DEVELOPMENT AND EVALUATION OF AN ARTERIAL ADAPTIVE TRAFFIC SIGNAL CONTROL SYSTEM USING REINFORCEMENT LEARNING

A Dissertation

by

YUANCHANG XIE

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2007

Major Subject: Civil Engineering

DEVELOPMENT AND EVALUATION OF AN ARTERIAL ADAPTIVE TRAFFIC SIGNAL CONTROL SYSTEM USING REINFORCEMENT LEARNING

A Dissertation

by

YUANCHANG XIE

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2007

Major Subject: Civil Engineering

ABSTRACT

Development and Evaluation of an Arterial Adaptive Traffic Signal Control System Using Reinforcement Learning. (December 2007)

Yuanchang Xie, B.S., Southeast University;

M.S., Southeast University

Chair of Advisory Committee: Dr. Yunlong Zhang

This dissertation develops and evaluates a new adaptive traffic signal control system for arterials. This control system is based on reinforcement learning, which is an important research area in distributed artificial intelligence and has been extensively used in many applications including real-time control.

In this dissertation, a systematic comparison between the reinforcement learning control methods and existing adaptive traffic control methods is first presented from the theoretical perspective. This comparison shows both the connections between them and the benefits of using reinforcement learning. A Neural-Fuzzy Actor-Critic Reinforcement Learning (NFACRL) method is then introduced for traffic signal control. NFACRL integrates fuzzy logic and neural networks into reinforcement learning and can better handle the curse of dimensionality and generalization problems associated with ordinary reinforcement learning methods.

This NFACRL method is first applied to isolated intersection control. Two different implementation schemes are considered. The first scheme uses a fixed phase sequence and variable cycle length, while the second one optimizes phase sequence in real time and is not constrained to the concept of cycle. Both schemes are further extended for arterial control, with each intersection being controlled by one NFACRL controller. Different strategies used for coordinating reinforcement learning controllers are reviewed, and a simple but robust method is adopted for coordinating traffic signals along the arterial.

The proposed NFACRL control system is tested at both isolated intersection and arterial levels based on VISSIM simulation. The testing is conducted under different traffic volume scenarios using real-world traffic data collected during morning, noon, and afternoon peak periods. The performance of the NFACRL control system is compared with that of the optimized pre-timed and actuated control.

Testing results based on VISSIM simulation show that the proposed NFACRL control has very promising performance. It outperforms optimized pre-timed and actuated control in most cases for both isolated intersection and arterial control. At the end of this dissertation, issues on how to further improve the NFACRL method and implement it in real world are discussed.

ACKNOWLEDGMENTS

I would like to specially thank my advisor, Dr. Yunlong Zhang, for his guidance and financial support, without which this work cannot be made possible. I would also like to thank Dr. Dominique Lord, Dr. Luca Quadrifoglio, and Dr. Sergiy Butenko for serving on my Ph.D. committee and for their suggestions that greatly improved the quality of this dissertation.

I am grateful to Dr. Ella Bingham for answering my questions and providing the source code of her work, which helped me better understand reinforcement learning. I would also like to thank Dr. Lin Zhang, Mr. Hua Wang, Dr. Wuping Xin, Mr. Srinivasa Sunkari, and Dr. Lihong Li for helps with VISSIM, Visual C++ programming, actuated traffic signal timing, and reinforcement learning.

Also, many thanks to my friends at Texas A&M and fellow students in the transportation engineering division for the friendships and happiness they brought to me.

Last but not least, I am deeply indebted to my family for their love and unconditional support.

TABLE OF CONTENTS

Page

ABSTRACT……………………………………………………………………..iii

ACKNOWLEDGMENTS…………………………………………………………..v

TABLE OF CONTENTS…………………………………………………………..vi

LIST OF FIGURES……………..………………………………………………..x

LIST OF TABLES………………………………………………………………..xiii

CHAPTER

I. INTRODUCTION 1

Problem Statement 1
Overview of the Proposed Methodology 2
Research Objectives 4
Dissertation Overview 4

II. TRAFFIC SIGNAL CONTROL BACKGROUND AND LITERATURE REVIEW 6

Introduction 6
Pre-Timed Traffic Signal Control 6
Pre-Timed Isolated Intersection Traffic Signal Control 6
Pre-Timed Arterial Traffic Signal Control 10
Actuated Traffic Signal Control 12
Actuated Signal Control at Isolated Intersection 12
Actuated Traffic Signal Control on Arterial 13
Adaptive Signal Control 14
Urban Traffic Control System (UTCS) 14
Split, Cycle and Offset Optimization Technique (SCOOT) 14
Sydney Coordinated Adaptive Traffic System (SCATS) 15
Dynamic Programmed Intersection Control (DYPIC) 16
Optimized Policies for Adaptive Control (OPAC) 19
Real-Time Hierarchical Optimized Distributed Effective System (RHODES) 22

CHAPTER Page

Urban Traffic Optimization by Integrated Automation (UTOPIA)..... 23
PRODYN 25
Adaptive Limited Look-ahead Optimization of Network Signals – Decentralized (ALLONS-D)..... 27
Markov Decision Process and Dynamic Programming (MDP&DP) ..31
Traffic Control Using Fuzzy Logic and Rules 32
Summary 35

III. REINFORCEMENT LEARNING – THEORETIC BACKGROUND 38

Why Using Reinforcement Learning 38
Reinforcement Learning 39
Reinforcement Learning Problems 39
Markov Property and Markov Decision Processes 41
Dynamic Programming for MDP 44
SARSA for MDP 47
Q-Learning for MDP 51
Actor-Critic Reinforcement Learning for MDP 54
Comparison between Dynamic Programming and Reinforcement Learning 55
Review of Existing Intersection Traffic Control Studies Using Reinforcement Learning 56
Traffic Control Using SARSA 56
Adaptive Traffic Signal Control Using Q-Learning 59
Signal Control Using Actor-Critic Reinforcement Learning 60
Other Signal Control Using Reinforcement Learning 62
Problems with the Existing Methods 62
Summary 63

IV. DEVELOPMENT OF AN ARTERIAL TRAFFIC SIGNAL CONTROL SYSTEM BASED ON NEURAL FUZZY ACTOR-CRITIC REINFORCEMENT LEARNING 65

Introduction 65
Fuzzy Logic Control and Neural Networks 66
Fuzzy Logic Control 66
Neural Networks 74
Neuro-Fuzzy Actor-Critic Reinforcement Learning (NFACRL)..... 77
Introduction 77
NFACRL Structure 78

CHAPTER Page

Calculation Procedure of the NFACRL 82
Learning Procedure of the NFACRL 84
Summary of NFACRL 87
Isolated Intersection Traffic Control Based On NFACRL 87
Fixed Phase Sequence Control Based on NFACRL 87
Variable Phase Sequence Control Based on NFACRL 94
Arterial Traffic Control Based on NFACRL 96
Multiagent Reinforcement Learning 96
Arterial Traffic Control Using Multiagent NFACRL 99
Summary 100

V. EVALUATION OF THE NFACRL TRAFFIC CONTROL METHOD BASED ON MICROSCOPIC SIMULATION 102

Introduction 102
Data Description 102
Microscopic Traffic Simulation 103
Testing Design 108
Testing Procedure 108
Testing Under Different Flow Patterns 109
Network Coding 112
Algorithm Implementation 113
Performance Evaluation Criteria 115
Performance Evaluation on Isolated Intersections 115
Evaluation with Morning Data 116
Evaluation with Noon Data 122
Evaluation with Afternoon Data 128
Summary and Comparison of Performance during Morning, Noon, and Afternoon Peak Periods 134
Performance Evaluation on Arterial 137
Evaluation with Morning Data 137
Evaluation with Noon Data 140
Evaluation with Afternoon Data 143
Summary and Comparison of Performance during Morning, Noon, and Afternoon Peak Periods 146
Summary 149

VI. SUMMARY AND CONCLUSIONS 151

Contributions 151
Major Findings 153

Page

Future Research....................154

REFERENCES....................156

VITA....................166

LIST OF FIGURES

Page

Figure 1 Modeling intersection traffic signal control as agent and environment system. 3

Figure 2 Typical four-approach intersection. 7

Figure 3 An example of protected left-leading pre-timed control 7

Figure 4 Offsets and signal coordination. 11

Figure 5 Actuated signal control. 13

Figure 6 Structure of SCATS. 16

Figure 7 Illustration of the DYPIC method. 17

Figure 8 A simple illustration of the optimal sequential constrained search method. 21

Figure 9 The hierarchical control structure of VFC-OPAC. 22

Figure 10 The hierarchical control structure of the initial version of PRODYN. 25

Figure 11 Initial decision path building of ALLONS-D. 28

Figure 12 Backtracking and exploration of ALLONS-D. 30

Figure 13 Fuzzy membership functions of the current queue length. 33

Figure 14 Agent and environment in reinforcement learning. 40

Figure 15 Policy iteration of dynamic programming. 45

Figure 16 Value iteration of dynamic programming. 46

Figure 17 SARSA for MDP. 49

Figure 18 Illustration of Q-Learning algorithm. 52

Page

Figure 19 Architecture of Actor-Critic RL method (16,67). 54

Figure 20 Fuzzy membership function examples. 68

Figure 21 Example of fuzzy reasoning. 72

Figure 22 Structure of a typical fuzzy logic controller. 73

Figure 23 A typical feed-forward back-propagation neural network. 76

Figure 24 Possible control actions for a four-approach intersection. 77

Figure 25 Example of the NFACRL (83). 79

Figure 26 Training process of the NFACRL method. 86

Figure 27 Phase plan for a four-approach isolated intersection. 88

Figure 28 Layout of a typical three-approach intersection. 88

Figure 29 Phase plan for a three-approach isolated intersection. 88

Figure 30 Fuzzy membership functions for queue length state variables. 91

Figure 31 Testing arterial network. 103

Figure 32 Northbound traffic flows of the intersection of FM 2818 and Welsh Avenue. 110

Figure 33 Southbound traffic flows of the intersection of FM 2818 and Welsh Avenue. 110

Figure 34 Westbound traffic flows of the intersection of FM 2818 and Welsh Avenue. 111

Figure 35 Eastbound traffic flows of the intersection of FM 2818 and Welsh Avenue. 111

Figure 36 Total entrance traffic volumes. 112

Figure 37 Coded arterial network. 113

Page

Figure 38 DLL interface and implementation of NFACRL control schemes......... 114

Figure 39 Simulation results for four-approach intersection based on morning peak period data.. 118

Figure 40 Simulation results for three-approach intersection based on morning peak period data.. 121

Figure 41 Simulation results for four-approach intersection based on noon peak period data. 124

Figure 42 Simulation results for three-approach intersection based on noon peak period data. 127

Figure 43 Simulation results for four-approach intersection based on afternoon peak period data.. 130

Figure 44 Simulation results for three-approach intersection based on afternoon peak period data.. 133

Figure 45 Delay improvements from the NFACRL methods relative to the pre-timed control and corresponding traffic volumes for the four-approach intersection.. 135

Figure 46 Delay improvements from the NFACRL methods relative to the pre-timed control and corresponding traffic volumes for the three-approach intersection. 136

Figure 47 Simulation results for arterial based on morning peak period data. 139

Figure 48 Simulation results for arterial based on noon peak period data............. 142

Figure 49 Simulation results for arterial based on afternoon peak period data....... 145

Figure 50 Delay improvements from the NFACRL methods relative to the coordinated pre-timed control and corresponding traffic volumes for the arterial.. 147

Figure 51 Cross-street traffic.. 148

Figure 52 Cross-street turning traffic. 148

LIST OF TABLES

Page

Table 1 Learning Results of Q-Learning Method 53

Table 2 Threshold Values for Each Category 66

Table 3 Traffic Volume Data during Morning Peak Hour 104

Table 4 Traffic Volume Data during Noon Peak Hour 105

Table 5 Traffic Volume Data during Afternoon Peak Hour 106

Table 6 Simulation Results for Four-Approach Intersection Based on Morning Peak Period Data 117

Table 7 Paired-t Test for Four-Approach Intersection Based on Morning Peak Period Data 119

Table 8 Simulation Results for Three-Approach Intersection Based on Morning Peak Period Data 120

Table 9 Paired-t Test for Three-Approach Intersection Based on Morning Peak Period Data 122

Table 10 Simulation Results for Four-Approach Intersection Based on Noon Peak Period Data 123

Table 11 Paired-t Test for Four-Approach Intersection Based on Noon Peak Period Data 125

Table 12 Simulation Results for Three-Approach Intersection Based on Noon Peak Period Data 126

Table 13 Paired-t Test for Three-Approach Intersection Based on Noon Peak Period Data 128

Table 14 Simulation Results for Four-Approach Intersection Based on Afternoon Peak Period Data 129

Page

Table 15 Paired-t Test for Four-Approach Intersection Based on Afternoon Peak Period Data........ 131

Table 16 Simulation Results for Three-Approach Intersection Based on Afternoon Peak Period Data 132

Table 17 Paired-t Test for Three-Approach Intersection Based on Afternoon Peak Period Data........ 134

Table 18 Simulation Results for Arterial Based on Morning Peak Period Data..... 138

Table 19 Paired-t Test for Arterial Based on Morning Peak Period Data............... 140

Table 20 Simulation Results for Arterial Based on Noon Peak Period Data.......... 141

Table 21 Paired-t Test for Arterial Based on Noon Peak Period Data.................... 143

Table 22 Simulation Results for Arterial Based on Afternoon Peak Period Data .. 144

Table 23 Paired-t test for Arterial Based on Afternoon Peak Period Data 146

Table 24 Cross Street Traffic during Morning, Noon, and Afternoon Peak Periods........ 149

CHAPTER I

INTRODUCTION

PROBLEM STATEMENT

Many urban areas have been experiencing explosive vehicular traffic growth on arterials, causing large amount of delay at arterial intersections. Optimal isolated intersection control and signal coordination along an arterial have been identified as efficient and low cost methods for reducing delay and congestion (*1*,*2*). It was estimated that traffic signal coordination alone reduced delay by 11 million hours and saved $187 million from congestion cost for 85 urban areas in the United States in 2003 (*2*). Considering there are more than 300,000 traffic signals in North America (*3*), the potential saving from improving traffic signal timing is very significant.

Most current traffic signal control systems used in real world are either pre-timed or actuated control. One major problem with the pre-timed signal control is that it does not have the capability to respond to short-term traffic demand and pattern changes (*4*). Traffic actuated control can partially solve this problem by extending green phases in response to real-time traffic arrivals. However, this green phase extension strategy makes decision primarily based on traffic arrivals of the movements being served. Even very long queues on other movements may not stop the extension of the current green phase (*5*,*6*). When traffic demand is heavy, actuated control can result in unsatisfying control performance (*6*).

Adaptive signal control, which adjusts signal timing parameters in response to real-time traffic flow fluctuations, has a great potential to outperform both pre-timed and actuated control and has been researched for the last few decades. Several adaptive signal control systems such as RHODES and OPAC have been developed, and better performances compared with pre-timed and actuated control were reported (*7*,*8*).

This dissertation follows the style of *Transportation Research Record.*

However, many existing adaptive traffic signal control systems are based on dynamic programming, and these systems' applicability may be limited due to restrictions from their problem formulations and solution procedures. In addition, for some of the adaptive control systems using centralized architecture, the maintenance and expansion are difficult and costly. Therefore, it is very significant to develop new and more flexible distributed adaptive control strategies.

