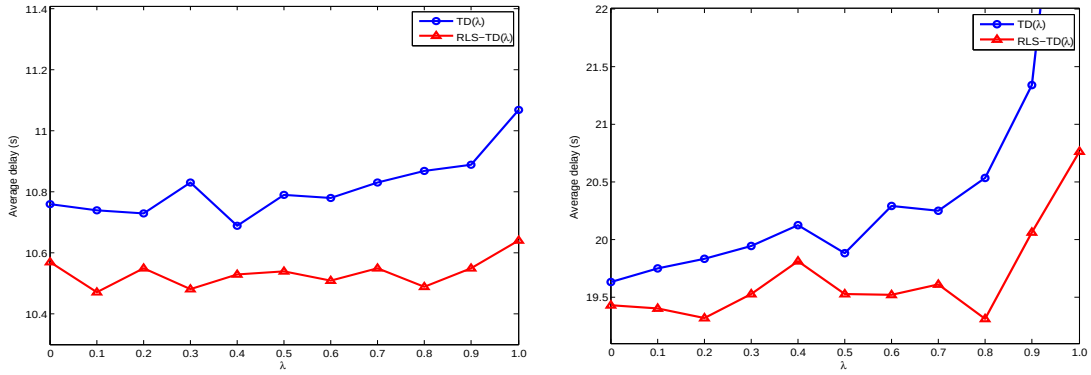
5.2.3 Comparisons and analysis

In this section, we mainly compare ADP_RLS-TD(λ) with different control methods in the aspects of different phase modes (FPS, VPS, and APS) and the two kinds of planning solutions (2-s and 0.5-s solution).

5.2.3.1 Different phase mode solutions

In this part, we focus on the comparisons of control methods in different phase modes under the normal 2-s solution. Some analysis is given.

In Tab. 5.3 and Tab. 5.4, results of average traffic delay are given by using different algorithms in Traffic Scenario A-x (asymmetric) and B-x (symmetric). Being different



(a) A-2 (b) B-3

Figure 5.6: Comparisons of average delays by using ADP method between RLS-TD(λ) and TD(λ) learning in APS

from FC methods, AC methods that operate in an adaptive way can adjust phase duration and phase sequence in time. Thus, the performances of FPS, VPS, and APS modes are included. Analysing the FC methods, Haijema-MDP works better than Webster's method. In fact, this will be more obvious in the unbalanced arrival rates if we set traffic scenario with the arrival rate of being (0.05, 0.20, 0.05, 0.20) in A-1. In the following study, Haijema-MDP represents the FC methods to compare with the AC methods. As a whole, the AC methods are better than the FC methods, except some results in the Greedy algorithm, such as the FPS and VPS modes in Traffic Scenario A-3, B-3. It indicates that the adaptive control mechanism maybe generate a bad control policy when the algorithm is not suitable for this kind of control. Without learning techniques, the heuristic algorithm FSDP is much better than Greedy algorithm. In the learning methods, the proposed ADP_RLS-TD(λ) algorithm (in case λ=0) performs very well, especially in the case of higher flow rates in Traffic Scenario B-3, owning the delay reduction about 7 s comparing with Q-learning in FPS and VPS, and 2.7 s in APS. More importantly, the FPS, VPS, and APS modes in each control algorithm make a difference in traffic delays.

In order to see performances of different phase modes, we pick up the results of A-2 and B-3 and obtain the improvements of these modes illustrated in Fig. 5.7.

By using proper control algorithms, APS outperforms on average traffic delays than FPS and VPS, which are related to the cyclic and acyclic signal control ways, respectively. APS can operate in a highly adaptive way, where not only the phase sequence is acyclic but also all the possible phase chances are mostly traversed and adaptively

Table 5.3: Results of average delay in asymmetric rates

Traffic Scenario			A-1			**A-2**			A-3	
FC delay(s)	Webster		48.25			25.25			30.24	
	Haijema MDP		18.85			23.68			29.79	
AC delay(s)	Phase mode	FPS	VPS	**APS**	FPS	VPS	**APS**	FPS	VPS	**APS**
	Greedy	18.91	13.95	8.40	23.21	21.88	12.37	35.27	35.49	15.76
	FSDP	15.86	10.39	5.90	18.65	14.80	8.74	21.09	19.47	10.04
	Q-learning	17.15	13.18	8.25	20.03	18.23	12.02	28.43	27.55	14.86
	ADP_RLS-TD(0)	18.29	11.86	7.96	20.60	17.44	10.57	26.75	26.03	14.17

Table 5.4: Results of average delay in symmetric rates

Traffic Scenario			B-1			B-2			**B-3**	
FC delay(s)	Webster		16.05			24.55			50.05	
	Haijema MDP		15.08			23.56			49.02	
AC delay(s)	Phase mode	FPS	VPS	**APS**	FPS	VPS	**APS**	FPS	VPS	**APS**
	Greedy	14.50	11.43	7.42	23.91	23.11	12.25	59.90	55.85	24.52
	FSDP	12.14	9.50	5.58	18.16	15.74	9.01	45.30	40.53	12.58
	Q-learning	14.15	11.20	7.38	20.86	19.39	11.87	49.32	48.90	22.10
	ADP_RLS-TD(0)	14.42	11.18	7.21	20.65	18.77	11.40	42.24	41.91	19.43

selected. Consequently, the average delays in APS are about 44.48% and 54.65% improvements comparing with those in FPS in Traffic Scenario A-2 and B-3, respectively. As for VPS, the average delay reductions are only 14.58% in Traffic Scenario A-2 and 5.51% for the higher demands in Traffic Scenario B-3.

However, in Fig. 5.7, it is shown that FSDP method looks like an appropriate approach as a whole with the lower traffic delays and higher improvements. Actually, FSDP method costs much time in the total simulation because of its large state space computation. This conclusion is mentioned in the previous case study, in Section 3.3. We will also give some related knowledge to discuss it later.

5.2.3.2 Fine planning solution

In [58], Sutton presented the evidence that planning in very small steps may be the most efficient approach even on pure planning problems if the problem is too large to be solved exactly. In [19], a fine solution was obtained from perturbation learning which was better than a coarse solution. We just investigate a fine step to see whether the ADP with RLS-TD(λ) learning is also suitable for these cases. Let a small step be 0.25 interval (0.5 s)

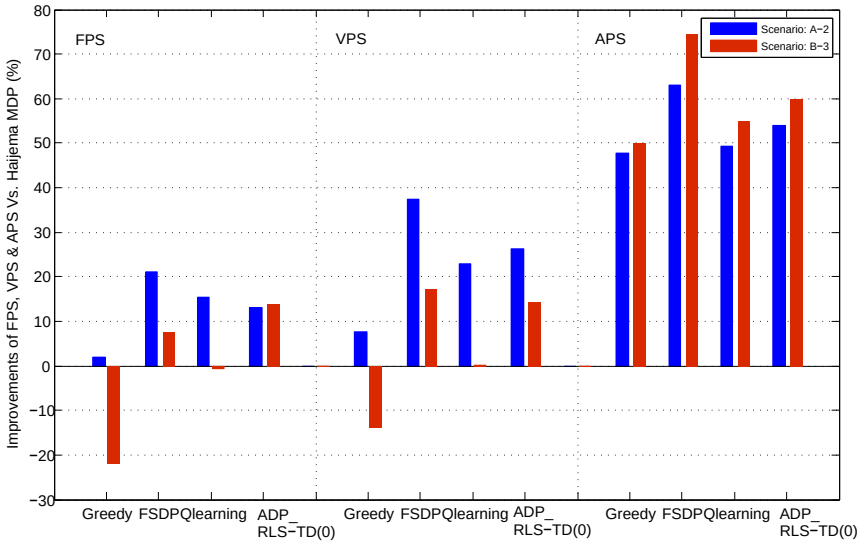


Figure 5.7: Improvements of average delays by using different methods in FPS, VPS, and APS, comparing with the Haijema-MDP in FC

Table 5.5: Results of average delay in fine solution

Traffic Scenario	A-1		A-2		A-3		B-1		B-2		B-3	
Phase mode	FPS	APS	FPS	APS	FPS	APS	FPS	APS	FPS	APS	FPS	APS
ADP_TD(0)	15.83	7.51	15.57	9.32	15.78	10.74	11.15	7.20	13.52	9.55	20.79	14.20
Delay ratio[1] (%)	-14.9	-52.6	-25.7	-39.0	-40.8	-31.9	-22.5	-35.4	-34.5	-29.4	-52.1	-31.7
ADP_RLSTD(0)	14.77	7.27	14.78	9.30	15.79	10.71	11.22	7.08	13.30	9.39	20.09	13.41
Delay ratio[1] (%)	-19.2	-50.8	-28.3	-37.1	-41.0	-32.2	-22.2	-36.9	-35.6	-29.4	-52.4	-33.3

[1] Delay ratio in FPS, is defined by $100\%(D_{\mathrm{FPS(0.5-s)}} - D_{\mathrm{FPS(2-s)}})/D_{\mathrm{FPS(2-s)}}$; Delay ratio in APS, is defined by $100\%(D_{\mathrm{APS(0.5-s)}} - D_{\mathrm{FPS(0.5-s)}})/D_{\mathrm{FPS(0.5-s)}}$.

in the experiments. Considering ADP with TD(0) and RLS-TD(0) algorithms, the results of traffic delay are really better than the original step setting of being one interval, as shown in Tab. 5.5. Meanwhile, APS mode is obviously better than FPS mode with delay reductions from -29.4% to -52.6%.

On the other hand, the performance of queue length in a microscopic way is shown in Fig. 5.8. Note that the first line bars represent arrival vehicles; the second and the third line bars are green indications counted by intervals (2 s per interval). It can be seen that different green signal settings result in different number of queuing vehicles and evolutions. By planning the green durations more intelligently and adaptively in this case, 0.5-s solution performs better than 2-s solution as a whole.