OVERVIEW OF THE PROPOSED METHODOLOGY

As one of the key elements of artificial intelligence, reinforcement learning has been successfully applied to control problems such as elevator operation (*9*) and robot soccer games (*10*). It has also been extensively used for supply chain modeling (*11*), activity-travel pattern analysis (*12*), dynamic resource allocation (*13*), and time series prediction (*14*). In this dissertation, a reinforcement learning method is proposed for arterial traffic signal control. In the field of reinforcement learning, the controller is often referred to as agent, which is formally defined as anything that can observe the environment and act upon it, and the environment is the subject to be controlled. A system consists of a group of agents that interact with each other is called a multiagent system (MAS) (*15*). At each decision step, the agent applies an action to the environment in response to the environment's current state. Under the effect of this action, the environment may change accordingly and results in a new state and a feedback signal called reward (or penalty). Based on the new state and the reward, the agent can adjust its policy and learn how to achieve a certain goal from the interactions with the environment (*16*). This learning approach is called reinforcement learning. One advantage of using the reinforcement learning for control applications is that it can learn the optimal control policy directly from interactions between the controller and the environment without knowing the underlying model of the subject to be controlled. In addition, the reinforcement learning method can well circumvent the problems associated with dynamic programming algorithms used in some of the existing adaptive traffic signal control systems. Also, it is conceptually desirable to model arterial traffic signal control problem using reinforcement learning and the MAS framework.

In the case of isolated intersection traffic control, the agent is the traffic signal controller and the environment consists of all other traffic and geometry factors related to the intersection. Queue length or total delay can be used as the penalty. The concept of using reinforcement learning for isolated intersection traffic control is shown in Figure 1.

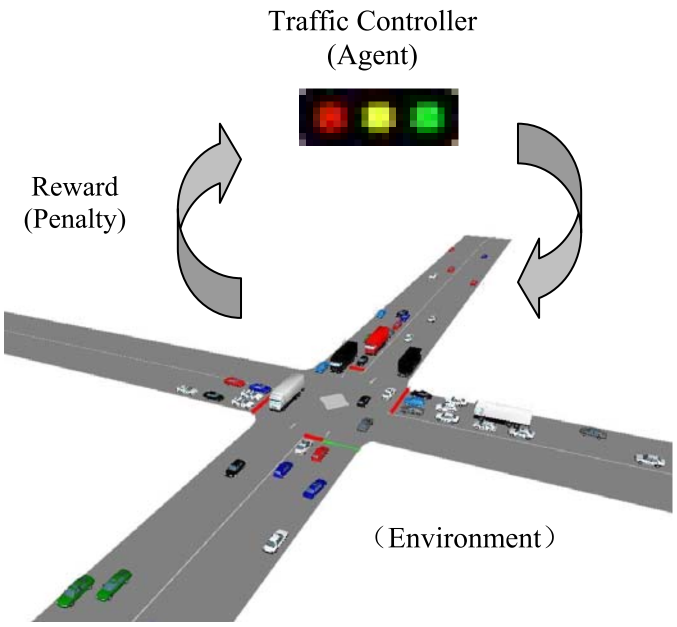


FIGURE 1 Modeling intersection traffic signal control as agent and environment system.

Arterial traffic signal control can be modeled as a MAS and solved by the reinforcement learning method. For a signalized arterial, the signal controller of each intersection is an individually-motivated agent. The agents at different intersections interact with each other and try to optimally control traffic along the arterial. Under the framework of MAS, it is possible to decompose a complicated control system by coordinating agents such that flexibility, efficiency, robustness, and cost effectiveness can be achieved.

Despite the many potential benefits of using MAS for arterial traffic signal control, few thorough investigations have been found in the literature. The potential of applying reinforcement learning and agent technology to traffic signal control, especially arterial traffic signal control, has not been explored fully. Thus, it is imperative to conduct an in-depth research on this topic.

RESEARCH OBJECTIVES

The following objectives are identified in this study.

1. To develop an isolated intersection control method using reinforcement learning. The new control method should be truly demand-responsive and has the ability to better solve the curse of dimensionality and generalization problems.
2. To develop a reinforcement learning control method for arterials based on the proposed isolated intersection reinforcement learning control method.
3. To perform a comprehensive evaluation of the proposed reinforcement learning arterial traffic control method based on a widely-accepted microscopic traffic simulation platform. The reinforcement learning arterial control method will be compared with optimized pre-timed and actuated control in a real-world traffic network.
4. To provide directions for further studies and field implementation of the proposed reinforcement learning arterial traffic control method.

DISSERTATION OVERVIEW

This dissertation consists of six chapters. In the next chapter, various traffic signal control types and strategies are reviewed. The reviewed control types include pre-timed control, actuated control, and adaptive control. In addition, other control strategies such as fuzzy logic control are also reviewed. The review focuses on adaptive traffic signal control methods, and covers all well-recognized adaptive control methods.

In Chapter III, a systematic introduction of reinforcement learning is presented, as reinforcement learning is a relatively new concept to traffic and transportation researchers. The introduction starts with the discussion of Markov property and Markov Decision Processes (MDP), and then proceeds to review various commonly-used reinforcement learning methods such as SARSA, Q-Learning, and Actor-Critic method. Following the introduction of reinforcement learning is a review of studies that applied reinforcement learning to traffic signal control. Problems with the existing applications of reinforcement learning in traffic signal control are also discussed.

In Chapter IV, fuzzy logic control and neural networks are briefly reviewed. After that a new reinforcement learning method based on fuzzy logic control and neural networks is discussed in details. This neuro-fuzzy reinforcement learning method is then applied for isolated intersection and arterial traffic control. In this chapter, two application schemes are considered. The first scheme uses a fixed phase sequence and variable cycle length, while the second scheme has the capability to choosing phases automatically and is not constrained to the traditional cycle concept. Both schemes are further extended for arterial traffic control. A number of strategies to coordinate different traffic signal controllers (agents) on an arterial are reviewed, and a simple but robust coordination strategy is selected to be used in this study.

Chapter V first describes the data and microscopic traffic simulation platform used for evaluating the proposed reinforcement learning control method. Following the description of data and simulation platform is test design, which describes how the proposed control method is evaluated at both isolated intersection and arterial levels, and also under different traffic demand conditions. Finally, the evaluation results from the proposed reinforcement learning control, pre-timed, and actuated control methods are presented, compared, and discussed.

Chapter VI summarizes findings and highlights contributions of this research. Possible future extensions on this research topic are also provided.

CHAPTER II

TRAFFIC SIGNAL CONTROL BACKGROUND AND LITERATURE REVIEW

INTRODUCTION

Intersection traffic control first emerged in the form of manually turned semaphores in London in 1868 (*17*). As an important method to resolve traffic conflicts, improve operational efficiency, and enhance safety at intersections, this idea was soon adapted by other nations and eventually evolved into three major types of traffic signal control strategies: pre-timed, actuated, and adaptive control. Each control type can be applied at an isolated intersection in its simplest form. By properly considering coordination, they can also be used for arterial and network traffic control. This research focuses on development of an adaptive traffic control method that can be used for isolated intersections and also for signalized arterials. Conceptually, the new algorithm introduced in this study can be expanded for network traffic control.

In the rest of this chapter, pros and cons of pre-timed, actuated, and existing adaptive traffic control methods are reviewed in details. In addition, several rule-based control methods are also discussed.

PRE-TIMED TRAFFIC SIGNAL CONTROL

Pre-Timed Isolated Intersection Traffic Signal Control

Figure 2 illustrates a typical four-approach isolated intersection with eight movements (each through movement and its associated right-turn movement are combined as one movement). Each movement is usually labeled by a number between 1 and 8 in NEMA convention (*18*). Pre-timed signal control operates in a cyclic manner. In each cycle there are several signal phases. For each signal phase, one or more non-conflicting movements are allowed. For pre-timed control, the phase sequence and phase duration are fixed. Thus, the cycle length is also fixed. Figure 3 shows a typical example of protected left-leading pre-timed control for an isolated intersection.

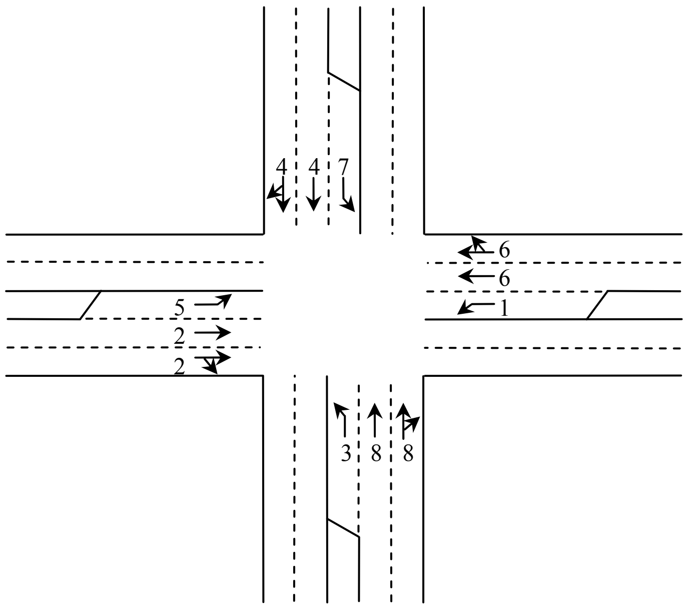


FIGURE 2 Typical four-approach intersection.

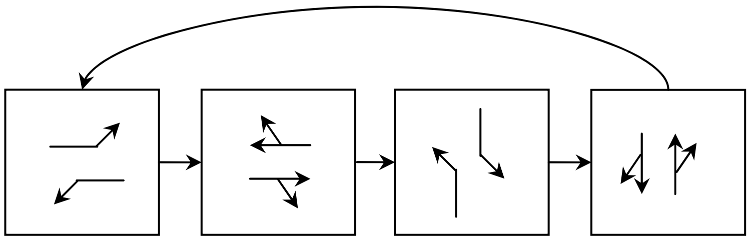


FIGURE 3 An example of protected left-leading pre-timed control.

For isolated intersections, the control parameters are usually optimized based on either the Webster (*19*) method or the procedure in Highway Capacity Manual (HCM) (*20*). In the Webster method, the best cycle length is determined by

$$C = \frac{1.5L + 5}{1 - \sum_{i=1}^{n} y_{ci}} \quad (1)$$

where

$C =$ optimal cycle length (s);

$L =$ total lost time per cycle (s);

$y_{ci} =$ observed flow divided by saturation flow rate for the critical lane group in phase i. This is often referred to as critical flow to saturation flow ratio, or the critical *v/s* ratio; and

$n =$ number of phases in a cycle.

After the optimal cycle length is determined, the effective green time is allocated to each phase based on the critical *v/s* ratios of all phases, calculated as

$$g_i = \frac{(v/s)_{ci}}{\sum_{i=1}^{n} (v/s)_{ci}} \times G \quad (2)$$

where

g_i = effective green time in phase *i*, typically equaling the actual green interval duration for the phase;

$v =$ flow rate;

$s =$ saturation flow rate; and

$G = C - L =$ total green time in a cycle (s).

The HCM method (*20*) uses the similar principle as the Webster method to allocate green time among different signal phases, which is to equalize the degree of saturation (*v/c* ratio) of critical lane group of each signal phase. The degree of saturation is defined as

$$X_{ci} = \frac{v_{ci}}{c_{ci}} = \frac{v_{ci}C}{s_{ci}g_{ci}} \qquad (3)$$

where

$C =$ cycle length (s);

$X_{ci} =$ critical degree of saturation for phase *i*;

$c_{ci} = s_{ci}\frac{g_{ci}}{C} =$ critical lane group capacity for phase *i*; and

$g_{ci} =$ green time allocated to phase *i* (s).

X_c, or the critical *v/c* ratio for the entire intersection, can be defined accordingly as

$$X_c = \sum_{i=1}^{n}\left(\frac{v}{s}\right)_{ci}\left(\frac{C}{C-L}\right) \qquad (4)$$

From this equation, it is easy to derive the formula for calculating cycle length

$$C = \frac{L \cdot X_c}{X_c - \sum_{i=1}^{n}\left(\frac{v}{s}\right)_{ci}} \qquad (5)$$

In practice, a desired value of X_c is chosen first, and the cycle length is then determined by Equation (5).

Pre-timed timing plans are determined based on average traffic volume data. In the real world, traffic volume may change considerably throughout the day and also in short intervals. Obviously, a control method based on average traffic volume data cannot effectively consider traffic flow fluctuations and may result in suboptimal control. Therefore, in practice the applications of pre-timed control are often limited to locations with less variable traffic. Besides, the control parameters of pre-timed control need calibrations from time to time to reflect mid-term or long-term traffic flow pattern changes.

Pre-Timed Arterial Traffic Signal Control

For closely spaced intersections on an arterial, vehicle arrivals to a downstream intersection are often affected by the control strategies of the upstream intersections. Vehicles also travel in platoons. Thus, it is desirable to coordinate the pre-timed traffic signals of adjacent intersections such that platoons of vehicles can get through a number of intersections without being stopped. For this purpose, offset is used in addition to cycle length, phase sequence, and phase duration to coordinate adjacent traffic signals (*21*). The offset is defined as the difference between the starting times of two reference phases. Assuming all three intersections in Figure 4 use the northbound through phase as the reference phase, the offset between intersections 1 and 2 is *off_*1, and the offset between intersections 1 and 3 is *off_2*.

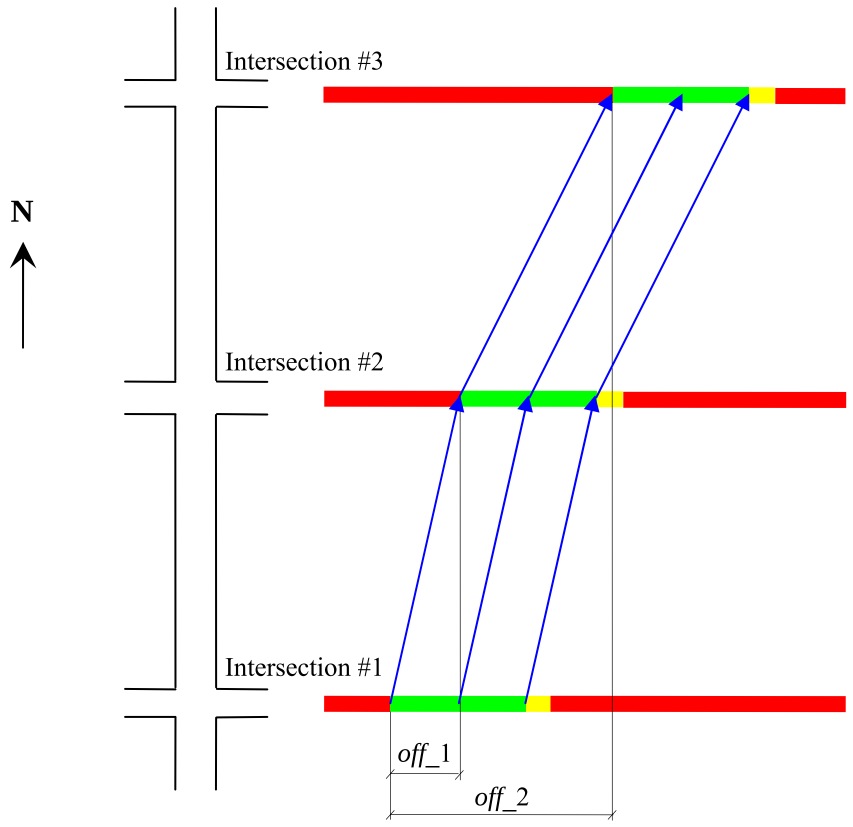


FIGURE 4 Offsets and signal coordination.

A commonly-used signal coordination strategy is to maximize the bandwidth of through movements along the arterial, which may maximize the number of vehicles that go through the arterial without being stopped. However, this strategy may not be effective due to the following reasons:

1. Coordinated pre-timed method usually requires all traffic signal controllers being coordinated to have the same cycle length. For different intersections on an arterial, their optimal cycle lengths most likely are different. Requiring a common cycle length for all intersections may cause increased level of delay to vehicles at some of the intersections.

2. Offsets are calculated based on the distances between two intersections and the average speed. In many cases, travel time between two adjacent intersections may vary depending on flow and queuing situations.
3. Large percentage of turning traffic can make the control strategy less efficient.
4. For two-way traffic, the offset in one direction also determines the offset in the other direction. It is difficult to give both directions equally good coordination.
5. Coordinated pre-timed control gives higher priority to traffic on main streets. This may cause cross street traffic to experience unreasonably large delays.

Due to these problems, some researchers proposed to minimize delay and the number of stops in addition to maximizing bandwidth. Other researcher developed a variable bandwidth control method for arterials (*22,23,24*). Despite these improvements, they are still pre-timed control methods. Similar to pre-timed control for isolated intersections, the cycle length, phase sequence, phase duration, and offset of coordinated pre-timed control are also fixed during a given period of operation. Thus, coordinated pre-timed control still suffers from the same problems that isolated pre-timed intersection control has, and it lacks the flexibility to deal with short-term traffic flow variations.

ACTUATED TRAFFIC SIGNAL CONTROL

Actuated Signal Control at Isolated Intersection

Actuated control provides an intermediate solution between pre-timed control and adaptive control (*17*). It can be further classified into semi- and fully-actuated control (*25*). Actuated control is based on the fundamental principle shown in Figure 5. The length of green phase falls between the preset minimum and maximum green times. After the minimum green time is served, as long as there is a vehicle actuation occurs before the preceding vehicle extension ends and the total green extension has not exceeded the preset maximum green time, another green extension will be given to the current green phase. This actuated control strategy can partially solve the criticism attributed to the pre-timed control strategy in a sense that it can respond to the real-time traffic arrivals of the current green phase. However, this actuated control strategy does not take into consideration of the queue lengths on other conflicting movements, and may result in suboptimal control especially under heavy traffic conditions.

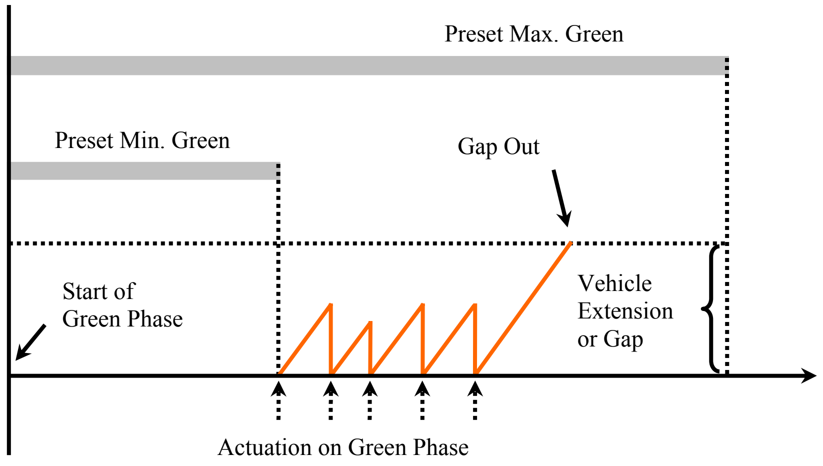


FIGURE 5 Actuated signal control (*25*).

Actuated Traffic Signal Control on Arterial

When applying actuated signal control to isolated intersections, the cycle length may vary from cycle to cycle, as the phase durations are variable depending on actual traffic arrivals. When applying actuated control to arterials, the coordinated actuated control must have a constant cycle length and a coordinated phase should be defined for each intersection. Actuated control is considered to be more suitable for arterial traffic signal control than pre-timed control (*8,17*). However, it still has unsolved problems such as "the-early-return-to-green" (*17*), which may cause unnecessary stops of vehicles.

ADAPTIVE SIGNAL CONTROL

In the following sections, a number of well known adaptive traffic signal control systems are reviewed. Adaptive traffic signal control systems are normally complicated and include prediction and estimation modules, it is difficult to cover every detailed aspect of each system. Therefore, this review only focuses on the system designs and architectures, problem formulations, solution procedures, and optimization algorithms of the existing systems. The existing systems that are reviewed include UTCS (*8*), SCOOT (*26*), SCAT (*27,28*), DYPIC (*29*), OPAC (*8*), RHODES (*7*), UTOPIA (*30,31*), PRODYN (*32*), ALLONS-D (*33*), and MDP&DP (*34*).

Urban Traffic Control System (UTCS)

Starting from the 1970s, the U.S. Department of Transportation (USDOT) conducted several research projects on urban traffic control system (UTCS) (*8*). The intersection control strategies proposed and evaluated in these projects can largely be classified into three categories: first-generation control (first-GC), second-generation control (second-GC), and third-generation control (third-GC). First-GC strategy generates traffic control plans based on historically averaged traffic volume data. Depending on the time-of-day (TOD), different pre-timed control plans are selected and implemented. The updating frequency for the control plans is usually 15 minutes; second-GC strategy optimizes traffic signal control plans every 5 minutes based on predicted traffic volume data instead of historical data. The updating frequency for traffic signal control plans is restricted to be no less than 10 minutes in belief that this can avoid transition disturbances; third-GC is similar to the second-GC, but updates signal timing plans using a shorter interval of 3-5 minutes (*35*).

Split, Cycle and Offset Optimization Technique (SCOOT)

Hunt et al. (*26*) developed the SCOOT system, which is considered to be equivalent to a second-GC (*36*) or third-GC (*17*) method. In SCOOT, intersections are grouped into many sub-areas and signal controllers in each sub-area operate at a common cycle length. SCOOT makes frequent and small changes to signal control parameters such as cycle length, phase duration, and offset of a pre-timed plan based on actual traffic flow variations (*17,37)*. The adjustment of signal control parameters is based on a traffic model that predicts delay and stops resulted from different signal timing plans. The plans that can best reduce delay and stops are then selected and implemented (*26*). SCOOT has been widely used in the United Kingdom. There are also a few implementations of it in other countries. The latest version of SCOOT is SCOOT MC3 (*38*), which has some new features such as the ability to skip phases for bus priority purpose.

Sydney Coordinated Adaptive Traffic System (SCATS)

SCATS (*27,28*) was developed by Australian researchers. It is similar to SCOOT and is considered to be an adaptive control method between first-GC and second-GC (*17*). A major difference between SCATS and SCOOT is that SCATS does not have a traffic model or a traffic signal control plan optimizer. SCATS selects the best phase durations and offsets from some predefined plans (*17*) based on real time traffic flow conditions.

SCATS has a hierarchical system structure, which has three levels as shown in Figure 6. The lowest level consists of the local controllers at each signalized intersection. They perform tasks such as data collection, data preprocessing, and assessment of detector malfunctions. In the middle level are the regional masters, which are the core of SCATS. Each regional master controls up to several hundred local controllers, and these controllers are further grouped into systems and sub-systems. Sub-systems usually consist of several intersections and are the smallest control element on multi-intersection level. The highest level is the control center, which does not really perform any specific control operations. The purpose of the control center is mainly to monitor the entire system.

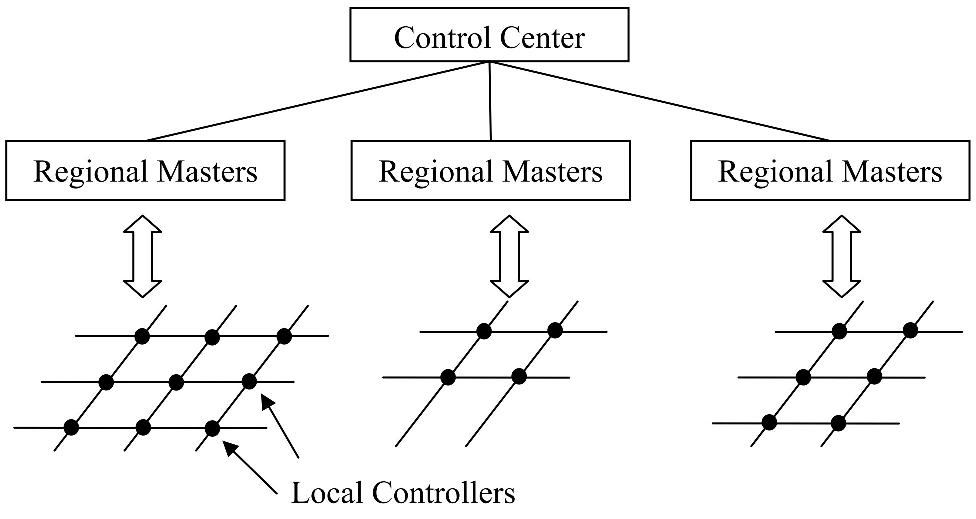


FIGURE 6 Structure of SCATS.

Dynamic Programmed Intersection Control (DYPIC)

Robertson and Bretherton (*29*) developed an optimal control method called DYPIC based on dynamic programming for an isolated intersection. A simple intersection with only two conflicting movements was used by Robertson and Bretherton to illustrate their method. Since there were only two conflicting movements, the control decisions were to either extend or terminate current green signal. In their study, Robertson and Bretherton assumed that exact traffic arrival information in the next few minutes (over the decision horizon) was known. However, this is impossible in real world applications. Thus, the DYPIC control method was used mainly for theoretical studies and for comparison with other practical control methods.

In the DYPIC method, the entire decision horizon was divided into *N* intervals. Each interval was 5 seconds long. At the end of each small interval (decision point), the control logic made a decision to either extend the current green phase or terminate it and give green to the other movement. There were no constraints such as minimum and maximum green times. Robertson and Bretherton (*29*) formulated this intersection control as a dynamic programming problem. Specifically, the decision point corresponded to the concept of stage in dynamic programming; states at each stage were characterized by the signal state (green or red) and the queues on each approach. As the exact traffic arrival information was assumed to be known for the entire decision horizon, queue lengths of each approach at any stage can be estimated using some traffic models. The optimization goal was to find an optimal control strategy consisting of a sequence of actions $A = \{a_1, ..., a_N\}$ that minimize the total delay. Based on the initial signal states, queue lengths of each approach, and future traffic arrival information, the entire decision process can be illustrated by a decision tree as shown in Figure 7.

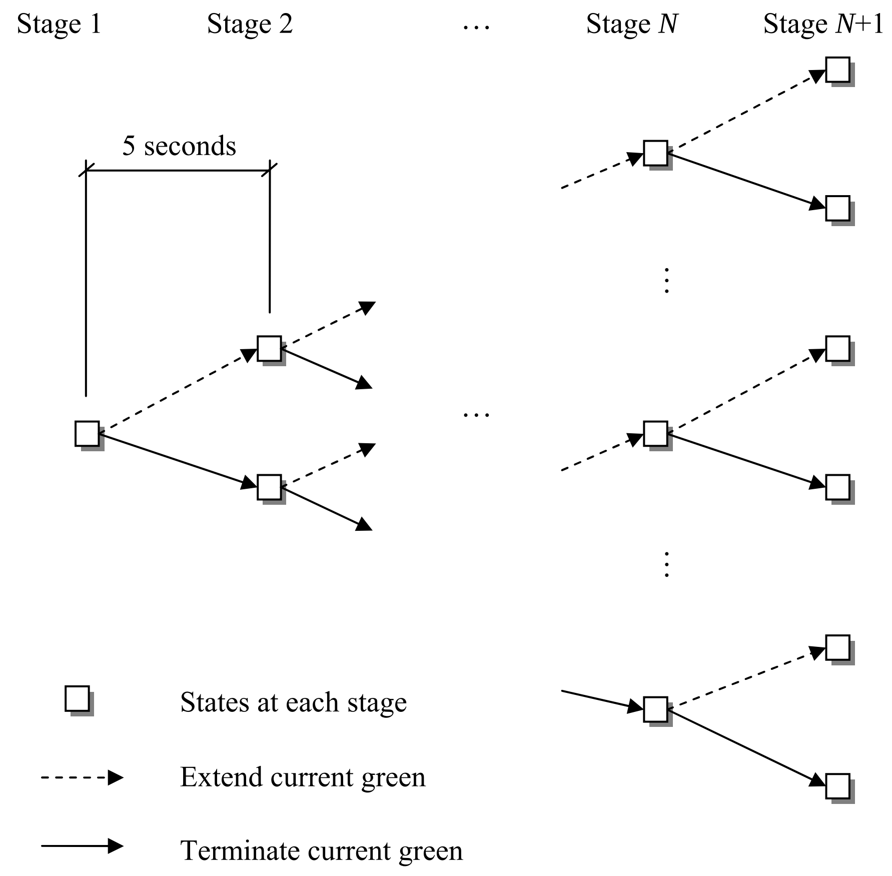


FIGURE 7 Illustration of the DYPIC method.

The following formula was used in DYPIC to find the optimal control strategy for the decision problem shown in Figure 7.

$$f_i(j) = \min_{a_i}\{C_{jk} + f_{i+1}(k)\}, \quad i = 1,...,N, \quad j \in S_i, \quad k \in S_{i+1} \tag{6}$$

where

S_i = all possible states at stage *i*;

S_{i+1} = all possible states at stage *i*+1;

C_{jk} = total delay associated with transition from state *j* at stage *i* to state *k* at stage *i*+1;

$f_i(j)$ = value function for state *j* at stage *i*;

a_i = action taken at stage *i* (either extension or termination); and

N = number of stages minus 1.

Starting from stage *N* and working backwards, the values of each state at stages 1 through *N* can be obtained using Equation (6). The value for the initial state actually is the minimum delay resulted from the optimal control strategy. By tracking the path that leads to the value of the initial state, one can find the best control strategy. This method is often referred to as the backward dynamic programming. It is based on the Bellman principle of optimality, which states that no matter what the previous decisions are, the remaining actions must be optimal given the current states. The detailed solving procedure of the backward dynamic programming will not be discussed here. Interested readers can refer to (*29*) or some other dynamic programming textbooks.

There are four major problems with the DYPIC method. First, as shown in Figure 7, since each action may result in two states in the next stage, if there are *N*+1 stages, the maximum possible number of states at the final stage is 2^N. If the decision horizon is 2 minutes and the interval is 5 seconds long, then the maximum possible number of states at the final stage could be $2^{120/5} = 16777216$. Although dynamic programming theoretically can give this problem a globally optimal solution, so many states will definitely make the computation time a serious problem for real time traffic signal control. Secondly, in this simple example only an isolated intersection with two movements was considered. For practical traffic signal control problems, usually there are eight movements and at each stage there could be multiple different actions. Thirdly, this example assumed all traffic arrivals during the decision horizon were known. This is impossible in reality. Finally, the DYPIC method assumed deterministic state transitions. Given current queue lengths, signal states, traffic arrival information, and the action to be applied, the resulted new state was determined. This assumption in fact may not always be true, as driver behaviors are very complex and no traffic models can perfectly predict future traffic states.