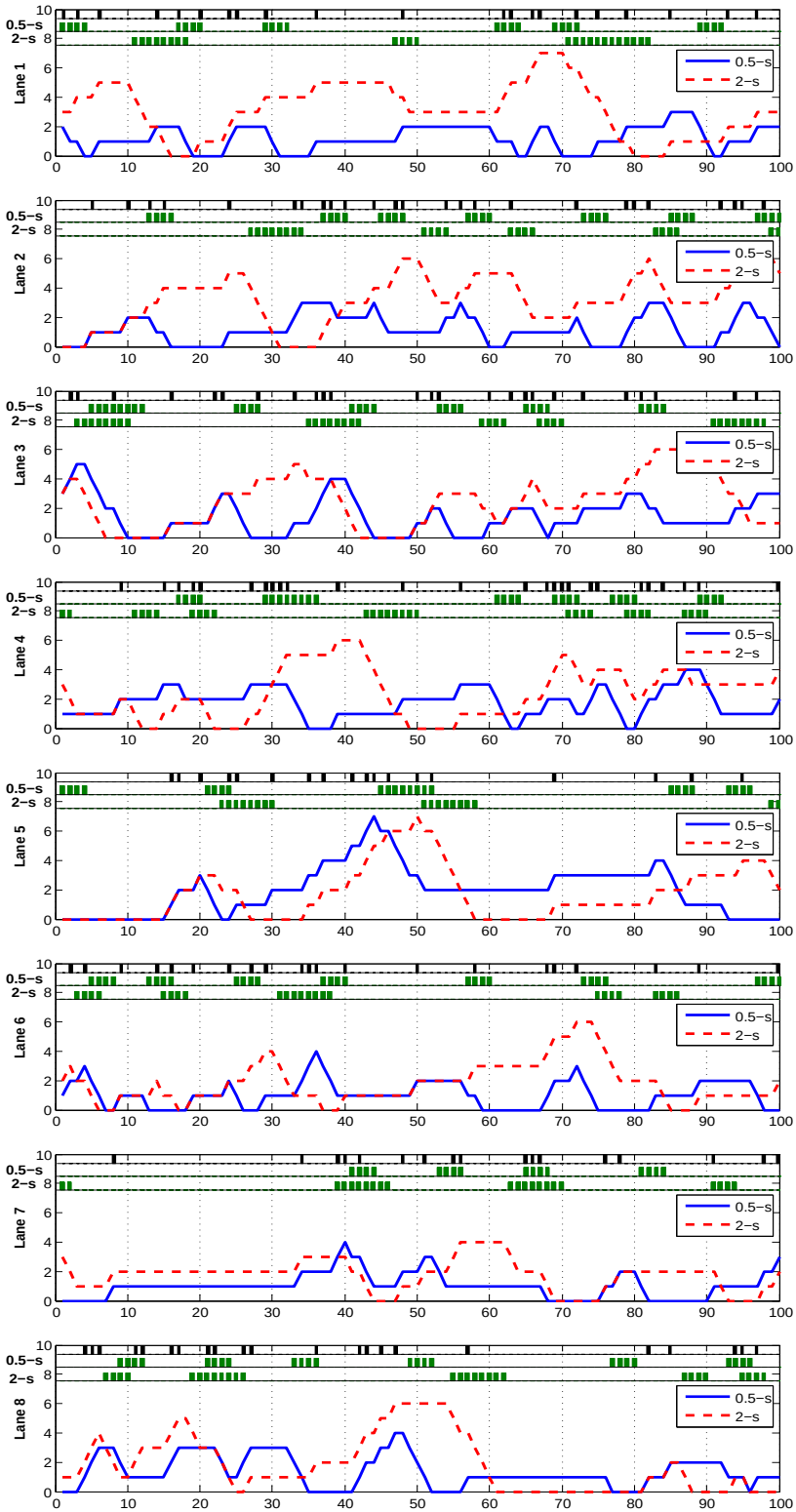
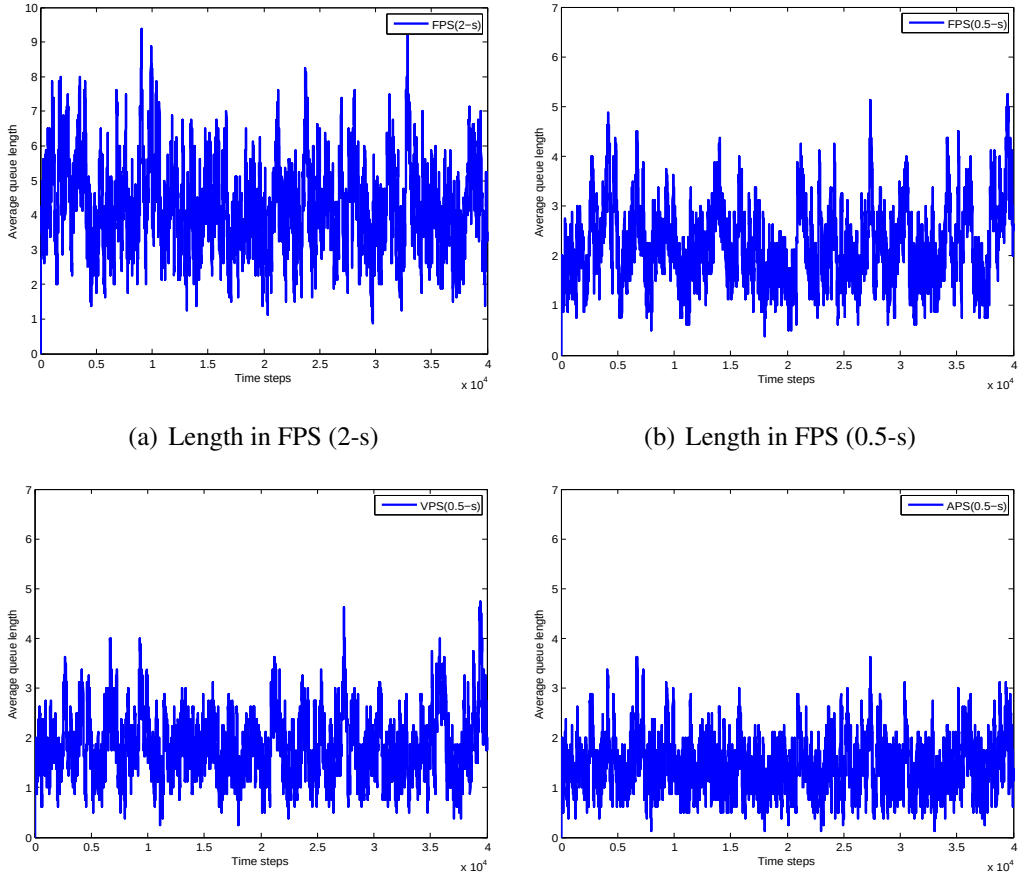


Figure 5.8: Evolutions of queue length by using ADP_RLS-TD(λ) ($\lambda = 0$ in APS at 0.5-s and 2-s solutions in Traffic Scenario B-3)



(a) Length in FPS (2-s)

(b) Length in FPS (0.5-s)

(c) Length in VPS (0.5-s)

(d) Length in APS (0.5-s)

Figure 5.9: Evolutions of average queue length by using ADP_RLS-TD(λ) ($\lambda = 0$ in Traffic Scenario B-3)

The performances of average queue length at the whole intersection are also clearly shown in Fig. 5.9, which demonstrates the relevant effects executed by different phase control patterns. Obviously, The flexible and adaptive control mode APS works very well with less variance and lower queue length. Accordingly, these fine solutions cost more computational time. However, it is also enough to guarantee online operation by using the ADP with RLS-TD(λ) learning.

Computation time

Besides the traffic delay and queue length, computation efficiency is an important performance index of the algorithm. In Tab. 5.6, it is clearly shown that the ADP approaches

Table 5.6: Comparisons of run time for isolated intersection

Methods	Haijema-MDP	FSDP			ADP_TD(λ)				ADP_RLS-TD(λ)			
Phase mode	Fixed	FPS	VPS	APS	FPS	VPS	APS	APS[1]	FPS	VPS	APS	APS[1]
Run time (min)	0.09	4.5	10.4	166.3	0.27	0.58	1.06	4.15	0.30	0.65	1.12	4.40

[1] fine 0.5-s solution.

with TD(λ) and RLS-TD(λ) learning take a little time for computation during the whole simulation. Even for APS mode, it runs about 0.002 s per step. On contrary, FSDP method costs much time, especially in APS mode. Therefore, a pure optimized DP method such as FSDP can't conveniently expand to a complex application. For example, it is hard to work in the cases of APS mode and the finer planning solution, or the traffic network control referring to a large state space. While, the approximate DP combining with the learning technique RLS-TD(λ) could show great potential to tackle the dimension problem, as well as good performances on average delay and queue length. The computational efficiency of ADP with RLS-TD(λ) learning could make it fully capable for the online control at the isolated intersection. In the next section, we investigate the ADP_RLS-TD(λ) algorithm for the traffic network control.

5.3 Application in traffic network

In this section, the case of signal control at the traffic network depicted in Fig. 2.6, is simulated. Different traffic arrival rates are simulated in 2-s solution under the three phase modes, i.e., FPS, VPS, and APS. ADP_RLS-TD(λ) and the other control methods are implemented in the experiments of the independent and coordinated traffic network.

5.3.1 Preparation

System settings

At first, besides the same parameters in Tab. 5.1, we give additional value settings of parameters in the traffic network system, as shown in Tab. 5.7. In detail about designing the urban traffic network, the equal size of place is set to be the head-to-head minimum distance of queuing vehicles, it assumes to be 8 meters. The lengths of links at network have the setting values of being 45~60 unit places (360~480 m). Assume that the maxi-

Table 5.7: Network system parameter settings

Parameters	Definitions	Value settings
T	simulation period	40000 intervals [1]
N	total lanes at intersection	8
M	total intersections at network	5
L_l	inside link lengths (unit place [2])	$L_5 = L_{17} = 45$ $L_7 = L_{15} = 60$ $L_6 = L_{18} = 48$ $L_8 = L_{16} = 50$
$v_{\max}$	maximum vehicle velocity	4 place/interval
P_{mer}	merging proportions	[30%,70%]
P_{div}	diverging proportions	[80%,20%]

[1] 1 interval=1 step=2 seconds.
[2] 1 unit place = 8 m.

mum of velocity is 4 place/int (57.6 km/h). Note that, near the entrance of lane (around 10 places), vehicles can be accelerated gradually with the constraints of final velocity underlying the vehicle-following model. Traffic distributions for left-turn, straight forward, and right-turn are determined as proportion 30%, 56% ($70\% \times 80\%$), and 14% ($70\% \times 20\%$), respectively.

Similarly, traffic demands in the network system are set by using the random traffic data generation, which satisfies the Bernoulli 0-1 distribution. Traffic data inputs into the network from the entrance link. They are shown in Fig. 5.10. There are total three kinds of traffic demands expressed by low, medium, and high arrival rates in Traffic Scenario D-x.

Performance measures

In the case of signal control at the network, performance measures include three parts. Besides average traffic delay and vehicle queue length, current vehicle mean speed is purposely added. In detail, we note that

- *Total average delay*. It is an evaluation of delay at whole network and expressed in seconds. According to [110], it is calculated by

$$T_{\text{AD}} = \sum_{m=1}^{M} T_{\text{D}}/T_{\text{N}} \tag{5.3}$$

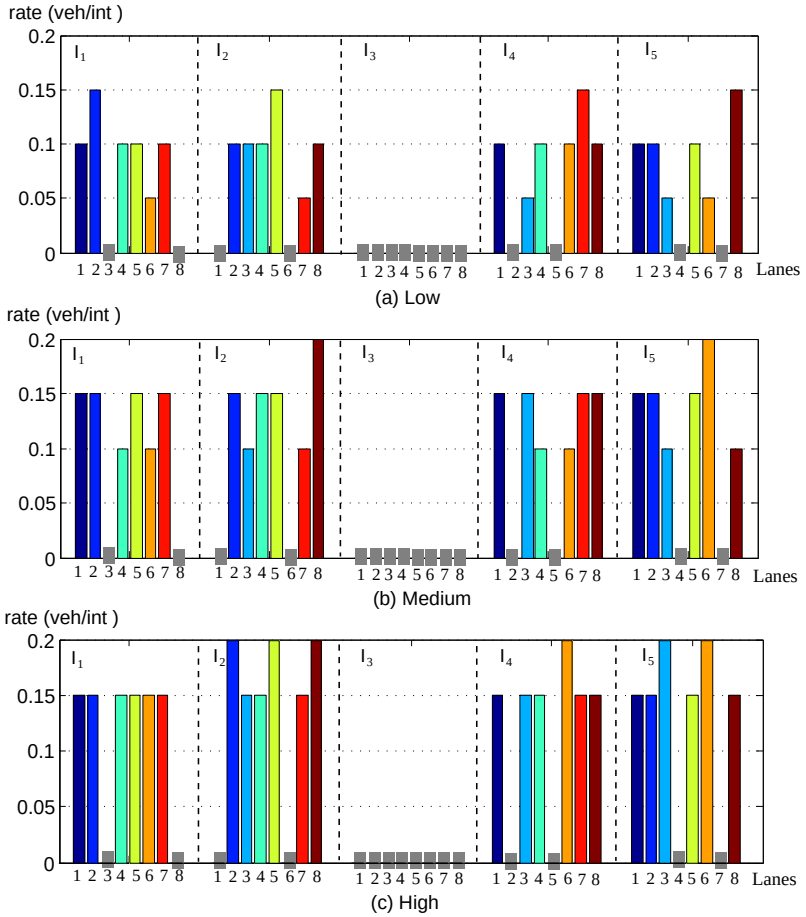


Figure 5.10: Traffic demand for network, except the inside link lanes represented by short gray bars, (a) D-1: low; (b) D-2: medium; (c) D-3: high

where M is the number of intersections, T_{D} is the delay experienced by vehicles at one intersection and T_{N} is the total number of vehicles released into the network.

- *Queue length.* It is calculated by the number of waiting vehicles on lane. The evaluation of traffic signal control at intersection gives whether or not the stability and fairness of queue length on each lane. For simplicity, we evaluate it by the maximum queue length associated to each phase.

- *Current vehicle average speed.* This performance measure presents traffic travel efficiency inside network. It is calculated as total moving vehicle speeds divided by total number of vehicles at network and expressed in place/int.

Comparing methods

To see the performance of ADP_RLS-TD(λ) algorithm for independent network and coordinated network, the independent multi-intersection extended from Algorithm 3 and the coordinated one in Algorithm 4 are both implemented. In order to compare the proposed traffic network control algorithms with others, we introduce the following algorithms.

- *Fixed time (FC)*. This algorithm, at first, calculates the optimal fixed cycle time and phase splits according to Webster's method considering the expected traffic arrival rates at each intersection (assuming arrival rate in the inside link has an average value). Secondly, the setting of initial phases and the offset between intersections are determined by numerical experiments. For example, in short-term 5000 intervals, execute that Step 1: determine the initial phase of each intersection; Step 2: the initial phase starting time is determined by looping the discrete intervals of the phase duration. The best performance of each traffic scenario in FC is compared with the proposed algorithm.