Note that for the DYPIC control method, since the phase durations and phase sequence are not fixed, the cycle length is not a fixed value. This is different from pre-timed control, coordinated actuated control, SCOOT, and SCAT. Robertson and Bretherton (*29*) compared the DYPIC control method with pre-timed control on an isolated intersection. Two different traffic arrival conditions were tested, which were random arrival and cyclic arrival. For random arrival condition, the results showed that the DYPIC method reduced delay by at least 50 percent. While for the cyclic arrival condition, limited tests showed that the DYPIC method reduced average delay by 3 seconds per vehicle.

Optimized Policies for Adaptive Control (OPAC)

The second-GC and third-GC strategies were expected to perform better than the first-GC strategy, as they seemed to provide better responsiveness to traffic conditions by using detected and predicted dada. However, some field tests showed that the first-GC strategy in general outperformed the other two strategies (*35,39,40*). Due to the unsatisfactory results of the second-GC and third-GC strategies (*35,39,40*), Gartner (*35*) suggested a truly demand-responsive control strategy that is not restricted to the conventional concepts of cycle length and phase durations (*8*).

In his research, Gartner first presented an isolated intersection traffic control example using a dynamic programming approach, later named as OPAC-1 (*41*), that was similar to DYPIC (*29*). Gartner discussed that though this dynamic programming approach can guarantee global optimality, it is not suitable for real time applications due to the excessive computation time and the requirement of exact traffic arrival data. Based on OPAC-1, Gartner proposed a simplified control algorithm using Optimal Sequential Constrained Search (OSCS) algorithm instead of dynamic programming (*8*). The resulted new control method was referred to as OPAC-2. The OSCS algorithm requires less computation time but can produce results that are close to the optimal ones produced by the dynamic programming approach. However, the OSCS algorithm is less straightforward compared with the dynamic programming approach. To implement the OSCS algorithm, there are three steps to follow (*8*):

1. The entire decision horizon is divided into several stages. Each stage is 50 to 100 seconds long.
2. In each stage, the signal must be changed once and at most three signal changes can be made. There could be many different signal change scenarios in each stage. For each scenario, the resulted delay is calculated.
3. For each stage, the OSCS algorithm is applied. The optimal signal change scenario is determined independently for each stage. As shown in Figure 8, the inputs to any intermediate stage include queues of all approaches at the end of previous stage, current signal status, and the last signal change. For all feasible signal change scenarios, their corresponding delays are evaluated and compared. The signal change scenario with the lowest delay value is stored and used as the optimal solution for the current stage. The resulted queues, signal status, and the last signal change information are passed on to the next stage for further computation.

In the same study (*8*), Gartner also proposed a rolling horizon approach to predict traffic arrivals such that the constraint of knowing exact traffic arrivals was removed. By adding the rolling horizon prediction method, OPAC-2 evolved into OPAC-3 (*41*). Gartner evaluated the OPAC-3 control strategy using a special version of NETSIM. The evaluation was based on data collected from an intersection in Tucson, Arizona. The results showed that compared to the existing control method deployed at that intersection, OPAC reduced average delay by 30-50 percent and increased average speed by 10-20 percent. Although the improvement from OPAC was significant in this study, Gartner did not specify if the original control strategy was optimized or not.

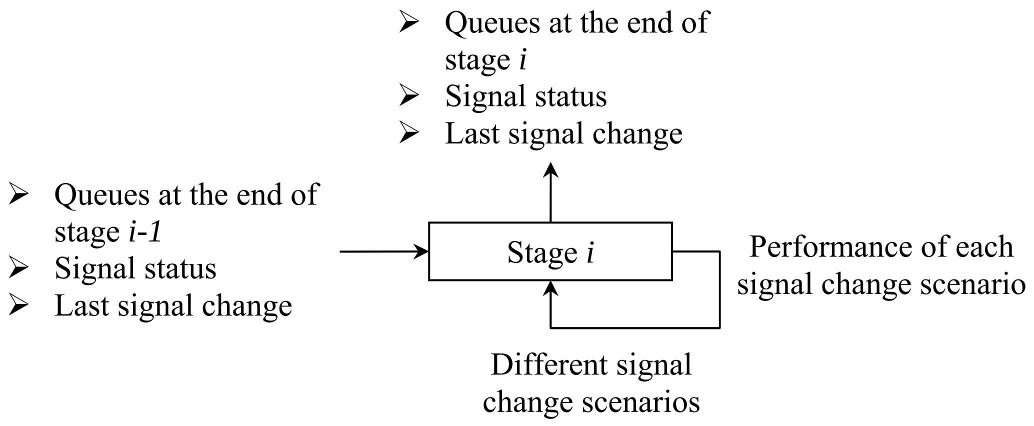

➢ Queues at the end of stage *i*
➢ Signal status
➢ Last signal change
➢ Queues at the end of stage *i-1*
➢ Signal status
➢ Last signal change
Stage *i*
Performance of each signal change scenario
Different signal change scenarios

FIGURE 8 A simple illustration of the optimal sequential constrained search method.

In a subsequent study, Garter et al. summarized the development of OPAC and the results of an application of its latest version OPAC-4 (*41*). OPAC-4 was developed to extend the application of OPAC from single intersections to arterials and networks. OPAC-4 uses a Virtual-Fixed-Cycle (VFC) technique and is often referred to as VFC-OPAC. The VFC-OPAC has a hierarchical structure with three layers as shown in Figure 9. The synchronization layer calculates the VFC every few minutes. Based on this VFC, the coordination layer optimizes the offset of each intersection. The local control layer optimizes signal changes subject to VFC and offset constraints from the synchronization and coordination layers. Although it is conceptually clear how the VFC-OPAC works for arterial and network traffic control. Details about this control process are not available in (*41*) and any other literature.

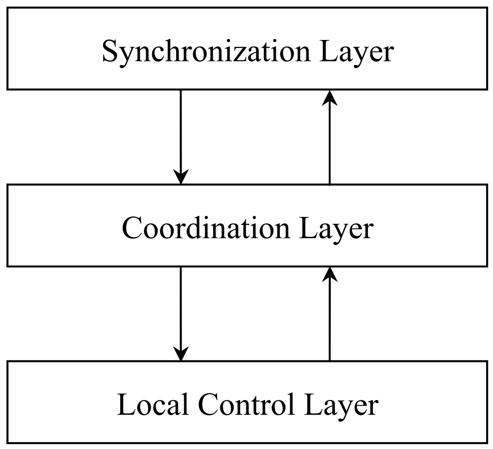


FIGURE 9 The hierarchical control structure of VFC-OPAC.

The OPAC-4 system was later tested on an arterial in Reston, Virginia. The field test was carried out in two steps. In step one, the existing coordinated pre-timed control system was retimed and performance data were collected. In step two, the OPAC-4 system was implemented and its performance data were also collected. Comparison of travel time data showed that the performance of the existing coordinated pre-timed control and OPAC-4 control were not significantly different from each other. Gartner et al. (*41*) explained that this might be caused by the traffic flow pattern changes between the two data collection periods.

Real-Time Hierarchical Optimized Distributed Effective System (RHODES)

RHODES is an adaptive traffic signal control system with a hierarchical structure. It was developed at the University of Arizona (*7*). RHODES has two core modules: prediction and control. The prediction module predicts future traffic arrival information such as when and how many vehicles will arrive, while the control module is used to control intersection and network traffic flows. The intersection control logic uses an algorithm developed by Sen and Head (*42*). This algorithm is called Controlled Optimization of Phases (COP) and is also based on dynamic programming. The network control logic used in RHODES is based on both COP and REALBAND (*43*). REALBAND algorithm is used to produce progression bands in terms of observed platoons in the network, and these progression bands are then used as constraints for COP to develop optimal control strategies for individual intersections.

Although the COP algorithm is also based on dynamic programming, it uses definitions for stages, states, and actions that are quite different from DYPIC and OPAC methods. In COP, stages are defined as a sequence of phases; states at each stage are defined as the number of time steps that could be assigned to the current stage; the optimization goal is to find an optimal plan to allocate time steps to each stage (phase) such that the overall vehicle delay/number of stops/queue lengths could be minimized. This modeling approach is similar to applying dynamic programming to resource allocation problems (*44*). The success of both the OPAC and RHODES models relies on an accurate prediction of traffic arrivals over the entire decision horizon, which is very difficult in reality.

Urban Traffic Optimization by Integrated Automation (UTOPIA)

Several Italian researchers proposed an adaptive control method called UTOPIA (*30,31*). UTOPIA explicitly considers the priority of public transport. It adopts a hierarchical structure with two levels: area control and local control.

The area control aims at minimizing the following objective

$$\min_{\alpha_i^k,\beta_i^k} \sum_i \sum_k p^k(x_i^k,\alpha_i^k,\beta_i^k) \tag{7}$$

where

x_i^k = number of vehicles on link *k* during the *i*th interval;

$\alpha_i^k =$ coefficient related to average overall travel speed on link k during the i^{th} interval;

$\beta_i^k =$ coefficient related to saturation flow on link k during the i^{th} interval;

$p^k(\cdot) =$ cost function for link k;

$i =$ index for time intervals; and

$k =$ index for links.

The area controller continuously provides the optimized α_i^k and β_i^k values to local controllers, and the local controllers try to find an optimal solution to the following optimization problem

$$\min_{c_i^m} \sum_i L^{\mathrm{m}}(y_i^m, c_i^m, \alpha_i^m, \beta_i^m) \tag{8}$$

where

$y_i^m =$ a vector of queue lengths for each approach of intersection m during the i^{th} interval;

$c_i^m =$ state of the traffic signal for intersection m during the i^{th} interval;

$L^m =$ cost function for intersection m;

$i =$ index for time intervals; and

$m =$ index for intersections.

UTOPIA models isolated intersection control as a multi-stage decision problem. The entire decision horizon is usually 120 seconds. Each stage (interval) is 6-second long. A rolling horizon technique is also adopted in UTOPIA (*30*).

Equations (7) and (8) only give a very general description of the optimization models used in UTOPIA. It was briefly mentioned in a paper by Davidsson and Taranto (*45*) that a branch and bound algorithm was used in UTOPIA to solve the local controller optimization problem. Other than these, no other detailed descriptions of the UTOPIA method can be found in existing literature.

PRODYN

PRODYN is an adaptive traffic signal control method developed by Henry et al. (*32*) that is also based on dynamic programming. Similar to many adaptive signal control methods based on dynamic programming, PRODYN does not have fixed phase sequence, phase durations, and cycle length, and it uses a hierarchical algorithm for network traffic control.

PRODYN has two versions (*32,46,47*). The initial version of PRODYN has a hierarchical structure with two levels (*32*). The lower level is for intersection control and the upper level is for arterial and network coordination. The optimization process of the initial version of PRODYN is shown in Figure 10.

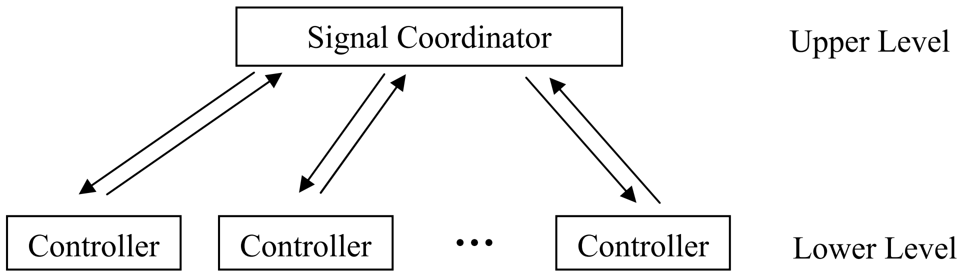


FIGURE 10 The hierarchical control structure of the initial version of PRODYN.

The lower level consists of many intersection controllers, which generate initial traffic signal timing plans to be sent to the upper level signal coordinator. Upper level signal coordinator then provides feedbacks to each intersection, and intersections use these feedbacks to improve their initial time plans. This is an iterative process and will stop until an agreement is reached between the coordinator and lower level controllers.

The hierarchical structure shown in Figure 10 was abandoned in the later version of PRODYN (*46,47*). The new version uses a forward dynamic programming method for individual intersection control and a decentralized structure for coordinating different intersection controllers. To use the forward dynamic programming method, the individual intersection control is first modeled as a multi-stage decision problem. Similar to OPAC, the decision horizon in PRODYN is divided into many small time intervals and each interval is a stage. States are characterized by a number of variables including current signal phase and queue lengths. The decision at each stage is either to keep the current signal phase green or to switch to the next signal phase. PRODYN also uses a rolling horizon method. Assume the length of each stage (time interval) is T, which is usually 5 seconds. The length of the decision horizon is $N*T$, and N is an integer value that can be set to any reasonable numbers such as 15. At stage *i*, the best control policy during time interval $[i*T,(i+N)*T]$ is decided by the current intersection states and traffic arrivals between $[i*T,(i+N)*T]$.

A decentralized coordination method is used in PRODYN. Based on some general descriptions in (*46,47*), this decentralized coordination method is summarized as the following procedure

1. Choose one intersection and optimize its control over the entire decision horizon;
2. Based on the optimized control decision, simulate traffic flow outputs from this intersection over the decision horizon;
3. Send the simulated traffic flow outputs to downstream intersections and move to the downstream intersections;
4. Based on the simulated flow outputs sent from upstream intersections, optimize traffic control for the current (downstream) intersection; and
5. Go to step 2.

One problem with this procedure is how to choose the first intersection. Another issue is that a downstream intersection can also be an upstream intersection if it is a two-way street, which is often the case. Thus for two adjacent intersections A and B, if the simulated traffic flow outputs of A are used for optimizing traffic control of intersection B, then the simulated traffic flow outputs from intersection B will retroact on the optimality of the optimized traffic control plan of intersection A. The interaction between intersections A and B will form a circle. An iteration process may be needed in the PRODYN's coordination algorithm to ensure that equilibrium can be eventually reached. However, it is unclear if such iteration process is included in PRODYN based on descriptions in available literature (*46,47*).

Adaptive Limited Look-ahead Optimization of Network Signals – Decentralized (ALLONS-D)

Porche (*33*) proposed a decentralized adaptive traffic signal control method called ALLONS-D in his dissertation. ALLONS-D is based on a depth-first branch and bound algorithm and uses a decision tree to help find the best control sequence (*33*). The decision tree used in ALLONS-D is similar to the one used in DYPIC as shown in Figure 7, in which each node represents a decision point and has a cost value associated with it while each arc is a control action. Figure 7 only shows the decision tree for an isolated intersection with two-phase control. For intersections with four or more phases, the size of the decision tree will make exhaustive search methods infeasible for real time applications. To improve searching efficiency, ALLONS-D uses the branch and bound algorithm and a special technique called "Serve the Largest Cost" (STLC) to find the best control sequence. The entire optimization process of ALLONS-D can be divided into two parts: 1) initial decision path (sequence) building, and 2) backtracking and exploration.

In the decision path building part, a feasible decision path is constructed using the STLC technique. In terms of the STLC technique, at each decision point, the control phase incurring the highest delay in a most recent time period should be turned green. Following this STLC policy, a sequence of decisions is made until the initial queues and predicted traffic arrivals are cleared. This process can be better illustrated in Figure 11. The reason why the STLC is used is that Porche (*33*) believed it may result in a path that is close to the optimal one. It is intuitive that the closer the initial decision path is to the optimal one, the less computation time is required for the backtracking and exploration part that follows.