- *Self-Organization Algorithm (SOA)*. This algorithm basically uses the control mechanism named SOTL, which is proposed in [136]. The basic rule is that a counter is used to calculate the approaching or waiting vehicles at each time step on the lane of red signal. After minimum green time of the current phase and satisfying the conditions of crossing platoons, green light can switch to red one with counter being zero while the counter reaches a threshold. In SOA, we set additionally a maximum green time of signal phase.

- *Q-learning*. In this method, normal rules for updating value function are used.

$$Q_{t+1}(\boldsymbol{s},\boldsymbol{a}) = Q_t(\boldsymbol{s},\boldsymbol{a}) + \eta_t(r_t(\boldsymbol{s},\boldsymbol{a}) + \gamma \min_{\boldsymbol{a}'\in A} Q_t(\boldsymbol{s}',\boldsymbol{a}') - Q_t(\boldsymbol{s},\boldsymbol{a})), \tag{5.4}$$

  where η_t is learning rate, $\boldsymbol{s}'$ is the transfered state from $\boldsymbol{s}$ taken action $\boldsymbol{a}$. Each intersection is an independent agent. To avoid computation complicity of full-state representations, the state of queue length at any time is simplified as traffic low, medium, or high. Thus, the state-action space of the network is reduced. Threshold is differently set depending on the traffic demands.

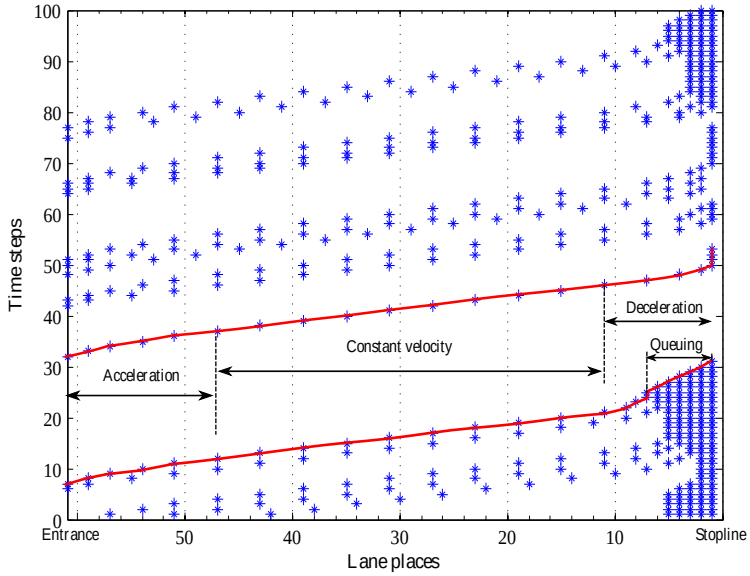


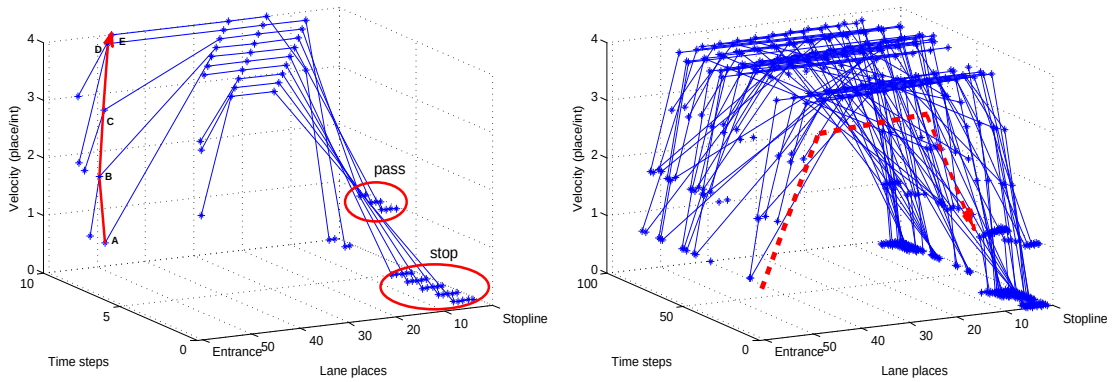
Figure 5.11: Behavior of vehicle movements on lane (60 places) during samples of 100 steps

5.3.2 Vehicle-following simulation

Before giving performance of the proposed algorithm, some properties of the traffic network loading based on the vehicle-following model are presented. We pick up one lane at the network for analysis.

In Fig. 5.11, the profile of vehicle movements on the lane in each time step is shown. We choose two vehicle movements represented in red line. According to the vehicle positions from the lane entrance to the stop line, behaviors of vehicle moving, including acceleration, constant velocity, and deceleration, could be clearly shown. Moreover, vehicles are queuing behind the stop line when receiving red signal.

In order to see the changes of vehicle velocity, we transform the figure 2-D (X-axis for lane spaces and Y-axis for time steps) to a figure 3-D with velocity illustrated in Z-axis, as shown in Fig. 5.12. For simplicity, in Fig. 5.12(a), we just choose samples of 0-10 steps to show the evolution of vehicle velocity, which is changing from 1 to $v_{\max} = 4$ place/int when vehicles are entering the lane until to stop before the stop line meeting red signal or directly pass when having right-of-way in green signal. In Fig. 5.12(b), samples of 0-100 steps are depicted. The whole tendency of velocities can be expressed by the dash red line. Obviously, three basic characteristics of velocity are implicated in the simulation.

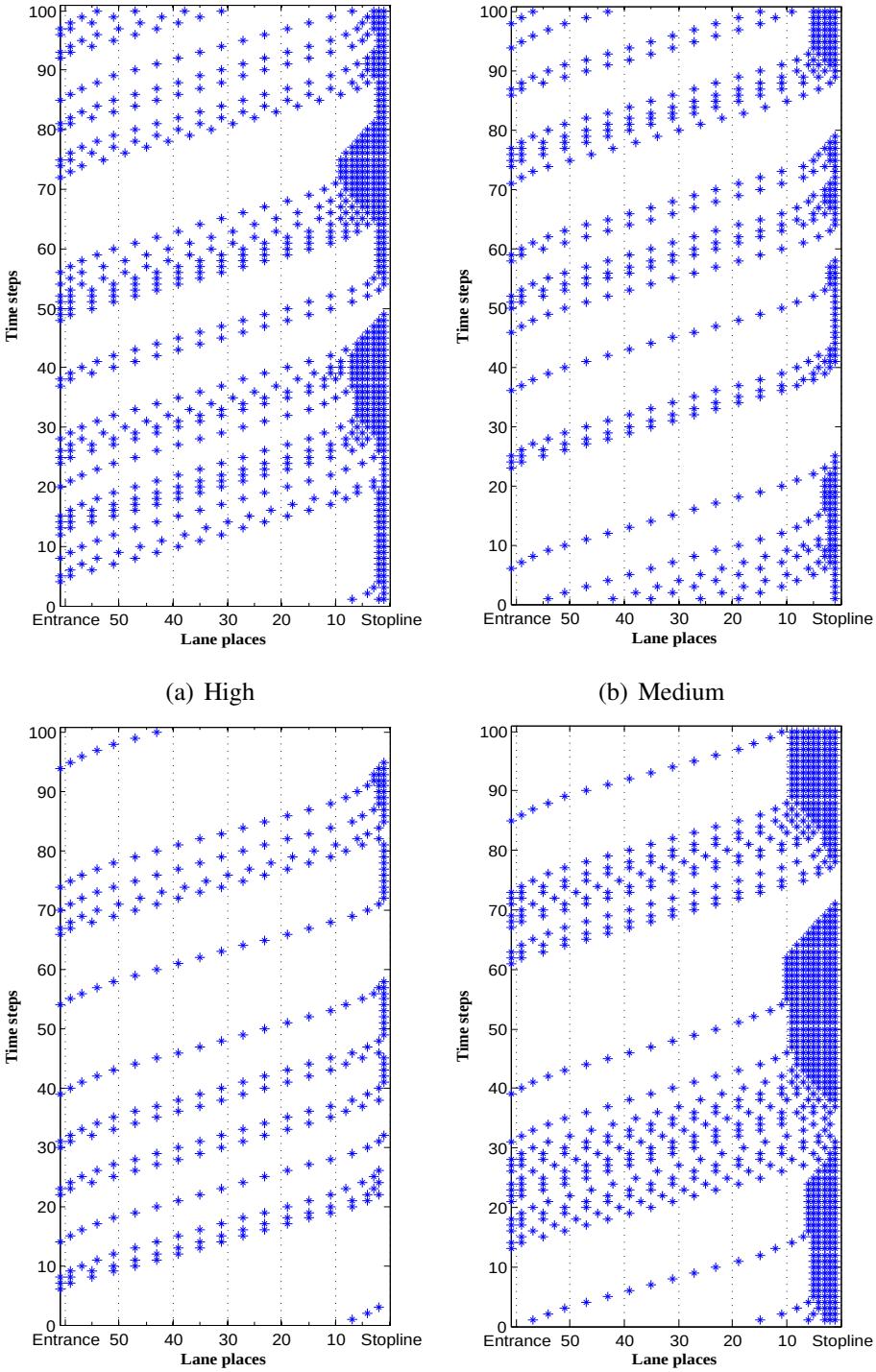


(a) 0-10 steps (b) 0-100 steps

Figure 5.12: Examples of the relationship between vehicle position (X-axis) and velocity (Z-axis) in 3-Dimension

Actually, different traffic signal control methods and traffic demands make different vehicle positions and velocity distributions, which in some degree can illustrate traffic flow intensity and queue length on lane. For example, as shown in Fig. 5.13, we do some comparisons in different cases and focus on the changes of the queue lengths in 2-D. In Fig. 5.13(a), 5.13(b), and 5.13(c), the queue lengths appear some differences by using ADP approach in high, medium, and low traffic demand, respectively. Obviously, the higher traffic demand makes more vehicles moving on the lane and more vehicles in the queue. On the other hand, the vehicle platoons generated by green split really appear on the lane. This characteristic is clearly shown in Fig. 5.13(d), in which FC method is used. The wide brands representing big platoons come from the released vehicles in upstream intersection. The appearance of single vehicle is also possible because of the right-turn proportion only being 14% in our case.

Totally speaking, the behaviors of vehicle movements are reasonable and imitate to realistic environment by the simulation, which supports us to implement some control algorithms for traffic signal control at each intersection of the network. In the next section, we will focus on comparisons and analysis of control performance.



(a) High

(b) Medium

(c) Low

(d) Medium

Figure 5.13: Evolutions of vehicle position on lane, ADP for (a), (b), (c), and FC for (d)

5.3.3 Comparisons and analysis

In this section, we will discuss the related algorithms for independent and coordinated traffic network control. We will emphasize on the coordinated one, as the proposed concept of tunable system states is employed for traffic network coordination.

5.3.3.1 Independent traffic network

The independent traffic network control is that for each intersection, the controller is independently to operate the signal plan, according to the local traffic information. It is easy to implement when we make a distributed traffic system and only do (near) optimization of the isolated intersection. So, the implementation of independent traffic network control is an extension of isolated intersection to multi-intersection. At each intersection, we also use the queue length and signal indication on lane as system state and define the cost function by using total queue lengths, which is the same definition for the isolated intersection control in (2.16).

Let us recall the definition of tunable system state in Section 2.4 of Chapter 2. We know that the tunable state parameter e ($0 \leq e \leq 1$) weights the state components of queue length and total number of vehicles on lane, underlying signal conditions of a local intersection and neighboring one. Thus, the system state is affected by joint actions of adjacent intersections. We view the tunable state parameter e as the factor of control coordination. It is easy to find that when $e = 1$ the coordination problem is transferred to the independent one. Mentioned that the independent traffic network control algorithm adopts the extension of the isolated intersection Algorithm 3 rather than the coordinated one in Algorithm 4 with $e = 1$. Because the later costs more time in exhausted search (or max-plus method) for calculation of joint actions, whereas simulation results are the same. In order to avoid repetition, performance of the independent traffic network control will be compared and analyzed in the following part, which is about discussion on coordinated traffic network control.

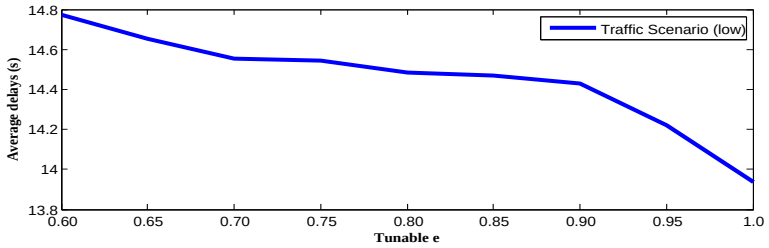
5.3.3.2 Coordinated traffic network

In this part, simulations of the coordinated traffic network with tunable state will be implemented by ADP_RLS-TD(λ) and some comparing methods. Signal control modes of FPS, VPS, and APS are also taken into account.