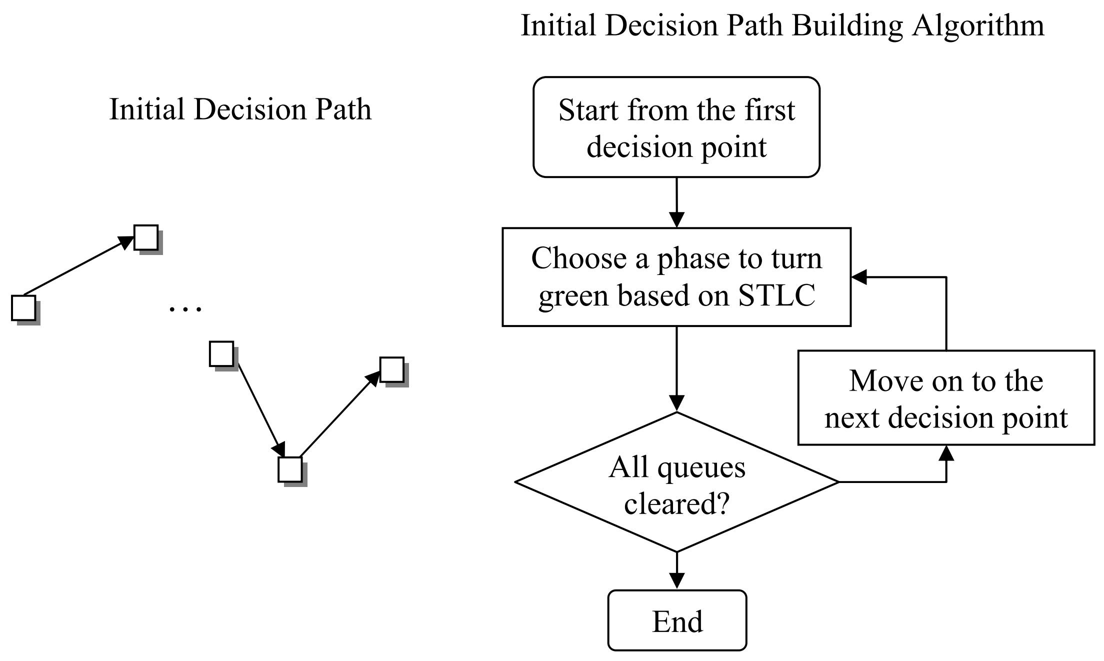

Initial Decision Path Building Algorithm
Initial Decision Path
…
Start from the first decision point
Choose a phase to turn green based on STLC
Move on to the next decision point
All queues cleared?
End

FIGURE 11 Initial decision path building of ALLONS-D.

The initial decision path in most cases is not optimal. Therefore, a backtracking and exploration process is needed to further improve the initial decision path. The backtracking process is similar to the backward recursive method for solving the dynamic programming problem shown in Equation (6), while the addition of an exploration process distinguishes it from the backward recursive method. The backtracking and exploration is a recursive process that starts from the end node of the initial decision path as shown in Figure 11. The corresponding cost value for the end node is zero, as all queues are assumed to be cleared at this point. Set the initial decision path as the Current Best Decision Path (CBDP). The process goes back one interval for each iteration and calculates the cost value of the current node. For every node except for the end one, all branches growing up from it will be evaluated and compared with the CBDP. A cost value is defined for both arcs and paths. The delay cumulated during each interval is defined as the arc cost, and the path cost is the summation of the costs of all arcs in the path. If any of the branches have a smaller cost than the branch in the CBDP, then the branch in the CBDP will be replaced by the new branch. Otherwise, the exploration from this node will be terminated, and the process will go back one interval and set the parent node as the current node. A flow chart in Figure 12 is used to better illustrate the backtracking and exploration process.

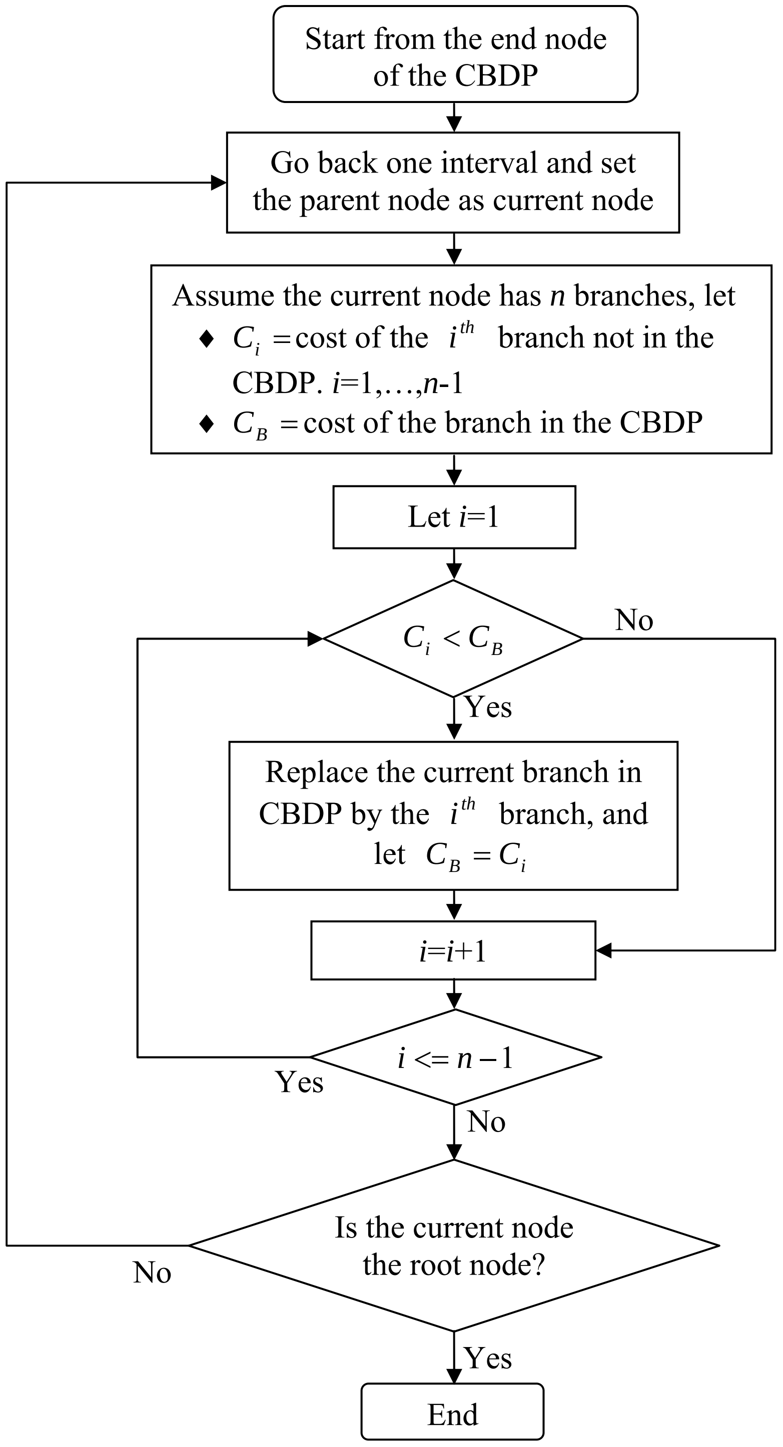

Start from the end node of the CBDP
Go back one interval and set the parent node as current node
Assume the current node has *n* branches, let
♦ C_i = cost of the i^{th} branch not in the CBDP. *i*=1,…,*n*-1
♦ C_B = cost of the branch in the CBDP
Let *i*=1
$C_i < C_B$
No
Yes
Replace the current branch in CBDP by the i^{th} branch, and let $C_B = C_i$
i=*i*+1
$i <= n-1$
Yes
No
Is the current node the root node?
No
Yes
End

FIGURE 12 Backtracking and exploration of ALLONS-D.

The ALLONS-D algorithm introduced so far is for isolated intersection control. For arterial traffic control, Porche (*33*) considered two coordination methods. The first coordination method assigned different weights to each direction. For example, if north-south direction was the main street, then larger weight was assigned to north-south direction. Porche tested this control method on a three-intersection arterial. However, it seemed that this method did not perform well. Another coordination method Porche proposed was game theory. Porche only conceptually showed that game theory may be used for coordinating traffic signal controllers. No experiments were conducted to show if this method can really be applied to coordinate traffic signal controllers. Also, Porche did not mention if this game theory coordination method is suitable for real time application, which is very important for adaptive traffic signal control systems.

Markov Decision Process and Dynamic Programming (MDP&DP)

More recently, Yu and Recker *(34)* developed a stochastic adaptive traffic signal control model. The authors formulated traffic signal control as a Markov Decision Process (MDP) and solved it by dynamic programming algorithms. MDP is a discrete time stochastic process characterized by a set of states (*S*), actions (*A*), reward function (*r*), and state-transition function (*p*). In the context of intersection traffic signal control, the state variables are the queue lengths of all approaches; the action variables are the control actions that can be taken for each state; the reward function tells the immediate reward of each action under specific state; and the state-transition probability function is time-varying and dependent on actual traffic arrivals. To solve control problems modeled as MDPs, the first step is to find the optimal value function $V^*(s)$ based on Equation (9) (*16*).

$$V^*(s) = \max_{a \in A(s)} \left\{ \sum_{k \in S} p_{sk}^{a} r_{sk}^{a} + \gamma \sum_{k \in S} p_{sk}^{a} V^*(k) \right\}, \quad \forall s \in S \tag{9}$$

where $a \in A(s)$; $s \in S$ is the current state and $k \in S$ is the next state after action a is taken; p_{sk}^{a} and r_{sk}^{a} are the transition probability and reward, respectively, from state s to state k after action a is taken; and $\gamma \in [0,1)$ is a discount factor. Equation (9) is often referred to as the Bellman optimality equation (*16*). Based on this Bellman equation, Yu and Recker (*34*) used a dynamic programming method to solve for the optimal value function $V^*(s)$. After the $V^*(s)$ is found, the control problems simply become identifying the current system state s and applying the control action $a \in A(s)$ that leads to the optimal value function $V^*(s)$. This mapping from system state to an action is called policy, which is a very important concept that will be used frequently in this dissertation.

Although dynamic programming algorithm can be used to solve this MDP problem and is guaranteed to find the optimal policy (*48*), it needs a well-defined state-transition probability function. In practice, this state-transition probability function is difficult and cumbersome to define. In the case of intersection traffic control, the state-transition probability function is affected by actual traffic arrivals and is often time-varying. Thus, it is even more difficult to give accurate estimation. In addition, for intersection traffic signal control applications, the number of states is usually very large. This makes the computation time of dynamic programming algorithms a serious problem (*9,49,48,34*). Nevertheless, it is a legitimate attempt to use MDP to model intersection traffic control problems. Unlike DYPIC and OPAC methods that assume a deterministic state transition, MDP implicitly acknowledges the uncertainty in state transition and reflects this uncertainty by a state-transition probability function.

TRAFFIC CONTROL USING FUZZY LOGIC AND RULES

Several studies have applied fuzzy logic to traffic signal control (*6,50,51,52,53,54*). These fuzzy logic methods use queue lengths and traffic arrivals on all approaches as inputs, and the control action is usually determined based on a number of fuzzy rules.

The following are two simple examples of fuzzy rules (6) that are used to determine the extension of the current green phase.

1. **IF** current queue length is {Short} **AND** arrival is {Low} **AND** conflicting queue length is {Medium}, **THEN** extension is {Short}
2. **IF** current queue length is {Medium} **AND** arrival is {High} **AND** conflicting queue length is {Short}, **THEN** extension is {Long}

The inputs to the fuzzy logic control system are fuzzified first such that they can be used in the fuzzy rules. The fuzzification of inputs is accomplished by membership functions. Figure 13 shows some examples of fuzzy membership functions.

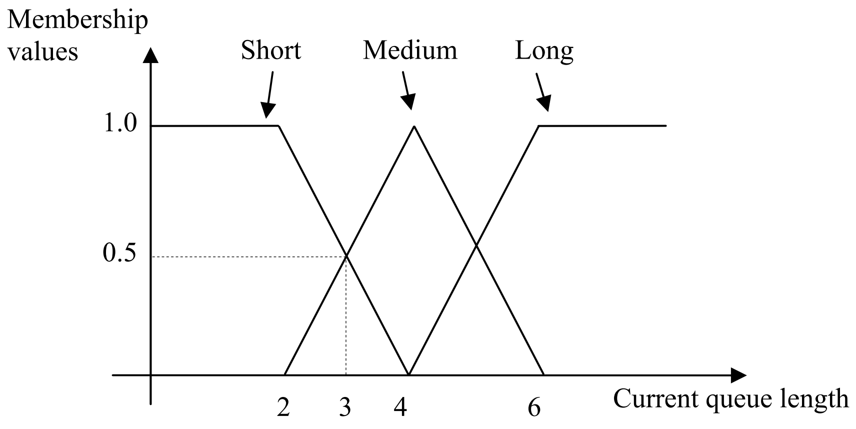


FIGURE 13 Fuzzy membership functions of the current queue length.

Based on the fuzzy membership functions in Figure 13, for a current queue length of 3 vehicles, the memberships that the current queue length belongs to {Short}, {Medium}, and {Long} are 0.5, 0.5, and 0.0, respectively. Similarly, one can apply the same procedure to the other two input variables, arrival and conflicting queue length, and obtain their corresponding membership values. All these membership values will be used for computing strength of each fuzzy rule, which is often referred to as firing strength. Each fuzzy rule corresponds to an output fuzzy set. These output fuzzy sets together with their corresponding firing strengths are used to obtain a crisp value, because one cannot directly use linguistic outputs such as “extension is {Short}” and “extension is {Long}” for practical traffic control. More discussions on fuzzy logic traffic signal control will be provided later in Chapter IV, and interested readers can also refer to (*55,56*) for detailed information on fuzzy logic.

An obvious advantage of using fuzzy logic for traffic signal control is that it needs minimal computation resources. Similar to pre-timed and actuated control, it is much more computationally efficient than other adaptive methods. Another nice feature of fuzzy logic is that it can better represent the current system state. For example, as shown in Figure 13, a queue of length 3 will not be absolutely classified as {short} or {medium}. On the contrary, it belongs to both {short} and {medium}, each with a degree of 0.5. These fuzzy membership values may give the fuzzy traffic signal controller better generalization ability.

Some researchers also proposed rule-based and knowledge-based adaptive traffic signal control systems (*57,58,59*). For instance, Owen and Stallard (*59*) developed an adaptive traffic signal control method called Generalized Adaptive Signal Control Algorithm Project (GASCAP). GASCAP is a distributed control system, in which each intersection is controlled by a rule-based GASCAP controller. The GASCAP controller does not have fixed cycle, phase sequence, and phase durations. The coordination between adjacent intersections is realized through upstream detectors of each approach. These detectors provide information to each intersection controller on when and how many vehicles will arrive. GASCAP has three key components: queue estimation model, a set of rules for controlling uncongested traffic, and an algorithm for producing pre-timed plans for congested traffic. The major differences distinguish this rule-based method from other aforementioned adaptive traffic control methods are the rules for controlling uncongested traffic. GASCAP has five sets of rules as shown below. Each set of these rules calculates a priority value for each movement based on how many estimated vehicles need to be served from that particular movement.

1. **Demand Rules:** This set of rules tends to give green time to movements with the largest queue lengths. Phase sequence is not considered in making decisions using the demand rules.
2. **Progression Rules:** The purpose of progression rules is to coordinate signal timings of adjacent intersections. Progression rules give suggestions on signal states of each intersection in terms of projected traffic arrivals.
3. **Urgency Rules:** Urgency rules are used to detect saturation conditions on any of the approaches to an intersection. If any upstream detectors are on consecutively for at least 15 seconds, urgency rules will recommend the corresponding movements to be given green signal.
4. **Cooperative Rules:** Cooperative rules are employed mainly to address problems such as spillback. For two adjacent intersections, if one movement of the downstream intersection is experiencing spillback, movements of the upstream intersection aggravating the spillback will not be given green signal.
5. **Safety Rules:** Safety rules are used to ensure proper minimum green times, prevent conflicting movements from being given green signal at the same time, and so forth.