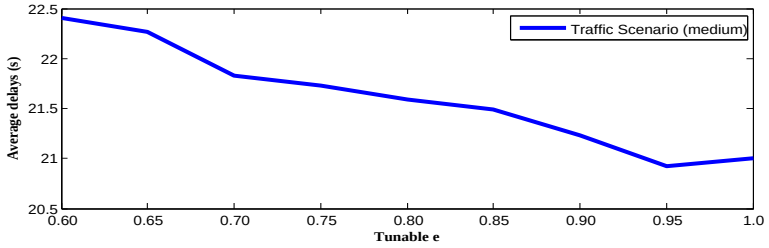
Before we do performance comparisons of the algorithms, the influence of tunable parameters will be analyzed. In Fig. 5.14, performances of average delay by using various tunable parameter e are presented. Some interesting results emerge in three traffic scenarios, namely low, medium, and high demand, where the best performance appears in $e = 1$, $e = 0.95$, and $e = 0.90$, respectively. When e is 1, it is an independent control that decision making is only based on the queuing information at local intersection. However, it is found that relatively efficient e values are almost in a range of 0.6 to 1 and the lower delays are near to $e = 1$. It could be explained in two aspects. One is that the number of vehicles on lane has already contained a part of queue length. The other one is that, more importantly, ADP can make a fine planning by small steps so that signal phase can be switched adaptively and frequently. Therefore, the state of queue length approaching the intersection has much more influence on decision making than the state of vehicles on the whole lane. In low traffic demand, the independently distributed control by using ADP_RLS-TD(λ) is enough for the traffic network system. But in high demand, the local information is not enough and the two state components, i.e., the local queue length and the total number of vehicles on lane are both needed.

In Tab. 5.8, we list average delays and improvements by using different methods. It is clearly shown that ADP_RLS-TD(λ) ($\lambda = 0$) method outperforms all others in traffic scenarios of low, medium, and high demands, especially in APS mode that has large delay reductions. Q-learning in VPS also performs well. Although, it is a coarse planning and learning approach in the condition of all state representations replaced by low, medium, or high level of vehicle queue length with thresholds. With parameters settings, SOA is a common adaptive self-organization method (actually it operates in VPS mode) but still owning large improvements compared by FC. It could be also found that large differences are effected by different phase modes in the same method, such as Q-learning and ADP_RLS-TD(λ). Notice that, in Traffic Scenario D-3, using FC and Q-learning in FPS causes vehicle spillback beyond the end of the link. That's why the delays about these two cases are empty in the table. In this way, Q-learning in FPS is not a good choice.

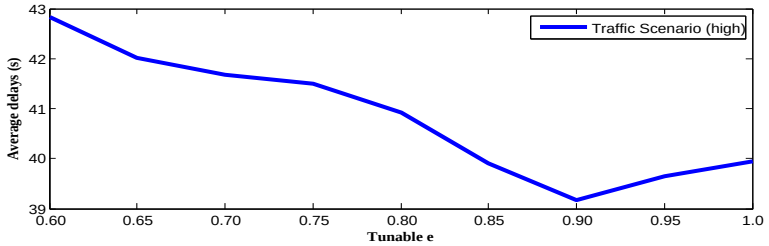
From the discussion of tunable parameters, we know that in low traffic demand the independent network control ($e = 1$) is simple to be implemented with its performance guaranteed. Actually, when an adaptive control mode (such as APS) and a planning step enough small (such as 2-s solution or finer solution) are satisfied, it is also convenient to use an independent control algorithm for traffic network, where isolated intersections are



(a) Low



(b) Medium



(c) High

Figure 5.14: Comparing results by different tunable e

highly self-organized by this intelligent algorithm. Therefore, we just consider $e = 1$ for the method of ADP_RLS-TD(λ) in APS in the following experiments.

On the other hand, the performance of queue length will be analyzed. It is easy to know that queue length is the cause of different average delays.

In Fig. 5.15, during the simulation period, queue lengths (by the mean of 100 sample steps) of all signal phase combinations at the central intersection I_3 are presented. Traffic Scenario D-2 with medium traffic demand is tested. The normal four-phase flow combinations, i.e., $G_1 = (1, 5), G_2 = (2, 6), G_3 = (3, 7), G_4 = (4, 8)$, are abstracted by using the maximum length on the lane of the combination. Obviously, using FC method makes some longest queue lengths, as shown in Fig. 5.15(a). Whereas, by using SOA, Q-learning, and ADP_RLS-TD(λ), we observe that the performances are better than FC. Comparing these three methods, the stability and fairness of queue lengths in phases can be analyzed. By using SOA, in Fig. 5.15(b), queue lengths are stable but unfair with differences appearing between phases. By using Q-learning, in Fig. 5.15(c), queue lengths

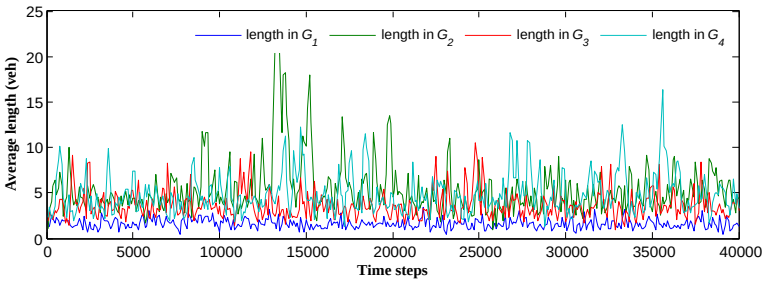
Table 5.8: Methods comparing on average delay (s) and improvements (%)

Traffic Scenario		D-1 (Low)		D-2 (Medium)		D-3 (High)	
FC		30.25	(0 %)	56.58	(0 %)	–	–
SOA		19.65	(-35.0 %)	28.79	(-49.1 %)	57.74	(0 %)
Q-learning	FPS	23.06	(-23.8 %)	31.14	(-45.0 %)	–	–
	VPS	17.13	(-43.4 %)	24.34	(-57.0 %)	49.75	(-13.8 %)
ADP_RLS-TD(0)	VPS[1]	13.93	(-54.0 %)	20.92	(-63.0 %)	39.17	(-32.2 %)
	APS[2]	6.60	(-78.2%)	10.80	(-80.9 %)	15.05	(-73.9 %)

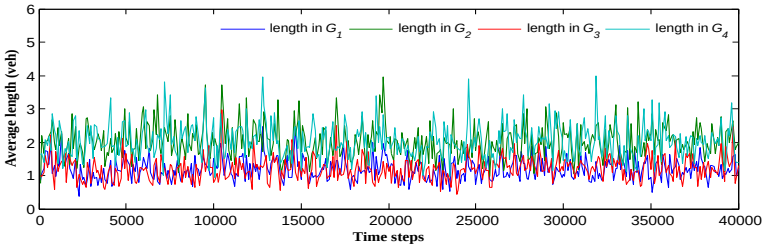
[1] In D-1, D-2 and D-3, e is set to be 1, 0.95 and 0.90, respectively.
[2] For simplicity, we choose $e = 1$ (independent) in APS mode.

are fair but unstable with long queue lengths at certain time steps. However, by using ADP_RLS-TD(λ), queue lengths are both relative stable and fair, as shown in 5.15(d) by VPS and in 5.15(e) by APS. Especially in APS mode, the queue lengths of all combinations are very low with a little variance. From the queue length performances in VPS and APS of ADP_RLS-TD(λ), we can find the reason of the lower traffic delays they have.

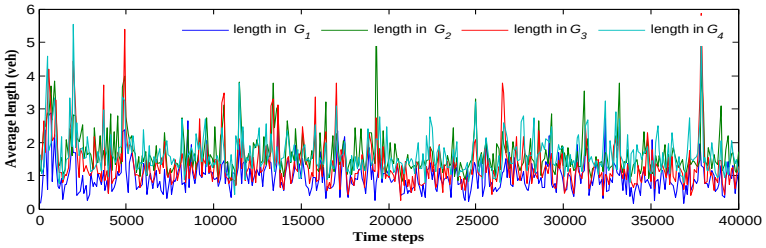
In Fig. 5.16 and 5.17, the total average queue length for the whole network is measured in medium traffic demand scenario D-2 and high traffic demand scenario D-3, respectively. Results comparisons are among FC (only in D-2), SOA, Q-learning, and ADP_RLS-TD(λ). During the simulation, the average sample results of every 100 intervals are given for analysis. Obviously, FC control has the largest fluctuations and highest queue lengths. While SOA, Q-learning, and ADP_RLS-TD(λ), which are all used for adaptive signal control, have relatively gentle variances and much lower vehicle lengths. In addition, by comparing ADP_RLS-TD(λ) with the SOA and the Q-learning both in VPS mode in Traffic Scenario D-2, as shown in Fig. 5.16, ADP_RLS-TD(λ) performs a little better than the others. While advantage of ADP_RLS-TD(λ) in VPS mode appears when high traffic demand D-3 is tested, as shown in Fig. 5.17. Obviously, the differences between these three methods are increased and the larger variations of them are also generated. More importantly, as for influences of phase control modes, ADP_RLS-TD(λ) in APS has much better performance than SOA and Q-learning in VPS, and ADP_RLS-TD(λ) in VPS. In both D-2 and D-3, the average queue lengths of this method stay in the lowest level with little variations. Totally speaking, we can see that ADP_RLS-TD(λ) can operate quite well, especially combined with APS mode for adaptive traffic signal control.



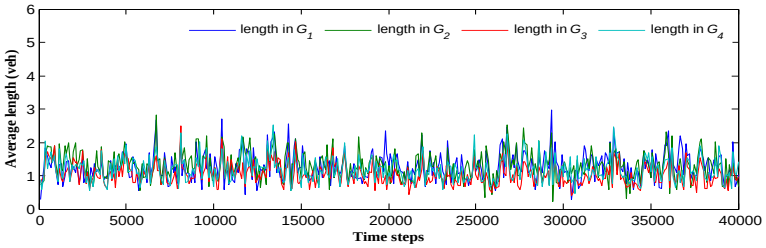
(a) FC



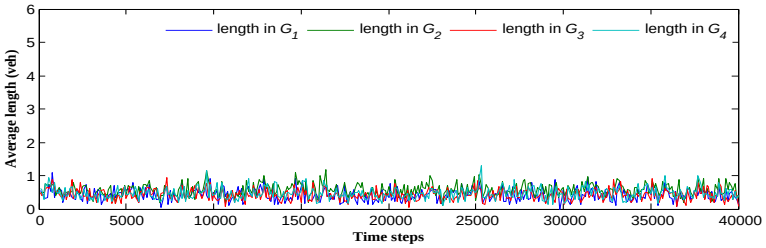
(b) SOA



(c) Q-learning (VPS)



(d) ADP_RLS-TD(λ) (VPS) (in $\lambda = 0, e = 0.95$)



(e) ADP_RLS-TD(λ) (APS) (in $\lambda = 0, e = 1$)

Figure 5.15: Comparing results of average queue length (mean of 100 sample intervals) at intersection I_3 in Traffic Scenario D-2

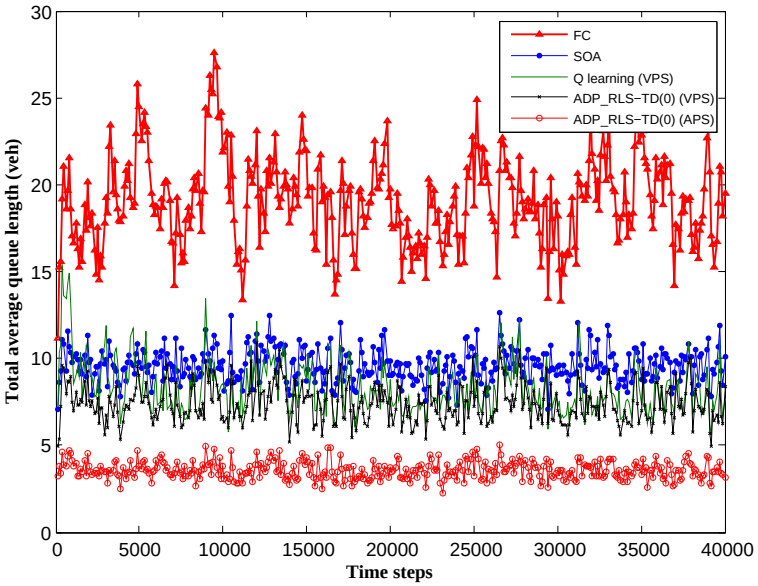


Figure 5.16: Comparing results of total average queue length at network using FC, SOA, Q-learning (VPS) and ADP_RLS-TD(λ) ($\lambda = 0$, with $e = 0.95$ in VPS and $e = 1$ in APS) methods in Traffic Scenario D-2