SUMMARY

This chapter reviewed pre-timed, actuated, and adaptive traffic signal control. The focus of the review was adaptive signal control systems or research prototypes including UTCS, SCOOT, SCAT, DYPIC, OPAC, RHODES, UTOPIA, PRODYN, ALLONS-D, and MDP&DP.

Pre-timed traffic signal control has fixed cycle length, phase sequence, and phase duration. It cannot adapt to short-term traffic flow dynamics and is only suitable for stable flow conditions. Actuated control can partially solve the problem with pre-timed control by introducing the concept of vehicle extension based on vehicle actuation information. However, actuated control still has many preset constraints and is not flexible enough.

Adaptive traffic control conceptually can better handle real time traffic flow fluctuations and significantly reduce control delay. There are two major types of adaptive traffic control systems. UTCS, SCOOT, and SCATS are typical examples of the first type of adaptive traffic control systems. The rest of the adaptive traffic signal control systems reviewed in this chapter can be generally classified as the second type. The first type of adaptive traffic signal control systems still has fixed cycle length, phase duration, phase sequence, and offset within short time periods. The control systems adaptively adjust these parameters based on real time or projected traffic conditions, and the parameters are updated every a few minutes to avoid disturbing normal traffic operations.

The second type of adaptive traffic control systems often model traffic control as a multi-stage problem or a MDP and solve it by dynamic programming or branch and bound. Fuzzy logic, rule-based methods, and knowledge-based methods have also been used in the second type of adaptive traffic control systems. For the second type of adaptive traffic signal control systems, there are no fixed cycle length, phase sequence and duration, and offset. All these parameters are determined in real time based on existing and projected traffic flow conditions. The second type of adaptive traffic control systems may not have the restrictions of cycle length, fixed phase sequence, phase duration, and offset, and has attracted considerable attention in recent years. However, this type of methods still has the following problems:

1. Under certain circumstances, the excessive computation requirement makes some systems based on dynamic programming not suitable for real time applications. Because of this, some approximate methods have to be used instead.
2. Both the multi-stage and MDP&DP modeling approaches require accurate traffic arrival information for the next one or two minutes to determine the

best control plans. This information is often affected by the control actions of adjacent intersections and is very difficult to obtain.

3. Although using fuzzy logic, rule-based, or knowledge-based methods has minimum computation time requirement, it is difficult to determine the optimal rules.

CHAPTER III

REINFORCEMENT LEARNING – THEORETIC BACKGROUND

WHY USING REINFORCEMENT LEARNING

Adaptive traffic signal control can better respond to short-term traffic fluctuations and has been the focus of recent traffic control studies. The review in Chapter II shows that dynamic programming has been adopted by many researchers for solving adaptive traffic signal control problems, as it is appropriate to model traffic signal control as a multi-stage decision problem or as a MDP that can be solved by dynamic programming. Also, dynamic programming can guarantee optimal solutions given accurate input information such as traffic arrivals and state-transition probabilities. However, in reality accurate traffic arrival information is very difficult to obtain, and the state-transition probabilities cannot be determined easily, either. More importantly for real-time applications, the computation time of dynamic programming could be a problem with multiple intersections and variable phasing schemes.

Fuzzy logic, rule-based, and knowledge-based methods have also been adopted for adaptive traffic signal control. These methods are generally referred to as rule-based adaptive traffic control methods. In contrast, adaptive control methods based on dynamic programming and branch and bound are called optimization-based adaptive traffic signal control. Compared to optimization-based adaptive traffic signal control, rule-based adaptive traffic control methods are much more computationally efficient. However, one problem with rule-based methods is the difficulty to determine the optimal control rules.

To overcome the aforementioned problems associated with optimization-based methods, especially those methods based on dynamic programming, a hybrid method based on reinforcement learning and neuro-fuzzy logic is proposed in this research. The new method is named as Neuro-Fuzzy Actor-Critic Reinforcement Learning (NFACRL). The intersection traffic control is still formulated as a MDP as did by Yu and Recker (*34*), but the NFACRL method is used in lieu of dynamic programming to solve for the optimal value function $V^*(s)$ in Equation (9) and to find the best control policy. There are two major advantages of using the NFACRL to solve MDP problems over using dynamic programming. First, the NFACRL does not require state-transition probabilities and traffic arrival predictions as inputs. It can learn the state-transition probabilities interactively from the system operations, and it can also learn the state-transition probabilities from simulations (*49*). Secondly, after the NFACRL is trained, it has the same low computational requirement as rule-based methods have. Thus, it is more suitable for real-time applications.

To better present the NFACRL method, a systematic introduction of the MDP and reinforcement learning is presented in this chapter, and the NFACRL method will be introduced in Chapter IV. The rest of this chapter is organized as follows: in the subsequent section, the MDP and various methods that can be used for solving MDP problems are discussed. These methods include dynamic programming, SARSA, Q-Learning, and Actor-Critic learning; following this discussion is a comprehensive review of existing applications of reinforcement learning to traffic control; after the review section is a section that analyzes the problems of the existing applications of reinforcement learning; and the final section summarizes this chapter.

REINFORCEMENT LEARNING

Reinforcement Learning Problems

Reinforcement learning is a sub-field of machine learning (*60*), and is different from supervised learning methods such as neural networks. For supervised learning methods, there must be a set of training pairs with input and expected output values. The training is to optimize the weights of neural networks such that the outputs from neural networks are as close to the expected outputs as possible. For some applications, it is extremely difficult to obtain such training pairs for supervised learning, and reinforcement learning is thus introduced to solve this problem. Reinforcement learning can learn directly from the interaction between the control agent and environment.

The concept of reinforcement learning is straightforward. As described in Chapter I, the control agent first senses the environment and identifies its current state. Based on the current state of the environment, the agent selects an action from the action set and applies it to the environment. The state of the environment affected by this action will change consequently. The control agent observes the state change and concludes a reward (penalty) value from the state change. This reward (penalty) value and the resulting new state are then used to update the control agent (*16,61*).

In reinforcement learning, the learner is often referred to as agent. Everything except for the agent is called environment. For different applications of reinforcement learning, the contents of agent and environment can be quite different. For traffic signal control, agent corresponds to the traffic signal controller and environment includes many factors such as the queue length of each approach, traffic arrivals, and current signal state. The interaction process between agent and environment is shown in Figure 14.

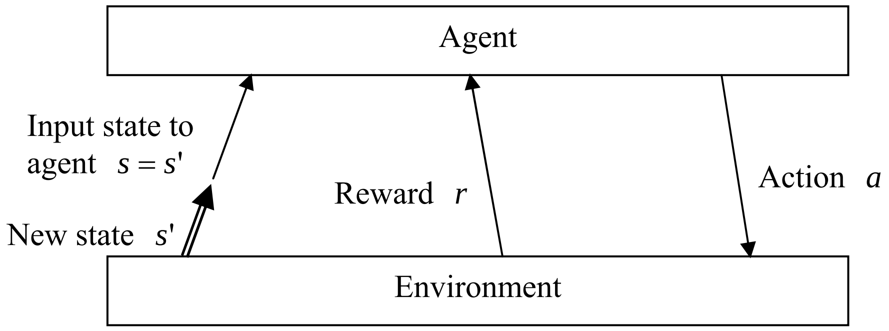


FIGURE 14 Agent and environment in reinforcement learning.

The interaction between agent and environment happens at any continuous time point, and theoretically the agent can make decisions at any time. For practical considerations, discrete time steps are often used. The following is a simple sample procedure to further explain how the interaction works at discrete time steps.

1. At time step *t*=0, observe the state of the environment $s_t \in S$. S is the collection of all possible states of the environment;
2. Based on $s_t \in S$, the agent chooses an action $a_t \in A(s_t)$. $A(s_t)$ is the collection of all available action choices for state s_t ;
3. Apply $a_t \in A(s_t)$ to the environment at time step t and observe new environment state $s_{t+1} \in S$ and reward r_{t+1} at time step $t+1$;
4. Use s_t, a_t, s_{t+1}, and r_{t+1} to update the agent; and
5. Let $t = t+1$ and go back to step 2.

At each time step, the agent chooses an action a_t based on the current environment state s_t. This mapping from states to actions is usually referred to as policy and represented by π . In the following subsections, why this procedure works and how the agent is updated will be explained.

Markov Property and Markov Decision Processes

Reinforcement learning method is built based on Markov property and MDP. A stochastic process is said to have the Markov property if it satisfies the following condition:

$$\Pr\{s_{t+1} = s' \mid s_h, \forall h \le t\} = \Pr\{s_{t+1} = s' \mid s_t\} \tag{10}$$

This equation suggests that the state of the stochastic process at time step $t+1$ only depends on the state of the process at time step *t*, not on any of the states of the process at time steps $h < t$. For reinforcement learning problems, the environment should satisfy the Markov property and the condition in Equation (11)

$$\Pr\{s_{t+1} = s', r_{t+1} = r \mid s_t, a_t, r_t, s_{t-1}, a_{t-1}, r_{t-1}, \ldots\} = \Pr\{s_{t+1} = s', r_{t+1} = r \mid s_t, a_t\} \tag{11}$$

If a stochastic process satisfies the Markov property, then it can be modeled as a MDP. A MDP is formally defined as a tuple (S, A, r, p) (*62,63*), where

1. S is the state space;
2. A is the action space;
3. r is a reward function, where $r_{ss'}^{a}$ represents the expected reward when the environment transfers from state s to state s' under the effect of action a at state s; and
4. p is a transition function, where $p_{ss'}^{a}$ represents the probability the environment will transfer from state s to state s' under the effect of action a at state s.

$r_{ss'}^{a}$ and $p_{ss'}^{a}$ can be expressed more precisely as the following: (*16*)

$$r_{ss'}^{a} = E\{r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s'\} \tag{12}$$

$$p_{ss'}^{a} = \Pr\{s_{t+1} = s' \mid s_t = s, a_t = a\} \tag{13}$$

In addition to state, action, reward function, and state-transition probability function, another important concept of MDP is value function, which includes state value function and action value function (*16*). State value function is a function representing how close each state is to the final (goal) state by following certain policy. In other words, it shows how good it is for the environment to be in each state under certain policy (*16*). The goal state is generally the control objective. For traffic signal control problems, the goal state is when all queues are minimized. The state value function following policy π is defined in Equation (14).

$$V_{\pi}(s) = \sum_{s'} p_{ss'}^{a} \left[r_{ss'}^{a} + \gamma V_{\pi}(s') \right] \tag{14}$$

where γ is a discount factor; a is the action decided by policy π when the environment is in state s; and $s' \in S$ are the resulted states after action a is taken when the environment is in state s. The action value function can be defined in a similarly way. If the current policy is π, the value of taking action a at state s is defined in Equation (15).

$$V_\pi(s,a) = \sum_{s'} p_{ss'}^a \left[r_{ss'}^a + \gamma V_\pi(s',a') \right] \tag{15}$$

where a' represents the action determined by policy π for state s'. In fact, Equations (14) and (15) are equivalent.

As the state function values of each state represent how close they are to the control goal (final state), solving a control problem modeled as MDP is equivalent to finding an optimal policy π^* (a mapping from states to actions) to minimize (or maximize, depending on the problem under study) the state function values for each state. With the optimal policy π^*, the following two equations hold.

$$V^*(s) = \max_{a \in A(s)} \sum_{s'} p_{ss'}^a \left[r_{ss'}^a + \gamma V^*(s') \right] \tag{16}$$

$$V^*(s,a) = \sum_{s'} p_{ss'}^a \left[r_{ss'}^a + \gamma \max_{a' \in A(s')} V^*(s',a') \right] \tag{17}$$

Equations (16) and (17) are two different forms of the Bellman optimality equation. They are often used in combination with dynamic programming to solve for the optimal state value or action value function. Once the optimal state value or action value function is obtained, the optimal control policy π^* can be readily determined by using Equation (18). For each state, one just needs to find the action that leads to the largest state value.

$$\pi^*(s) = \arg\max_{a \in A(s)} \sum_{s'} p_{ss'}^a \left[r_{ss'}^a + \gamma V^*(s') \right], \quad \forall s \tag{18}$$

where argmax means the argument of the maximum. It returns the action that maximizes the state value of s.

It can also be shown that Equation (14) is equivalent to Equation (19), which is the summation of discounted rewards (*16*).

$$V_\pi(s) = E_\pi\{R_t \mid s_t = s\} = E_\pi\left\{\sum_{k=1}^{\infty} \gamma^{k-1} r_{t+k} \mid s_t = s\right\} \tag{19}$$

where

R_t = summation of discounted rewards; and

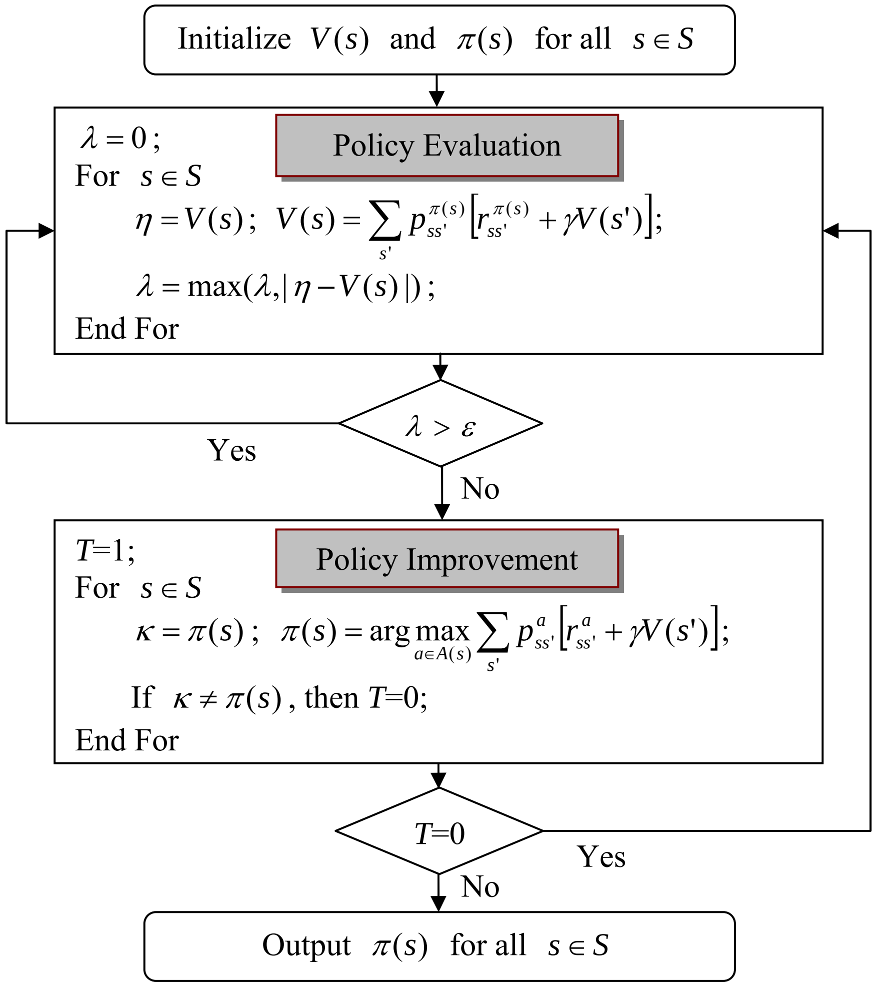
r_{t+k} = reward at the $(t+k)^{\text{th}}$ time step.

Thus, finding the optimal control policy $\pi*$ and state value function $V^*(s)$ actually is to maximize the summation of discounted rewards shown in Equation (19).

Dynamic Programming for MDP

There are mainly three methods that can be used to solve MDP problems: dynamic programming, Monte Carlo simulation, and reinforcement learning. In this section, dynamic programming method for MDP will be briefly discussed. The discussion serves as a basis for introducing the reinforcement learning method.

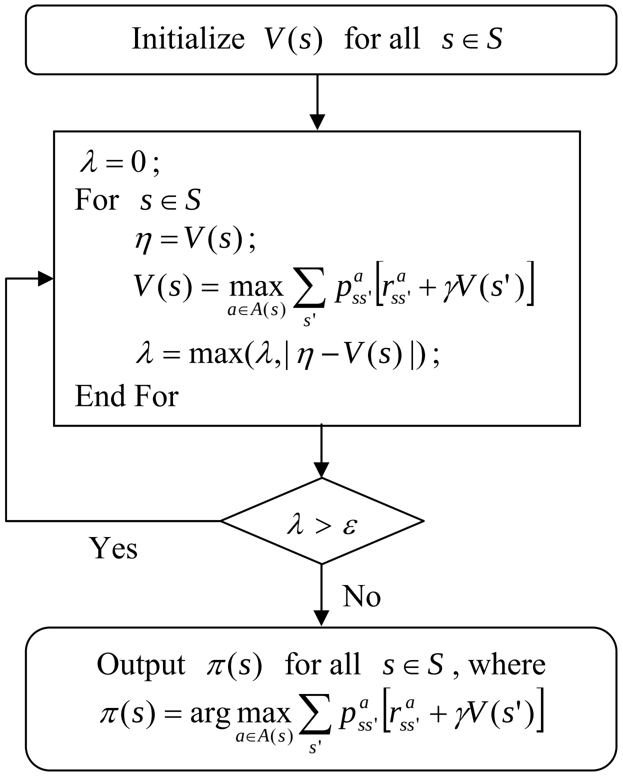
Two dynamic programming methods have been used to solve MDP problems: policy iteration and value iteration. Policy iteration has two components, which are policy evaluation and policy improvement. Policy evaluation and policy improvement are two iterative processes. Given certain policy π, policy evaluation tries to approximate the values of each state under this policy using Equation (14). The values of each state are the inputs to the policy improvement process. The purpose of policy improvement process is to adjust the policy according to the new state values, and the output of policy improvement is a new policy. Figure 15 shows how policy iteration is used to find the optimal policy for MDP problems (*16*), where $\pi(s)$ is the action decided by policy π for state *s*.



Initialize $V(s)$ and $\pi(s)$ for all $s \in S$
Policy Evaluation
$\lambda = 0$;
For $s \in S$
$\eta = V(s)$; $V(s) = \sum_{s'} p_{ss'}^{\pi(s)} \left[r_{ss'}^{\pi(s)} + \gamma V(s') \right]$;
$\lambda = \max(\lambda, |\eta - V(s)|)$;
End For
$\lambda > \varepsilon$
Yes
No
Policy Improvement
T=1;
For $s \in S$
$\kappa = \pi(s)$; $\pi(s) = \arg\max_{a \in A(s)} \sum_{s'} p_{ss'}^{a} \left[r_{ss'}^{a} + \gamma V(s') \right]$;
If $\kappa \neq \pi(s)$, then *T*=0;
End For
T=0
Yes
No
Output $\pi(s)$ for all $s \in S$

FIGURE 15 Policy iteration of dynamic programming.

Both policy evaluation and policy improvement need to visit each state multiple times and are computationally inefficient. Compared to policy iteration method, the value iteration method effectively integrates policy evaluation and policy improvement and has better computational efficiency. The value iteration method is illustrated in Figure 16.



Initialize $V(s)$ for all $s \in S$
$\lambda = 0$;
For $s \in S$
$\eta = V(s)$;
$V(s) = \max_{a \in A(s)} \sum_{s'} p_{ss'}^{a} \left[r_{ss'}^{a} + \gamma V(s') \right]$
$\lambda = \max(\lambda, |\eta - V(s)|)$;
End For
$\lambda > \varepsilon$
Yes
No
Output $\pi(s)$ for all $s \in S$, where
$\pi(s) = \arg\max_{a \in A(s)} \sum_{s'} p_{ss'}^{a} \left[r_{ss'}^{a} + \gamma V(s') \right]$

FIGURE 16 Value iteration of dynamic programming.

Although the policy iteration and value iteration methods are different, both of them can guarantee the optimal solutions if accurate knowledge of the probability $p_{ss'}^{a}$ is provided (*16*). For many practical problems such as adaptive traffic signal control (*34*), it is extremely difficult to obtain accurate estimation of state transition probabilities. In addition, the dynamic programming method may have considerably high computational requirements if the state space is large. It would be great if some methods can solve MDP problems without relying on the state transition probabilities and also have a low computational requirement. Fortunately, the reinforcement learning method can meet both requirements and will be introduced in the following subsections.

SARSA for MDP

SARSA Reinforcement Learning

SARSA is one of the three major types of reinforcement learning methods. The other two reinforcement learning methods are Q-Learning and Actor-Critic reinforcement learning. All these three reinforcement learning methods are based on a Temporal-Difference (TD) error (*16*). The TD error is calculated in terms of observed changes from the environment, and is used to update the state value function and the action value function. Unlike dynamic programming methods, reinforcement learning methods based on the TD error do not require the knowledge of state transition probabilities $p^a_{ss'}$.

Equation (20) shows a simple example of using the TD error to update state value function (*16*).

$$V_\pi(s_t) = V_\pi(s_t) + \phi\left[r_{t+1} + \gamma V_\pi(s_{t+1}) - V_\pi(s_t)\right] \qquad \textbf{(}20\textbf{)}$$

where

s_t = observed state of the environment at time step *t*;

s_{t+1} = observed state of the environment at time step *t*+1;

ϕ = learning rate;

$r_{t+1} + \gamma V_\pi(s_{t+1}) - V_\pi(s_t)$ = TD error;

r_{t+1} = observed reward at time step *t*+1; and

γ = discount factor.

Equation (20) can be rewritten in the form of Equation (21), which is used in SARSA to update the action value function.

$$V_\pi(s_t, a_t) = V_\pi(s_t, a_t) + \phi\left[r_{t+1} + \gamma V_\pi(s_{t+1}, a_{t+1}) - V_\pi(s_t, a_t)\right] \qquad \textbf{(}21\textbf{)}$$

where a_t and a_{t+1} are the actions determined by the current policy π for states s_t and s_{t+1}, respectively. By comparing Equations (21) and (16), one can see the similarity and difference between dynamic programming and the SARSA reinforcement learning. Both dynamic programming and the SARSA method use the one-step reward and the state or action value of the resulted state to update the state or action value of the current state. The major difference is that the dynamic programming method requires predefined state transition probabilities, while the SARSA method does not. The SARSA method introduces a learning rate ϕ and updates the action value by a linear combination of its current action value and the TD error. By using the SARSA method, $V_\pi(s,a)$ can converge to the optimal value $V^*(s,a)$ asymptotically (*16*). After the action value function has converged, the following Equation (22) is used to extract the optimal policy $\pi*$ from the action value function.

$$\pi^*(s) = \arg\max_{a \in A(s)} V^*(s,a) \tag{22}$$

Before using Equations (20) and (21), a reward function r_{t+1} has to be properly defined. The calculation of the reward function involves direct interactions between the control agent and the environment. This means that finding the optimal control policy requires implementation of the control system in real world or more likely through simulation. Using simulation as an example, the SARSA method is illustrated in Figure 17. This method can be better understood by taking a look at Figure 14, which shows the interaction between agent and environment.

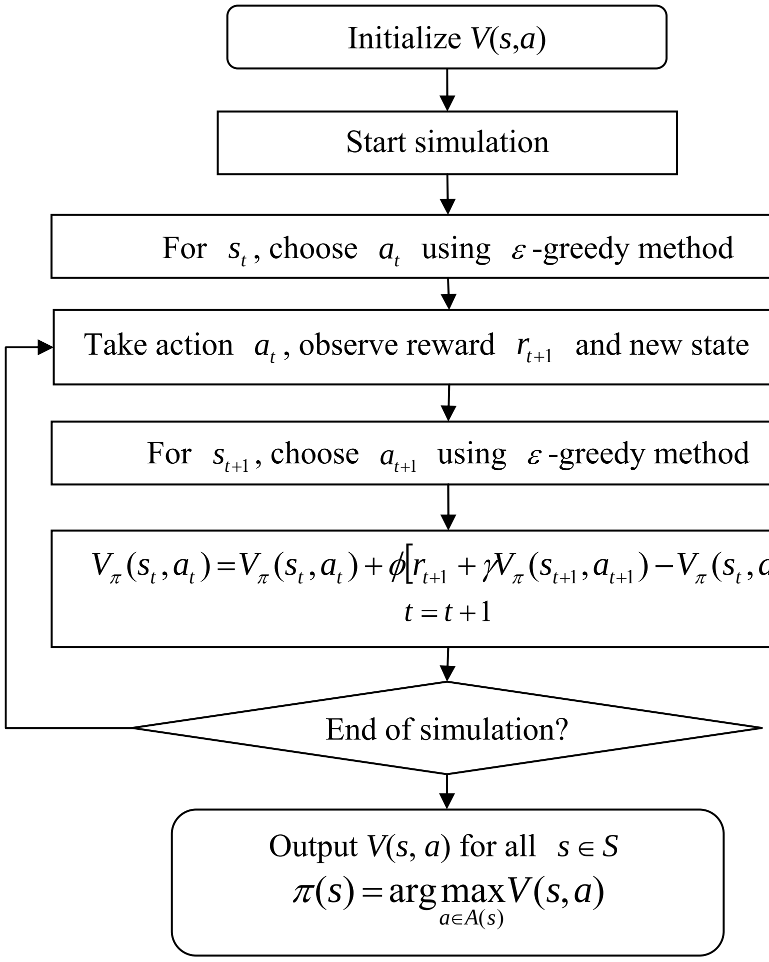

Initialize $V(s,a)$
Start simulation
For s_t, choose a_t using ε-greedy method
Take action a_t, observe reward r_{t+1} and new state s_{t+1}
For s_{t+1}, choose a_{t+1} using ε-greedy method
$V_\pi(s_t,a_t)=V_\pi(s_t,a_t)+\phi[r_{t+1}+\gamma V_\pi(s_{t+1},a_{t+1})-V_\pi(s_t,a_t)]$
$t=t+1$
End of simulation?
Output $V(s, a)$ for all $s \in S$
$\pi(s)=\arg\max_{a\in A(s)} V(s,a)$

FIGURE 17 SARSA for MDP.

Action Selection Methods

There are several methods that can be used for action selection given current state of the environment. These methods include greedy, ε-greedy and softmax action selection methods (*16*). The greedy method is the simplest one. For given state *s*, it always chooses an action with the largest action value *V*(*s*,*a*).

Sometimes two actions a_1 and a_2 may have approximately the same action value, and $V(s,a_1)$ is just slightly larger than $V(s,a_2)$. By using the greedy method, action a_1 will always be chosen. In fact, a_2 may be better than a_1, and $V(s,a_2)$ will be larger than $V(s,a_1)$ after one more value updating. To address this problem, an exploration strategy is incorporated into the greedy method and results in the ε-greedy selection method. For the ε-greedy selection method, actions with the largest action values are selected for most of the time. The remaining actions are selected with a small probability $\frac{\varepsilon}{|A(s)|}$. This method is described in Equation (23) more clearly (*16*).

$$\pi(s,a_i) = \begin{cases} 1-\varepsilon+\frac{\varepsilon}{|A(s)|} & \text{if } a_i = \arg\max_{a\in A(s)} V(s,a) \\ \frac{\varepsilon}{|A(s)|} & \text{otherwise} \end{cases} \qquad (23)$$

where

$\pi(s,a_i) =$ the probability that action a_i will be chosen for state *s*;

$\varepsilon =$ a small value; and

$|A(s)| =$ total number of possible actions for state *s*.

It can be seen that for the ε-greedy selection method, except for the action with the highest action value, all other actions are given the same probability to be chosen. Assuming actions a_1, a_2, and a_3 have the highest, second highest, and lowest action values, respectively, and action a_2 has a action value that is slightly less than the action value of action a_1. In terms of the ε-greedy selection method, action a_1 will have a large probability to be chosen and the other two actions will have the same small probability to be selected. However, it is intuitive that different actions should be given different probabilities commensurate with their action values. For this reason, the following softmax action selection was proposed (*16*).

$$\pi(s,a_i) = \frac{\exp\left[\frac{V(s,a_i)}{\tau}\right]}{\sum_{a\in A(s)} \exp\left[\frac{V(s,a)}{\tau}\right]} \tag{24}$$

where τ is a nonnegative parameter called *temperature* to be specified. When this *temperature* parameter is very large, all actions are given approximately the same probability to be chosen. When the *temperature* value is small, an action with a larger action value is given a greater chance to be selected, and the selection method tends to be greedy. Usually at the beginning of the learning process, large *temperature* is used. While at the end of the learning process, small *temperature* value should be chosen. Although the softmax is more sophisticated than the ε-greedy action selection method, determining the *temperature* parameter is cumbersome and there is no rigid rule to follow. In this study, the ε-greedy is used.

Q-Learning for MDP

Q-Learning is similar to SARSA. It uses Equation (25) to update action values.

$$V(s_t,a_t) = V(s_t,a_t) + \phi\left[r_{t+1} + \gamma \max_{a_{t+1}\in A(s_{t+1})} V(s_{t+1},a_{t+1}) - V(s_t,a_t)\right] \tag{25}$$

Equation (25) is slightly different from Equation (21), which is used by SARSA to update action values. SARSA is considered as an on-policy method while Q-Learning is an off-policy method. An on-policy method updates action values using the next step action determined by the current policy, and an off-policy method updates action values using the next step action with the largest action value. The following Figure 18 shows how the Q-Learning works.