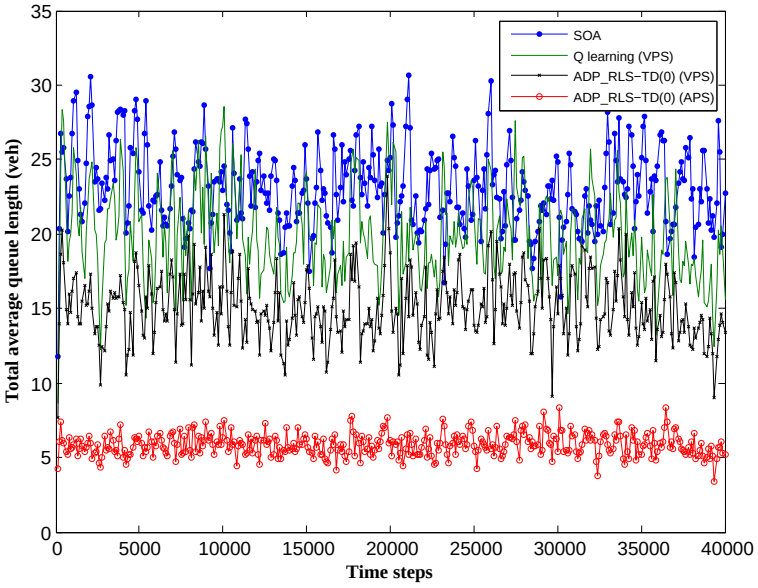


Figure 5.17: Comparing results of total average queue length at network using SOA, Q-learning (VPS) and ADP_RLS-TD(λ) ($\lambda = 0$, with $e = 0.90$ in VPS and $e = 1$ in APS) methods in Traffic Scenario D-3

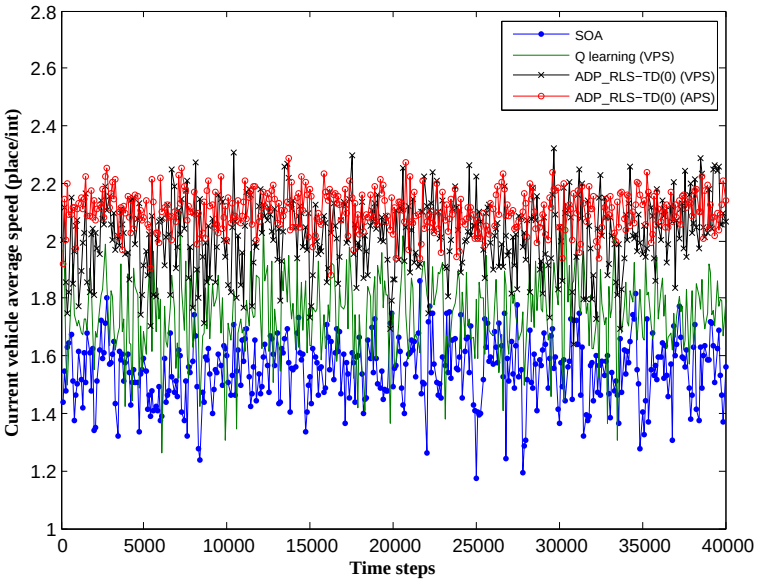


Figure 5.18: Comparing results of current vehicle average speed using SOA, Q-learning (VPS) and ADP_RLS-TD(λ) ($\lambda = 0$, with $e = 0.90$ in VPS and $e = 1$ in APS) methods in Traffic Scenario D-3

In the simulation of high traffic demand scenario D-3, we also obtain the current vehicle average speed at the whole network with all road lanes considered. The comparisons of SOA, Q-learning in VPS and ADP_RLS-TD(λ) in VPS and APS are shown in Fig. 5.18. As a whole, ADP_RLS-TD(λ) method can get the best performance with relative high average speed, which mainly ranges from 1.8 to 2.2 place/int. Especially in APS, the speeds are mostly concentrated in the range of 2.0 to 2.2 place/int. It means that the traffic network is smooth and drivers can save the travel time on the road by using this efficient control mechanism. In addition, Q-learning with main results from 1.5 to 1.9 place/int is better than SOA, which only has average speed from about 1.4 to 1.7 place/int. Interestingly, the vehicle average speed and average queue length discussed previously are negative relevances. For example, the low average queue length has the high vehicle average speed correspondingly.

Computation time

Besides the performance measures discussion above, let us look at computation efficiency of the algorithm. In Tab. 5.9, a list of run time in every simulation step by different

Table 5.9: Comparisons of run time for network

Methods	FC	SOA	Q-learning		ADP_RLS-TD(λ)		
Phase mode	Fixed	VPS	FPS	VPS	FPS	VPS	APS
Run time (10^{-3}s/step)	3.7	5.9	16.8	28.1	4.3	7.0	20.6

algorithms is given. As a whole, the cost time by implementing ADP_RLS-TD(λ) in three phase modes is much less than those of Q-learning. Although the APS mode takes 20.6×10^{-3} s/step, it can satisfy online operation enough for adaptive traffic signal control. Actually, Q-learning takes much time to update tabular values of state-action representations. It is known that APS mode (12 phases) is more complex than VPS mode (4 phases). If Q-learning is combined with APS mode, undoubtedly, it will take the most time for this control. In addition, SOA takes less time than Q-learning, but the performance is not so good as Q-learning in VPS.

5.4 Summary

By doing numerous experiments in the isolated intersection and traffic network, the investigation of ADP method using RLS-TD(λ) shows great advantages in the performance measures and computation efficiency.

In the experiments of the isolated intersection, firstly, the importance of properties of functional parameters in function approximation is mentioned. By comparing with TD(λ), parameters in recursive least-squares TD(λ) (RLS-TD(λ)) refer to the earlier stable trends and less vibrations. After all, this may lead to different performances of traffic delays and RLS-TD(λ) operates better in the experiments of numerical λ. Secondly, the comparisons of performances on traffic delay and queue length by implementing ADP_RLS-TD(λ) and other methods are discussed. Furthermore, three phase modes FPS, VPS, and APS are compared. Consequently, ADP_RLS-TD(λ) performs very well, especially in APS mode with less traffic delay and computation time. Although exact algorithm FSDP makes the largest delay reduction, the computational cost is too high and can not conveniently expand to traffic network application. In addition, ADP with RLS-TD(λ) learning is also suitable for the fine planning solution, which obtains even better results.

In the experiments for network, at first, we analyze the simulation results of vehicle-following model. This ensures the environment of algorithms implemented in the network application. Then, by comparing and analyzing performances of total average delay, average queue length, and current average vehicle speed, the proposed ADP_RLS-TD(λ) still performs very well in the traffic network control. Considering the tunable parameter for network coordination, difference appears between the independent control and coordinated control. However, the performance of the independent control in APS mode can also be guaranteed with easy implementation. In a word, advantages of the computation efficiency and facility of learning make ADP_RLS-TD(λ) enough for online adaptive operation in a traffic network.

CONCLUSION AND PERSPECTIVES

Concluding remarks

This thesis addressed an urban real-time adaptive traffic signal control problem at intersection and network, which were modeled as a distributed and dynamic system in discrete-time. The efficient and near-optimal algorithm using ADP with RLS-TD(λ) was finally confirmed after the exploration of the exact DP algorithms. This study also investigated kinds of signal phase control mechanisms called FPS, VPS, and APS, which were integrated into the algorithms and obtained different performances.

After generalities of traffic signal control system introduced and related work reviewed, the thesis was mainly completed by the following three parts.

The first part of the thesis focused on the dynamic modeling for adaptive traffic signal control problems. With discussion about phase control modes FPS, VPS, and APS, the proposed APS would be more adaptive than FPS and VPS in prediction. As for the modeling of the intersection and multi-intersection network, problems were formulated respectively by MDP and multiagent MDP with a set of characteristic definitions, including traffic state, traffic action, state transition, and reward function. Two formulations of state transition were presented. One was the stochastic state transition with probability transition function. The other one was the determined state transition which works by the way of Monte-Carol sampling. In addition, we proposed a new vehicle-following model to support the traffic network loading environment as well as the algorithm study later.

Covering the case of independent network control, the concept of tunable system state was proposed for the traffic network coordination.

The second part of the study investigated optimal solutions at isolated intersection by using two exact DP algorithms. For a simple 2-phase intersection problem, a value iteration algorithm was implemented by using the stochastic state transition model. The optimal stationary policy was finally obtained after the convergence of value iteration. With case study, this method performed well but much time was taken for convergence in offline operation. The improved algorithm referred to the second one. We developed a forward search DP algorithm called FSDP. This algorithm was applied to a deterministic state transition for a typical intersection problem and performed quite well with traffic delay reduction. But it just guaranteed the real-time operation in FPS and VPS. For the extension work of APS or more complicated case, FSDP encountered the computation burden as well. From these two algorithms, it was suggested that for simple designing of complex control problem, the determined state transition with environment noise working by Monte-Carol sampling could be easier implemented than the stochastic state transition with probability transition function, which is hard to be obtained in the model with a large state space. Whatever, these two algorithms were both limited by the computational problem in DP, which has the "curse of dimensionality". Subsequently, this problem was settled successfully in the final part.

The final part of the thesis found near-optimal algorithm both for the isolated intersection and the network study. The ADP method has a great advantage for the reduction of computational complexity by using approximation function. Rather than neural network approximator, we chose the linear function approximation, in which the functional parameters for feature function were adjusted by the recursive least-squares TD(λ) method in multi-step version. RLS-TD(λ) was tested in the experiments with better performances than TD(λ). Therefore, the ADP with RLS-TD(λ) algorithms were developed for an isolated intersection, especially in APS mode and the fine solution, and for the extension to multi-intersection including the independent and coordinated one. By doing numerous experiments at the isolated intersection and traffic network, ADP with RLS-TD(λ) indicated great advantages in performance measures and computation efficiency. It was enough to guarantee the real-time operation by using ADP with RLS-TD(λ). Moreover, different performances appeared among the FPS, VPS, and APS mode. In particular, APS is very suitable in our algorithms, and the results by implementing APS were surely to

be the best satisfied. With both considerations of safety for flows passing and efficiency of adaptivity control, its potential supports us to explore more intelligent and adaptive forms combining the vehicle control in autonomous traffic intersection management at un-signalized intersection.

Future research

This thesis may be extended in the future in the following aspects:

- The proportion of traffic merging or diverging flow in thesis is constant although it satisfies stochastic distribution. For the traffic network loading, route choice mechanism should be integrated into the system with validation of the algorithms.
- The control algorithms could be integrated and implemented robustly in other visual simulation platforms, or the developing one in our own task. It is also significant to test realistic traffic data from urban traffic.
- The thesis is to further work especially in the coordination of more intersections. It is required to find a coordinative mechanism in the larger network study in order to reduce the computational complexity in a microscopic way. Meanwhile, an appropriate algorithm to find joint action is still required.
- The study of thesis offers a highly adaptive control method at signalized intersections. It will be an interesting work to extend the study to autonomous intersection management (AIM), where the autonomous or connected vehicles are employed with speed advisory associated to cooperative adaptive cruise control system.

APPENDICES

A1. Derivation of RLS-TD(λ)

Before we do the derivation of RLS-TD(λ), the matrix inverse lemma is given as follows:

Lemme .1 *([137]) If $A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times 1}, C \in \mathbb{R}^{1\times n}$, and A is invertible, then*

$$(A+BC)^{-1} = A^{-1} - A^{-1}B(I+CA^{-1}B)^{-1}CA^{-1} \tag{5}$$

Let

$$P_t = \tilde{A}_t^{-1} \tag{6}$$

$$P_0 = \epsilon I \tag{7}$$

$$H_t = P_t z_t \tag{8}$$

According to Lemma .1 and (4.58) (4.59),

$$\begin{aligned} P_t &= \tilde{A}_t^{-1} \\ &= (\tilde{A}_{t-1} + z_t(\phi_t - \gamma\phi_{t+1})')^{-1} \\ &= P_{t-1} - P_{t-1}z_t(1 + (\phi_t - \gamma\phi_{t+1})'P_{t-1}z_t)^{-1}(\phi_t - \gamma\phi_{t+1})'P_{t-1} \end{aligned} \tag{9}$$

$$\begin{aligned} H_t &= P_t z_t \\ &= (P_{t-1} - P_{t-1}z_t(1 + (\phi_t - \gamma\phi_{t+1})' P_{t-1}z_t)^{-1}(\phi_t - \gamma\phi_{t+1})' P_{t-1})z_t \\ &= P_{t-1}z_t - \frac{P_{t-1}z_t(\phi_t - \gamma\phi_{t+1})' P_{t-1}z_t}{1 + (\phi_t - \gamma\phi_{t+1})' P_{t-1}z_t} \\ &= \frac{P_{t-1}z_t}{1 + (\phi_t - \gamma\phi_{t+1})' P_{t-1}z_t} \end{aligned} \tag{10}$$

$$\begin{aligned} \theta_t &= \tilde{A}_t^{-1}\tilde{b}_t \\ &= P_t \sum_{i=1}^{t} z_i r_i \\ &= P_t(\tilde{A}_{t-1}\tilde{A}_{t-1}^{-1}\tilde{b}_{t-1} + z_t r_t) \\ &= P_t((\tilde{A}_t - z_t(\phi_t - \gamma\phi_{t+1})')\theta_{t-1} + z_t r_t) \\ &= P_t P_t^{-1}\theta_{t-1} + P_t(z_t r_t - z_t(\phi_t - \gamma\phi_{t+1})'\theta_{t-1}) \\ &= \theta_{t-1} + P_t z_t(r_t - (\phi_t - \gamma\phi_{t+1})'\theta_{t-1}) \end{aligned} \tag{11}$$

Use $H_t = P_t z_t$ and TD error $\tilde{\delta}_t = r_t - (\phi_t - \gamma\phi_{t+1})'\theta_{t-1}$, thus

$$\begin{aligned} \theta_t &= \theta_{t-1} + H_t(r_t - (\phi_t - \gamma\phi_{t+1})'\theta_{t-1}) \\ &= \theta_{t-1} + \frac{P_{t-1}}{1 + (\phi_t - \gamma\phi_{t+1})' P_{t-1}z_t}\tilde{\delta}_t z_t \end{aligned} \tag{12}$$

A2. The Proof of RLS-TD(λ) Convergence

Firstly, the formal statements are the following.

Assumption .1 *The Markov chain $\{s_t\}$ is ergodic with transition probability matrix P, and there is an unique distribution π that satisfies*

$$\pi' P = \pi' \tag{13}$$

with $\pi(s) > 0$ for all $s \in S$ and π is a finite or infinite vector, depending on the cardinality of S.

Assumption .2 *Let $E_0(\cdot)$ stand for expectation with respect to distribution of π. Transition rewards r_t satisfy*

$$E_0(r_t^2) < \infty. \tag{14}$$

Assumption .3 *The matrix* $\Phi = (\phi^1, \phi^2, ..., \phi^K) \in \mathbb{R}^{|S| \times K}$ *has full column rank, that is, the basis function* $\phi^k (k = 1, 2, ..., K)$ *are linear independent.*

Assumption .4 *For every* k*, the basis function* ϕ^k *satisfies*

$$E_0[(\phi^k(s_t))^2] < \infty. \tag{15}$$

Assumption .5 *The matrix* $[P_0^{-1} + \frac{1}{t}\sum_{i=1}^{t} A(X_i)]$ *is non-singular for all* $t > 0$.

Theorem .1 *[132] For a Markov chain which satisfied Assumption 1-5, the asymptotic estimate found by RLS-TD(*λ*) converges, with probability 1, to* θ^* *determined by*

$$E_0[A(X_t)]\theta^* - E_0[b(X_t)] = 0 \tag{16}$$

Proof: According to [98], $E_0[A(X_t)]$ and $E_0[b(X_t)]$ are well defined and finite. Furthermore, $E_0[A(X_t)]$ is negative definite, thus it is invertible.

Consider the condition of $t = 0$. According to (11), we have

$$\begin{aligned}
\theta &= (\tilde{A}_0 + \sum_{i=1}^{t} A(X_i))^{-1}(\tilde{A}_0\theta_0 + \sum_{i=1}^{t} b(X_i)) \\
&= (P_0^{-1} + \sum_{i=1}^{t} A(X_i))^{-1}(P_0^{-1}\theta_0 + \sum_{i=1}^{t} b(X_i)) \\
&= (\frac{1}{t}P_0^{-1} + \frac{1}{t}\sum_{i=1}^{t} A(X_i))^{-1}(\frac{1}{t}P_0^{-1}\theta_0 + \frac{1}{t}\sum_{i=1}^{t} b(X_i))
\end{aligned} \tag{17}$$

Since

$$E_0[A(X_t)] = \lim_{t\to\infty} \frac{1}{t}\sum_{i=1}^{t} A(X_i) \tag{18}$$

$$E_0[b(X_t)] = \lim_{t\to\infty} \frac{1}{t}\sum_{i=1}^{t} b(X_i) \tag{19}$$

and $E_0[A(X_t)]$ is invertible,

$$\lim_{t\to\infty} \theta = E_0^{-1}[A(X_t)]E_0[b(X_t)] = \theta^* \tag{20}$$

Thus, θ converges to θ^* with probability 1. ■

A3. Derivation of Multi-step Planning of RLS-TD(λ)

In multi-step planning, according to (4.66), the objective function (4.50) can be rewritten as:

$$O(\theta_t) = \frac{1}{t}\sum_{i=1}^{t}\left(\sum_{k=i}^{i+t_M-1}\gamma^{k-i}r_k - (\phi_i - \gamma^{t_M}\phi_{i+t_M})'\theta_t\right)^2. \tag{21}$$

It is easy to see that r_i and $(\phi_i - \gamma\phi_{i+1})'$ in (4.50) are substituted by $\sum_{k=i}^{i+t_M-1}\gamma^{k-i}r_k$ and $(\phi_i - \gamma^{t_M}\phi_{i+t_M})'$, respectively. According to related theory in [133] and [99], we can rewrite (4.51) as following by using ϕ_t as the instrumental variable in LS-TD(0). That is,

$$\theta_t = \left(\frac{1}{t}\sum_{i=1}^{t}\phi_i(\phi_i - \gamma^{t_M}\phi_{i+t_M})'\right)^{-1}\left(\frac{1}{t}\sum_{i=1}^{t}\phi_i\sum_{k=i}^{i+t_M-1}\gamma^{k-i}r_k\right) \tag{22}$$

In LS-TD(λ), θ_t can be estimated as

$$\theta_t = \left(\frac{1}{t}\sum_{i=1}^{t}z_i(\phi_i - \gamma^{t_M}\phi_{i+t_M})'\right)^{-1}\left(\frac{1}{t}\sum_{i=1}^{t}z_i\sum_{k=i}^{i+t_M-1}\gamma^{k-i}r_k\right) \tag{23}$$

$$= \left(\sum_{i=1}^{t}z_i(\phi_i - \gamma^{t_M}\phi_{i+t_M})'\right)^{-1}\left(\sum_{i=1}^{t}z_i\sum_{k=i}^{i+t_M-1}\gamma^{k-i}r_k\right) \tag{24}$$

where using the eligibility vector z_i in (4.38) substitutes the ϕ_i.

According to matrix inverse Lemma .1 and derivation of RLS-TD(λ) in Appendix A1, the parameter vector θ_t in (4.68) of multi-step planning of RLS-TD(λ) can be guaranteed.

BIBLIOGRAPHY

[1] M. Papageorgiou, C. Diakaki, V. Dinopoulou, A. Kotsialos, and Y. Wang, "Review of road traffic control strategies," *Proceedings of the IEEE*, vol. 91, no. 12, pp. 2043–2067, 2003.

[2] X. Zheng and W. Recker, "An adaptive control algorithm for traffic-actuated signals," *Transportation Research Part C: Emerging Technologies*, vol. 30, pp. 93–115, 2013.

[3] K. Dresner and P. Stone, "A multiagent approach to autonomous intersection management," *Journal of artificial intelligence research*, pp. 591–656, 2008.

[4] S. I. Guler, M. Menendez, and L. Meier, "Using connected vehicle technology to improve the efficiency of intersections," *Transportation Research Part C: Emerging Technologies*, vol. 46, pp. 121–131, 2014.

[5] A. L. Bazzan, "Opportunities for multiagent systems and multiagent reinforcement learning in traffic control," *Autonomous Agents and Multi-Agent Systems*, vol. 18, no. 3, pp. 342–375, 2009.

[6] S. El-Tantawy and B. Abdulhai, "Multi-agent reinforcement learning for integrated network of adaptive traffic signal controllers (MARLIN-ATSC)," in *Proceedings of IEEE Conference on Intelligent Transportation Systems*, 2012, pp. 319–326.

[7] P. Koonce, L. Rodegerdts, K. Lee, S. Quayle, S. Beaird, C. Braud, J. Bonneson, P. Tarnoff, and T. Urbanik, "Traffic signal timing manual," Tech. Rep., 2008.

[8] Transportation Research Board, "Highway capacity manual," *National Research Council,Washington, DC*, 2000.

[9] D. I. Robertson, "TRANSYT method for area traffic control," *Traffic Engineering & Control*, vol. 11, no. 6, 1969.

[10] R. Vincent and C. Young, "Self-optimising traffic signal control using microprocessors. the trrl mova strategy for isolated intersections," *Traffic engineering & control*, vol. 27, no. 7-8, pp. 385–387, 1986.

[11] P. Lowrie, "The Sydney coordinated adaptive traffic system-principles, methodology, algorithms," in *International Conference on Road Traffic Signalling, London*, no. 207, 1982.

[12] P. Hunt, D. Robertson, R. Bretherton, and M. Royle, "The SCOOT on-line traffic signal optimisation technique," *Traffic Engineering & Control*, vol. 23, no. 4, 1982.

[13] N. H. Gartner, "OPAC: A demand-responsive strategy for traffic signal control," *Transportation Research Record*, no. 906, pp. 75–81, 1983.

[14] V. Mauro and C. Di Taranto, "UTOPIA," in *Proceedings of IFAC/IFORS Conference on Control, Computers and Communications in Transport*, 1989.

[15] D. Robertson and R. Bretherton, "Optimum control of an intersection for any known sequence of vehicle arrivals," in *Proceedings of the 2nd IFAC/IFIP/IFORS Symposium on Traffic Control and Transportation Systems*, 1974.

[16] J.-J. Henry, J.-L. Farges, and J. Tuffal, "The PRODYN real time traffic algorithm," in *IFACIFIP-IFORS Conference on control in transportation systems*, 1984.

[17] K. L. Head, P. B. Mirchandani, and D. Sheppard, "Hierarchical framework for real-time traffic control," *Transportation Research Record*, no. 1360, pp. 82–88, 1992.

[18] P. Mirchandani and L. Head, "A real-time traffic signal control system: architecture, algorithms, and analysis," *Transportation Research Part C: Emerging Technologies*, vol. 9, no. 6, pp. 415–432, 2001.

[19] C. Cai, C. K. Wong, and B. G. Heydecker, "Adaptive traffic signal control using approximate dynamic programming," *Transportation Research Part C: Emerging Technologies*, vol. 17, no. 5, pp. 456–474, 2009.

[20] J. Wu, A. Abbas-Turki, and A. El Moudni, "Cooperative driving: an ant colony system for autonomous intersection management," *Applied Intelligence*, vol. 37, no. 2, pp. 207–222, 2012.

[21] V. Milanés, J. Villagrá, J. Godoy, J. Simó, J. Pérez, and E. Onieva, "An intelligent v2i-based traffic management system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 1, pp. 49–58, 2012.

[22] J. Lee and B. Park, "Development and evaluation of a cooperative vehicle intersection control algorithm under the connected vehicles environment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 1, pp. 81–90, 2012.

[23] K. Dresner and P. Stone, "Multiagent traffic management: A reservation-based intersection control mechanism," in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, vol. 2, 2004, pp. 530–537.

[24] L. Li, D. Wen, and D. Yao, "A survey of traffic control with vehicular communications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 1, pp. 425 – 432, 2014.

[25] B. Van Arem, C. J. Van Driel, and R. Visser, "The impact of cooperative adaptive cruise control on traffic-flow characteristics," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 4, pp. 429–436, 2006.

[26] H. K. Lo, "A novel traffic signal control formulation," *Transportation Research Part A: Policy and Practice*, vol. 33, no. 6, pp. 433–448, 1999.

[27] H. K. Lo, E. Chang, and Y. C. Chan, "Dynamic network traffic control," *Transportation Research Part A: Policy and Practice*, vol. 35, no. 8, pp. 721–744, 2001.

[28] K. Aboudolas, M. Papageorgiou, A. Kouvelas, and E. Kosmatopoulos, "A rolling-horizon quadratic-programming approach to the signal control problem in large-

scale congested urban road networks," *Transportation Research Part C: Emerging Technologies*, vol. 18, no. 5, pp. 680–694, 2010.

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

[30] X.-H. Yu and W. W. Recker, "Stochastic adaptive control model for traffic signal systems," *Transportation Research Part C: Emerging Technologies*, vol. 14, no. 4, pp. 263–282, 2006.

[31] R. Haijema and J. van der Wal, "An mdp decomposition approach for traffic control at isolated signalized intersections," *Probability in the Engineering and Informational Sciences*, vol. 22, no. 04, pp. 587–602, 2008.

[32] M. Dell'Orco, "A dynamic network loading model for mesosimulation in transportation systems," *European Journal of Operational Research*, vol. 175, no. 3, pp. 1447–1454, 2006.

[33] H. B. Celikoglu and M. Dell'Orco, "Mesoscopic simulation of a dynamic link loading process," *Transportation Research Part C: Emerging Technologies*, vol. 15, no. 5, pp. 329–344, 2007.

[34] M. J. Lighthill and G. B. Whitham, "On kinematic waves I: flood movement in long rivers II: a theory of traffic flow on long crowded roads," *the Royal Society of London A 229*, no. 281-345, 1955.

[35] P. I. Richards, "Shock waves on the highway," *Operations research*, vol. 4, no. 1, pp. 42–51, 1956.

[36] C. F. Daganzo, "The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory," *Transportation Research Part B: Methodological*, vol. 28, no. 4, pp. 269–287, 1994.

[37] A. Sumalee, R. Zhong, T. Pan, and W. Szeto, "Stochastic cell transmission model (SCTM): A stochastic dynamic traffic model for traffic state surveillance and assignment," *Transportation Research Part B: Methodological*, vol. 45, no. 3, pp. 507–533, 2011.

[38] H. Abouaissa, M. Fliess, and C. Join, "Fast parametric estimation for macroscopic traffic flow model," in *17th IFAC World Congress*, 2008.

[39] "PARAMICS." [Online]. Available: www.paramics-online.com

[40] "MITSIM." [Online]. Available: https://its.mit.edu/software

[41] "VISSIM." [Online]. Available: http://vision-traffic.ptvgroup.com/en-uk/products/ptv-vissim/

[42] K. Nagel and M. Schreckenberg, "A cellular automaton model for freeway traffic," *Journal de physique I*, vol. 2, no. 12, pp. 2221–2229, 1992.

[43] O. K. Tonguz, W. Viriyasitavat, and F. Bai, "Modeling urban traffic: a cellular automata approach," *IEEE Communications Magazine*, vol. 47, no. 5, pp. 142–150, 2009.

[44] J. Esser and M. Schreckenberg, "Microscopic simulation of urban traffic based on cellular automata," *International Journal of Modern Physics C*, vol. 8, no. 05, pp. 1025–1036, 1997.

[45] S. Maerivoet and B. De Moor, "Cellular automata models of road traffic," *Physics Reports*, vol. 419, no. 1, pp. 1–64, 2005.

[46] M. Florian, M. Mahut, and N. Tremblay, "Application of a simulation-based dynamic traffic assignment model," *European Journal of Operational Research*, vol. 189, no. 3, pp. 1381–1392, 2008.

[47] M. Wiering, "Multi-agent reinforcement learning for traffic light control," in *International Conference on Machine Learning (ICML)*, 2000, pp. 1151–1158.

[48] M. Wiering, J. Vreeken, J. Van Veenen, and A. Koopman, "Simulation and optimization of traffic in a city," in *IEEE Intelligent Vehicles Symposium*, 2004, pp. 453–458.

[49] R. E. Bellman, *Dynamic Programming*. Princeton University Press, Princeton, 1957.

[50] R. J. Dakin, "A tree-search algorithm for mixed integer programming problems," *The Computer Journal*, vol. 8, no. 3, pp. 250–255, 1965.

[51] T. H. Heung, T. K. Ho, and Y. F. Fung, "Coordinated road-junction traffic control by dynamic programming," *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 3, pp. 341–350, 2005.

[52] J. Wu, A. Abbas-Turki, and A. El Moudni, "Discrete methods for urban intersection traffic controlling," in *IEEE 69th Vehicular Technology Conference (VTC Spring)*. IEEE, 2009, pp. 1–5.

[53] D. Teodorović, V. Varadarajan, J. Popović, M. R. Chinnaswamy, and S. Ramaraj, "Dynamic programming-neural network real-time traffic adaptive signal control algorithm," *Annals of Operations Research*, vol. 143, no. 1, pp. 123–131, 2006.

[54] F. Yan, M. Dridi, and A. El Moudni, "A scheduling approach for autonomous vehicle sequencing problem at multi-intersections," *International Journal of Operational Research*, vol. 9, no. 1, pp. 57–68, 2011.

[55] F. Yan, M. Dridi, and A. El-Moudni, "New vehicle sequencing algorithms with vehicular infrastructure integration for an isolated intersection," *Telecommunication Systems*, vol. 50, no. 4, pp. 325–337, 2012.

[56] B. Yin, M. Dridi, and A. El Moudni, "Forward search algorithm based on dynamic programming for real-time adaptive traffic signal control," *IET Intelligent Transportation Systems*, vol. 9, no. 7, pp. 754–764, 2015.

[57] Y. Zhang and Y. Xie, "Traffic signal timing and optimization," *Artificial Intelligence Applications to Critical Transportation Issues*, p. 11, 2012.

[58] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 1998.

[59] B. Park, C. Messer, and T. Urbanik, "Traffic signal optimization program for oversaturated conditions: genetic algorithm approach," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1683, pp. 133–142, 1999.

[60] B. Park, C. Messer, and T. Urbanik II, "Enhanced genetic algorithm for signal-timing optimization of oversaturated intersections," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1727, pp. 32–41, 2000.

[61] H. Ceylan and M. G. Bell, "Traffic signal timing optimisation based on genetic algorithm approach, including drivers' routing," *Transportation Research Part B: Methodological*, vol. 38, no. 4, pp. 329–342, 2004.

[62] K. B. Kesur, "Advances in genetic algorithm optimization of traffic signals," *Journal of Transportation Engineering*, 2009.

[63] J. Lee, B. Abdulhai, A. Shalaby, and E.-H. Chung, "Real-time optimization for adaptive traffic signal control using genetic algorithms," *Journal of Intelligent Transportation Systems*, vol. 9, no. 3, pp. 111–122, 2005.

[64] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *IEEE International Conference on Neural Networks*, vol. 4. IEEE, 1995, pp. 1942–1948.

[65] D. Teodorović, "Swarm intelligence systems for transportation engineering: Principles and applications," *Transportation Research Part C: Emerging Technologies*, vol. 16, no. 6, pp. 651–667, 2008.

[66] Y. Wei, Q. Shao, Y. Han, and B. Fan, "Intersection signal control approach based on pso and simulation," in *Second InternationalConference on Genetic and Evolutionary Computing*. IEEE, 2008, pp. 277–280.

[67] J. García-Nieto, E. Alba, and A. C. Olivera, "Swarm intelligence for traffic light scheduling: Application to real urban areas," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 2, pp. 274–283, 2012.

[68] J. Garcia-Nieto, A. C. Olivera, and E. Alba, "Optimal cycle program of traffic lights with particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 6, pp. 823–839, 2013.

[69] R. Putha, L. Quadrifoglio, and E. Zechman, "Comparing ant colony optimization and genetic algorithm approaches for solving traffic signal coordination under oversaturation conditions," *Computer-Aided Civil and Infrastructure Engineering*, vol. 27, no. 1, pp. 14–28, 2012.

[70] D. McKenney and T. White, "Distributed and adaptive traffic signal control within a realistic traffic simulation," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 1, pp. 574–583, 2013.

[71] S. Chiu and S. Chand, "Adaptive traffic signal control using fuzzy logic," in *Second IEEE International Conference on Fuzzy Systems*. IEEE, 1993, pp. 1371–1376.

[72] J. Niittymäki and M. Pursula, "Signal control using fuzzy logic," *Fuzzy sets and systems*, vol. 116, no. 1, pp. 11–22, 2000.

[73] M. B. Trabia, M. S. Kaseko, and M. Ande, "A two-stage fuzzy logic controller for traffic signals," *Transportation Research Part C: Emerging Technologies*, vol. 7, no. 6, pp. 353–367, 1999.

[74] L. Zhang, H. Li, P. D. Prevedouros, *et al.*, "Signal control for oversaturated intersections using fuzzy logic," in *Transportation Research Board Annual Meeting, Washington DC, USA*, 2005.

[75] Y. S. Murat and E. Gedizlioglu, "A fuzzy logic multi-phased signal control model for isolated junctions," *Transportation Research Part C: Emerging Technologies*, vol. 13, no. 1, pp. 19–36, 2005.

[76] M. C. Choy, D. Srinivasan, and R. L. Cheu, "Cooperative, hybrid agent architecture for real-time traffic signal control," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 33, no. 5, pp. 597–607, 2003.

[77] J.-H. Lee and H. Lee-Kwang, "Distributed and cooperative fuzzy controllers for traffic intersections group," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 29, no. 2, pp. 263–271, 1999.

[78] B. P. Gokulan and D. Srinivasan, "Distributed geometric fuzzy multiagent urban traffic signal control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 3, pp. 714–727, 2010.

[79] D. Zhao, Y. Dai, and Z. Zhang, "Computational intelligence in urban traffic signal control: A survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 42, no. 4, pp. 485–494, 2012.

[80] M. C. Choy, D. Srinivasan, and R. L. Cheu, "Neural networks for continuous online learning and control," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1511–1531, 2006.

[81] G. Shen and X. Kong, "Study on road network traffic coordination control technique with bus priority," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 39, no. 3, pp. 343–351, 2009.

[82] D. Srinivasan, M. C. Choy, and R. L. Cheu, "Neural networks for real-time traffic signal control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 3, pp. 261–272, 2006.

[83] D. Srinivasan and M. Choy, "Cooperative multi-agent system for coordinated traffic signal control," in *IEE Proceedings-Intelligent Transport Systems*, vol. 153, no. 1. IET, 2006, pp. 41–50.

[84] E. Bingham, "Reinforcement learning in neurofuzzy traffic signal control," *European Journal of Operational Research*, vol. 131, no. 2, pp. 232–241, 2001.

[85] L. Busoniu, R. Babuska, B. De Schutter, and D. Ernst, *Reinforcement learning and dynamic programming using function approximators*. CRC Press,USA, 2010.

[86] D. P. Bertsekas and J. N. Tsitsiklis, "Neuro-dynamic programming: an overview," in *Proceedings of IEEE 34th International Conference on Decision and Control (CDC)*, vol. 1, 1995, pp. 560–564.

[87] W. B. Powell, *Approximate Dynamic Programming: Solving the curses of dimensionality*. John Wiley & Sons,USA, 2007.

[88] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 38, no. 2, pp. 156–172, 2008.

[89] P. J. Werbos, "Approximate dynamic programming for real-time control and neural modeling," *Handbook of intelligent control: Neural, fuzzy, and adaptive approaches*, vol. 15, pp. 493–525, 1992.

[90] F.-Y. Wang, H. Zhang, and D. Liu, "Adaptive dynamic programming: an introduction," *Computational Intelligence Magazine, IEEE*, vol. 4, no. 2, pp. 39–47, 2009.

[91] J. Si and Y.-T. Wang, "Online learning control by association and reinforcement," *IEEE Transactions on Neural Networks*, vol. 12, no. 2, pp. 264–276, 2001.

[92] X. Xu, L. Zuo, and Z. H. Huang, "Reinforcement learning algorithms with function approximation: Recent advances and applications," *Information Sciences*, vol. 261, pp. 1–31, 2014.

[93] I. Arel, C. Liu, T. Urbanik, and A. G. Kohls, "Reinforcement learning-based multi-agent system for network traffic signal control," *IET Intelligent Transportation Systems*, vol. 4, no. 2, pp. 128–135, 2010.

[94] S. Box and B. Waterson, "An automated signalized junction controller that learns strategies by temporal difference reinforcement learning," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 1, pp. 652–659, 2013.

[95] L. Prashanth and S. Bhatnagar, "Reinforcement learning with function approximation for traffic signal control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 2, pp. 412–421, 2011.

[96] T. Li, D. B. Zhao, and J. Q. Yi, "Adaptive dynamic programming for multi-intersections traffic signal intelligent control," in *Proceedings of IEEE Conference on Intelligent Transportation Systems*, 2008, pp. 286–291.

[97] L. Baird and A. W. Moore, "Gradient descent for general reinforcement learning," *Advances in neural information processing systems*, pp. 968–974, 1999.

[98] J. N. Tsitsiklis and B. Van Roy, "An analysis of temporal-difference learning with function approximation," *IEEE Transactions on Automatic Control*, vol. 42, no. 5, pp. 674–690, 1997.

[99] S. J. Bradtke and A. G. Barto, "Linear least-squares algorithms for temporal difference learning," *Machine Learning*, vol. 22, no. 1-3, pp. 33–57, 1996.

[100] J. A. Boyan, "Technical update: Least-squares temporal difference learning," *Machine Learning*, vol. 49, no. 2-3, pp. 233–246, 2002.

[101] D. Ormoneit and Ś. Sen, "Kernel-based reinforcement learning," *Machine learning*, vol. 49, no. 2-3, pp. 161–178, 2002.

[102] S. El-Tantawy, B. Abdulhai, and H. Abdelgawad, "Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (MARLIN-ATSC): Methodology and large-scale application on downtown Toronto," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1140–1150, 2013.

[103] ——, "Design of reinforcement learning parameters for seamless application of adaptive traffic signal control," *Journal of Intelligent Transportation Systems*, vol. 18, no. 3, pp. 227–245, 2014.

[104] M. Wiering and M. van Otterlo, *Reinforcement Learning: State-of-the-art*. Springer Science & Business Media, 2012, vol. 12.

[105] S. Richter, D. Aberdeen, and J. Yu, "Natural actor-critic for road traffic optimisation," in *Advances in neural information processing systems*, 2006, pp. 1169–1176.

[106] A. A. Sherstov and P. Stone, "Function approximation via tile coding: Automating parameter choice," in *Abstraction, Reformulation and Approximation*. Springer, 2005, pp. 194–205.

[107] T. T. Pham, T. Brys, M. E. Taylor, T. Brys, M. M. Drugan, P. Bosman, M.-D. Cock, C. Lazar, L. Demarchi, D. Steenhoff, *et al.*, "Learning coordinated traffic light control," in *Proceedings of the Adaptive and Learning Agents workshop (at AAMAS-13)*, vol. 10, 2013, pp. 1196–1201.

[108] M. Abdoos, N. Mozayani, and A. L. Bazzan, "Hierarchical control of traffic signals using q-learning with tile coding," *Applied intelligence*, vol. 40, no. 2, pp. 201–213, 2014.

[109] A. L. Bazzan, "A distributed approach for coordination of traffic signal agents," *Autonomous Agents and Multi-Agent Systems*, vol. 10, no. 1, pp. 131–164, 2005.

[110] P. Balaji, X. German, and D. Srinivasan, "Urban traffic signal control using reinforcement learning agents," *IET Intelligent Transportation Systems*, vol. 4, no. 3, pp. 177–188, 2010.

[111] A. Salkham, R. Cunningham, A. Garg, and V. Cahill, "A collaborative reinforcement learning approach to urban traffic control optimization," in *Proceedings of*

the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology-Volume 02, 2008, pp. 560–566.

[112] M. A. Khamis and W. Gomaa, "Adaptive multi-objective reinforcement learning with hybrid exploration for traffic signal control based on cooperative multi-agent framework," *Engineering Applications of Artificial Intelligence*, vol. 29, pp. 134–151, 2014.

[113] J. France and A. A. Ghorbani, "A multiagent system for optimizing urban traffic," in *International IEEE/WIC Conference on Intelligent Agent Technology*, 2003, pp. 411–414.

[114] Z. S. Yang, X. Chen, Y. S. Tang, and J. P. Sun, "Intelligent cooperation control of urban traffic networks," in *International Conference on Machine Learning and Cybernetics*, vol. 3, 2005, pp. 1482–1486.

[115] A. L. Bazzan, D. de Oliveira, and B. C. da Silva, "Learning in groups of traffic signals," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 4, pp. 560–568, 2010.

[116] L. Kuyer, S. Whiteson, B. Bakker, and N. Vlassis, "Multiagent reinforcement learning for urban traffic control using coordination graphs," in *Machine Learning and Knowledge Discovery in Databases*. Springer, 2008, pp. 656–671.

[117] J. C. Medina and R. F. Benekohal, "Traffic signal control using reinforcement learning and the max-plus algorithm as a coordinating strategy," in *Proceedings of IEEE Conference on Intelligent Transportation Systems*, 2012, pp. 596–601.

[118] J. R. Kok and N. Vlassis, "Collaborative multiagent reinforcement learning by payoff propagation," *Journal of Machine Learning Research*, vol. 7, pp. 1789–1828, 2006.

[119] P. J. Werbos, "Reinforcement learning and approximate dynamic programming (RLADP)-foundations, common misconceptions, and the challenges ahead," *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*, pp. 1–30, 2012.

[120] P. Mannion, J. Duggan, and E. Howley, "An experimental review of reinforcement learning algorithms for adaptive traffic signal control," *Autonomic Road Transport Support Systems, Autonomic Systems. Birkhauser/Springer*, 2015.

[121] *NEMAStandardsPublicationTS2-Traffic Controller Assemblies with NTCIP Requirements*, National Electrical Manufacturers Association Std., 2003.

[122] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[123] A. G. Barto, S. J. Bradtke, and S. P. Singh, "Learning to act using real-time dynamic programming," *Artificial Intelligence*, vol. 72, no. 1, pp. 81–138, 1995.

[124] B. Yin, M. Dridi, and A. El Moudni, "Traffic control model and algorithm based on decomposition of MDP," in *IEEE International Conference on Control, Decision and Information Technologies*, 2014, pp. 225–230.

[125] R. Dechter and J. Pearl, "Generalized best-first search strategies and the optimality of A*," *Journal of the ACM (JACM)*, vol. 32, no. 3, pp. 505–536, 1985.

[126] E. A. Hansen and S. Zilberstein, "LAO*: a heuristic search algorithm that finds solutions with loops," *Artificial Intelligence*, vol. 129, no. 1, pp. 35–62, 2001.

[127] N. J. Nilsson, *Artificial intelligence: a new synthesis*. Morgan Kaufmann, 1998.

[128] P. J. Werbos and X. Pang, "Generalized maze navigation: SRN critics solve what feedforward or hebbian nets cannot," in *IEEE International Conference on Systems, Man, and Cybernetics*, vol. 3, 1996, pp. 1764–1769.

[129] C. Cai, "Adaptive traffic signal control using approximate dynamic programming," Ph.D. dissertation, UNIVERSITY COLLEGE LONDON, 2009.

[130] D. V. Prokhorov, D. C. Wunsch, *et al.*, "Adaptive critic designs," *IEEE Transactions on Neural Networks*, vol. 8, no. 5, pp. 997–1007, 1997.

[131] D. Zhao, Z. Hu, Z. Xia, C. Alippi, Y. Zhu, and D. Wang, "Full-range adaptive cruise control based on supervised adaptive dynamic programming," *Neurocomputing*, vol. 125, pp. 57–67, 2014.

[132] X. Xu, H.-g. He, and D. Hu, "Efficient reinforcement learning using recursive least-squares methods," *Journal of Artificial Intelligence Research*, vol. 16, no. 1, pp. 259–292, 2002.

[133] T. Söderström and P. Stoica, "Instrumental variable methods for system identification," *Circuits, Systems and Signal Processing*, vol. 21, no. 1, pp. 1–9, 2002.

[134] N. H. Gartner, P. J. Tarnoff, and C. M. Andrews, "Evaluation of optimized policies for adaptive control strategy," *Transport Research Record*, no. 1324, pp. 105–114, 1991.

[135] F. V. Webster, "Traffic signal settings," Road Research Laboratory, London, U.K.," Road Res. Tech. Paper no. 39, 1958.

[136] S. B. Cools, C. Gershenson, and B. D'Hooghe, "Self-organizing traffic lights: A realistic simulation," in *Advances in Applied Self-organizing Systems*. Springer, 2008, pp. 41–50.

[137] L. Ljung and T. Söderström, "Theory and practice of recursive identification," 1983.

Document réalisé avec LaTeX et :
le style LaTeX pour Thèse de Doctorat créé par S. Galland — http://www.multiagent.fr/ThesisStyle
la collection de paquets `tex-upmethodology` — http://www.arakhne.org/tex-upmethodology/

Résumé :

La régulation adaptative des feux de signalisation est un problème très important. Beaucoup de chercheurs travaillent continuellement afin de résoudre les problémes liés à l'embouteillage dans les intersections urbaines. Il devient par conséquent très utile d'employer des algorithmes intelligents afin d'améliorer les performances de régulation et la qualité du service. Dans cette thèse, nous essayons d'étudier ce problème d'une part à travers une modèlisation microscopique et dynamique en temps discret, et d'autre part en explorant plusieurs approches de résoltion pour une intersection isolée ainsi que pour un réseau distribué d'intersections.

La première partie se concentre sur la modélisation dynamique des problèmes des feux de signalisation ainsi que de la charge du réseau d'intersections. Le mode de la "séquence de phase adaptative" (APS) dans un plan de feux est d'abord considéré. Quant à la modélisation du contrôle des feux aux intersections, elle est formulée grâce à un processus décisionnel de markov (MDP). En particulier, la notion de "l'état du système accordable" est alors proposée pour la coordination du réseau de trafic. En outre, un nouveau modèle de "véhicule-suiveur" est proposé pour l'environnement de trafic.

En se basant sur la modélisation proposée, les méthodes de contrôle des feux dans cette thèse comportent des algorithmes optimaux et quasi-optimaux. Deux algorithmes exacts de résolution basées sur la programmation dynamique (DP) sont alors étudiés et les résultats montrent certaines limites de cette solution DP surtout dans quelques cas complexes où l'espace d'états est assez important. En raison de l'importance du temps d'execution de l'algorithme DP et du manque d'information du modèle (notamment l'information exacte relative à l'arrivée des véhicules à l'intersection), nous avons opté pour un algorithme de programmation dynamique approximative (ADP). Enfin, un algorithme quasi-optimal utilisant l'ADP combinée à la méthode d'amélioration RLS-TD (λ) est choisi. Dans les simulations, en particulier avec l'intégration du mode de phase APS, l'algorithme proposé montre de bons résultats notamment en terme de performance et d'efficacité de calcul.

Mots-clés : Contrôle de trafic, Intersections, Processus décisionnel markovien (MDP), Programmation dynamique (DP), Programmation dynamique approximative (ADP) avec RLS-TD (λ)

Abstract:

Adaptive traffic signal control is a decision making optimization problem. People address this crucial problem constantly in order to solve the traffic congestion at urban intersections. It is very popular to use intelligent algorithms to improve control performances, such as traffic delay. In the thesis, we try to study this problem comprehensively with a microscopic and dynamic model in discrete-time, and investigate the related algorithms both for isolated intersection and distributed network control.

At first, we focus on dynamic modeling for adaptive traffic signal control and network loading problems. The proposed adaptive phase sequence (APS) mode is highlighted as one of the signal phase control mechanisms. As for the modeling of signal control at intersections, problems are fundamentally formulated by Markov decision process (MDP), especially the concept of tunable system state is proposed for the traffic network coordination. Moreover, a new vehicle-following model supports for the network loading environment.

Based on the model, signal control methods in the thesis are studied by optimal and near-optimal algorithms in turn. Two exact DP algorithms are investigated and results show some limitations of DP solution when large state space appears in complex cases. Because of the computational burden and unknown model information in dynamic programming (DP), it is suggested to use an approximate dynamic programming (ADP). Finally, the online near-optimal algorithm using ADP with RLS-TD(λ) is confirmed. In simulation experiments, especially with the integration of APS, the proposed algorithm indicates a great advantage in performance measures and computation efficiency.

Keywords: Traffic control, Intersections, Markov decision process (MDP), Dynamic programming (DP), Approximate dynamic programming (ADP) with RLS-TD(λ)

■ École doctorale SPIM - Université de Technologie Belfort-Montbéliard
F - 90010 Belfort Cedex ■ tél. +33 (0)3 84 58 31 39
■ ed-spim@univ-fcomte.fr ■ www.ed-spim.univ-fcomte.fr