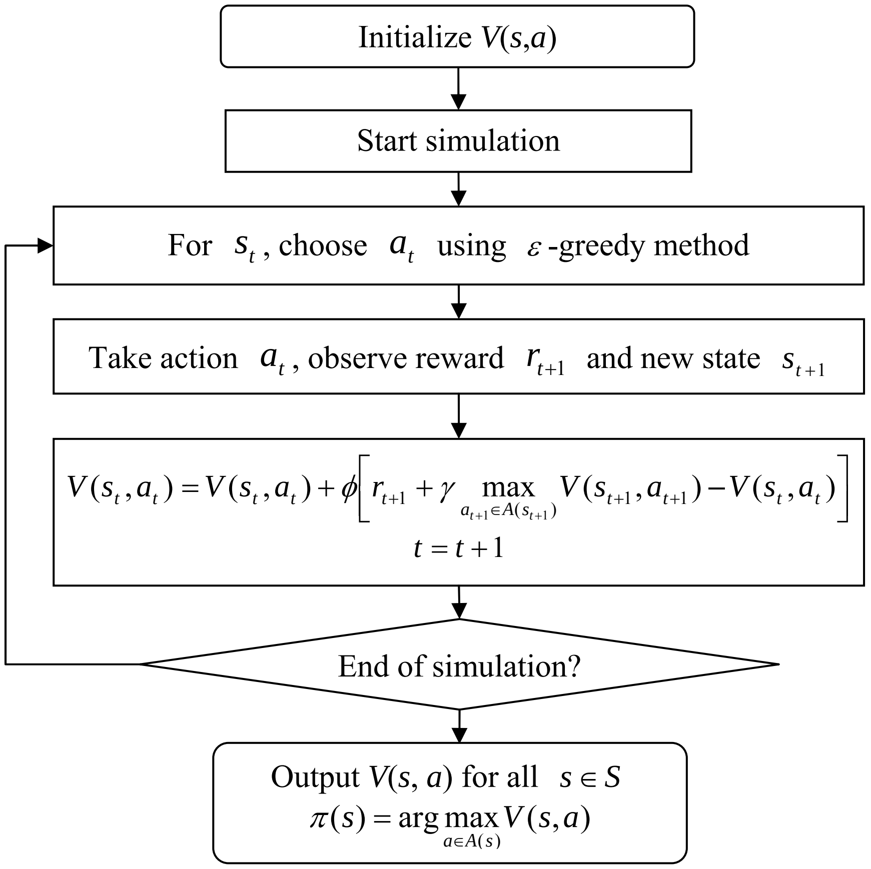

Initialize $V(s,a)$
Start simulation
For s_t, choose a_t using ε-greedy method
Take action a_t, observe reward r_{t+1} and new state s_{t+1}
$V(s_t,a_t)=V(s_t,a_t)+\phi\left[r_{t+1}+\gamma \max_{a_{t+1}\in A(s_{t+1})} V(s_{t+1},a_{t+1})-V(s_t,a_t)\right]$
$t=t+1$
End of simulation?
Output $V(s, a)$ for all $s\in S$
$\pi(s)=\arg\max_{a\in A(s)} V(s,a)$

FIGURE 18 Illustration of Q-Learning algorithm.

From Figures 18 and 19, the difference between SARSA (on-policy method) and Q-Learning (off-policy method) can be further manifested. A formal expression of the difference is "the distinguishing feature of on-policy methods is that they estimate the value of a policy while using it for control. In off-policy methods these two functions are separated. The policy used to generate behavior, called the behavior policy, may in fact be unrelated to the policy that is evaluated and improved, called the estimation policy. An advantage of this separation is that the estimation policy may be deterministic (e.g., greedy), while the behavior policy can continue to sample all possible actions" (*16*).

The Q-Learning results are stored in the action value function $V(s,a)$, which is in a table form as shown in Table 1. This table is often called Q-Table. Note that the number of actions for different states could be different. When the environment is in certain state, in terms of Equation (22), the best action is determined by finding the corresponding row in Table 1 for the current state and then locating the action with the highest action value in that row.

TABLE 1 Learning Results of Q-Learning Method

State #	Action #		
	1	2	…
1	8	9	…
2	11	6	…
…	…	…	…

For each cell in Table 1, its action value is updated by Equation (25) using the iteration process shown in Figure 18. To approximate the true action value, the corresponding cell needs to be visited as often as possible. However, when the state or action space is large, visiting each cell many times requires considerable computation time. This is often referred to as the curse of dimensionality problem. Thus, the traditional Q-Learning may not be directly applicable for problems with large state or action space. Another relevant problem with the traditional Q-Learning based on Q-Table is generalization. During the learning process some cells in Table 1 may only be visited one or two times even though their neighboring cells are visited many times. This may produce inaccurate action values for those less visited cells. When the environment happens to be in the corresponding states during actual application, it is possible that suboptimal actions will be chosen that may lead to poor control performance. In fact, it is reasonable to expect neighboring states to have similar action values. However, by using this traditional Q-Learning method, action values of neighboring cells cannot be used to update the action values of those less visited cells.

Actor-Critic Reinforcement Learning for MDP

Another well known reinforcement learning method is Actor-Critic Reinforcement Learning (ACRL) (*64,65,66,67*). ACRL has a more complicated structure than SARSA and Q-Learning. For SARSA and Q-Learning, optimal policies are stored in action value functions. After the optimal action value functions are obtained, Equation (22) is used to extract the optimal policies from the optimal action value functions. Storing optimal policies in action value functions is straightforward and easy to understand. For ACRL, the policy and state value functions are stored separately. Although this increases the complexity of the method and makes it difficult to analyze, the ACRL method does have two major advantages as discussed in (*16*).

For the ACRL method, the unit used to store policy is called *Actor*, and the unit used to store state value function is referred to as *Critic*. *Actor* and *Critic* can use different techniques such as neural networks and fuzzy logic (*68,69*) to store policy and state value function. To simplify the introduction of ACRL, a generic description of this method is provided here. The following figure has been used by many researchers to illustrate the architecture of ACRL (*16,65*).

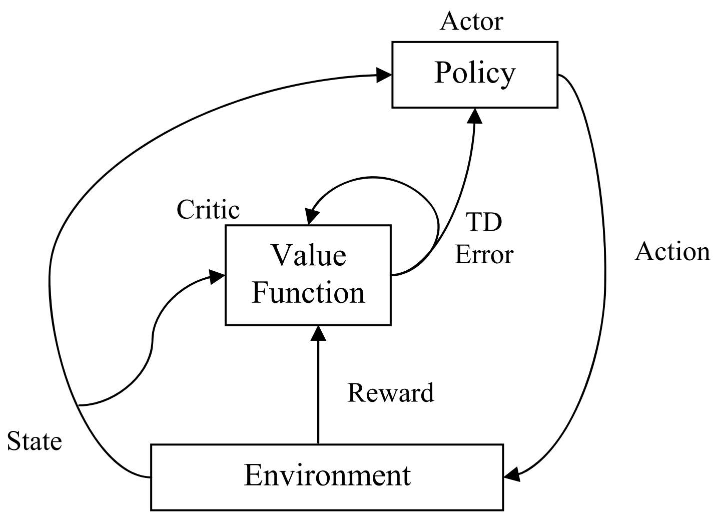


FIGURE 19 Architecture of Actor-Critic RL method (*16,65*).

At any decision point *t*, the *Actor* generates an action a_t based on the current environment state s_t. This action is then applied to the environment. Under the effect of action a_t, the environment may change accordingly. A reward value r_{t+1} and a new state s_{t+1} can be obtained. Also, a TD error is calculated using Equation (26).

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \tag{26}$$

For SARSA and Q-Learning, the TD error is defined based on action values and used for updating action value functions, since for them policies are stored in action value functions. For ACRL, the TD error is used to update both state value function and policy using Equations (27) and (28), respectively.

$$V(s_t) = V(s_t) + \alpha\delta_t \tag{27}$$

$$V(s_t, a_t) = V(s_t, a_t) + \beta\delta_t \tag{28}$$

where

$\alpha, \beta =$ step-size parameters; and

$V(s_t, a_t) =$ action value representing the preference to choose action a_t when the environment is in state s_t.

For an action, if its corresponding TD error is positive, then the preference of choosing this action should be reinforced. Otherwise, the preference of choosing this action should be decreased.

Comparison between Dynamic Programming and Reinforcement Learning

Both dynamic programming and reinforcement learning can be used for solving MDP problems. A major difference is that dynamic programming has to have an accurate model of the MDP problems. The reward and state transition probability functions need to be exactly known. While reinforcement learning methods such as SARSA, Q-Learning, and ACRL do not require perfect models of the MDP problems under study. They can implicitly learn the state transition probability functions and observe rewards from interactions between the agent and the environment. This property of reinforcement learning is very important and useful. For many practical problems that can be modeled as MDPs, it is usually very difficult to estimate the state transition probability and reward functions accurately. For instance, if an intersection has four approaches and eight movements as in Figure 1, assuming the queue lengths of each movement can be categorized into 5 classes, then there would be $5^8 \approx 39 \times 10^4$ possible states if one uses queue lengths as state variables. Finding the state transition probability function for this problem would be computationally very intensive. In practice, reinforcement learning would be a better choice for such type of problems.

REVIEW OF EXISTING INTERSECTION TRAFFIC CONTROL STUDIES USING REINFORCEMENT LEARNING

Traffic Control Using SARSA

Thorpe (*70*) conducted one of the pioneering studies on traffic signal control using reinforcement learning. In his study, SARSA was used to train each intersection control agent and the learning result was stored in a Q-Table similar to the one shown in Table 1. Each cell in the Q-Table corresponded to an action value $V(s,a)$ for a state-action pair (s,a). After the Q-Table was obtained, the intersection traffic control was simply to find and implement the best action $a^* = \arg\max_{a_t \in A(s_t)} V(s_t, a_t)$ for the current state s_t.

Thorpe tested the SARSA control method on a simple 4×4 grid traffic network with 16 intersections. Each intersection had four approaches and each approach had exactly one lane. The distance between any two intersections was 440 feet. Left-turn phase was not considered. Thus each intersection only had two through (right-turn movements were combined with the corresponding through movements) phases. Each of the 16 intersections was controlled by one agent, and there was no coordination explicitly considered. The test was carried out based on a self-developed simulation program. A 2-second yellow time and a 1-second all red time were used between phase switches for safety consideration. The control decision was made at a 1-second interval.

One key issue for reinforcement learning application is how to define the environment state. In Thorpe's study, four different methods were used to define the environment state, which were:

1. **Vehicle count representation:** vehicle count representation first summed up vehicle counts in each direction (east-west and north-south bounds) and then categorized them into 10 states. Since there were two control actions, the total number of states using the vehicle count representation method was 200.
2. **Fixed-distance representation:** in Thorpe's study, each approach of an intersection was 440 feet long, which was divided evenly into four segments. Each segment had two states: with and without vehicles on it. This representation method finally resulted in 512 states.
3. **Variable-distance representation:** this representation was almost the same as the fixed-distance representation except for how each approach was divided into segments. For the variable-distance method, each approach was divided into four segments at distances 50, 110, and 220 feet starting from the stop line. The total number of states was also 512.
4. **Count/duration representation:** the count/duration representation was based on the vehicle count representation. In this case, the vehicle counts were classified into 8 groups. Since there were two directions and two signal states (green or not green), the total number of states was 128. The action space in this representation was expanded to 16, which consisted of different minimum green times for each direction. Thus, the total number of state action pairs became 2048.

Another very important issue that affects reinforcement learning's performance is the definition of reward. Thorpe used two different definitions of reward. For the first definition, if at each decision point the environment state was not the goal state (goal state: all vehicles were cleared), then the value for the action taken at the previous decision point was updated by minus 1. For the second definition, the reward was defined in Equation (29).

$$r = \text{constant} + moved - stopped \tag{29}$$

where

$\text{constant} =$ a constant value that was set to -3 in Thorpe's study;

$moved =$ number of vehicles that have passed the intersection from approaches being given green signal; and

$stopped =$ number of vehicles that have been stopped due to a red signal in the last interval.

Thorpe tested the four state representation methods and two reward definitions based on computer simulation, and compared the SARSA control method with a number of other strategies such as greatest-volume strategy and pre-timed control. A greedy action selection method was used to choose actions for each state. The test was conducted under different traffic demand levels. The results showed that the SARSA method with count/duration state representation performed the best in terms of average travel time. For average stopped time, the SARSA method with fixed-distance and variable-distance representations performed better than the other methods tested. Thorpe also showed that for count/duration representation, the best reward definition was the first one, while for fixed-distance and variable-distance representations, the best reward definition was the second one.

There were a few problems not well considered in Thorpe's study. First, Thorpe used the greatest-volume and pre-timed control strategies as benchmarks for comparison with the SARSA control method. However, he did not describe clearly how the greatest-volume and pre-timed strategies were designed. Secondly, two-phase control without considering left-turn movements is very uncommon in practice unless may be for urban grids with one-way streets and left-turn restriction. Finally, Thorpe did not use any commonly-used simulation tools such as CORSIM (*71*) or VISSIM (*72*) for the comparison of different control strategies. These commonly-used microscopic traffic simulation packages should provide a more accurately simulated traffic environment and more rigorous performance measure calculation, consequently a more convincing results comparison. In spite of all these problems, Thorpe's study provided useful information for conducting further research on this topic.

Adaptive Traffic Signal Control Using Q-Learning

Abdulhai et al. (*73*) proposed a truly adaptive traffic signal control strategy based on Q-Learning. In their study, they discussed how to apply Q-Learning to both isolated intersection and arterial traffic control, and provided testing results for isolated intersection control. However, the authors did not provide testing results for arterial control, which are of primary interests to many traffic engineering researchers and practitioners.

For the application of Q-Learning to isolated intersection control, Abdulhai et al. considered an intersection without turning vehicles. Therefore, there were only two phases. Different from most of the adaptive traffic signal control methods reviewed in Chapter II, Abdulhai et al. considered a fixed cycle length for the isolated intersection control. Since the isolated intersection was controlled by a fixed cycle length strategy and there were just two phases, in each cycle there was only one decision to make, and the action set was whether to make the phase switch or not. In their study, Abdulhai et al. used total delay accumulated between two consecutive phase switch points as the reward. As for state variables, they used queue lengths on each approach and the elapsed time since last phase change. However, they did not make it clear how the states were defined in terms of queue lengths. If an approach can store up to 20 vehicles, then for this approach alone there could be 21 states in terms of the number of vehicles in the storage bay. When the state space is large, there could be a generalization problem. In their study, Abdulhai et al. used a technique called Cerebellar Model Articulation Controller (CMAC) for storing and generalizing the learned action value function.

The Q-Learning control was tested and compared with pre-timed control under different traffic flow patterns. The results showed that under uniform and constant-ratio flow conditions, Q-Learning control performed approximately the same as pre-timed control. While for variable traffic flow condition, Q-Learning control reduced average delay by more than 50% compared to pre-timed control. Although the results were very promising under the variable flow condition, the authors did not mention if the pre-timed control was optimized or not. In addition, this comparison was only for two-phase control. In practice, most intersections have four-phase signal operation.

In their study, Abdulhai et al. also proposed a general framework for arterial and network traffic control using Q-Learning. They suggested including queue information from adjacent intersections as the state variables for the current intersection control agent, to facilitate the coordination among these intersections. However, they acknowledged that this may considerably increase the state space and make the training time of Q-Learning a serious problem.

Signal Control Using Actor-Critic Reinforcement Learning

Bingham (*5*) proposed an isolated intersection traffic control strategy based on a Generalized Approximate Reasoning-based Intelligent Control (GARIC) algorithm developed by Berenji and Khedkar (*74*). The GARIC algorithm was essentially an Actor-Critic Reinforcement Learning (ACRL) method (*16*), in which there were two major components called action selection network (ASN) and action evaluation network (AEN). ASN was in the form of a fuzzy logic controller and it corresponded to the *Actor* in Figure 19. Given certain state of the environment, the ASN generated a continuous action output representing how long the current green signal should be extended. AEN was a fully connected feed-forward neural network used to approximate values of each state, and it corresponded to the *Critic* in Figure 19.

Bingham considered a very simple isolated intersection as the test bed. This intersection had two one-way streets. She used two state variables. The first state variable *APP* was the number of vehicles in the movement being given green signal. The second state variable *QUE* was the number of vehicles in the movement being given red signal. The *APP* and *QUE* were the inputs to both the *Actor* and *Critic*. The action output of the ACRL was a continuous value, which represented the amount of extension that should be given to the current green signal.

Recall the previous discussions on ACRL in this chapter. A TD error defined in Equation (26) is used to update the *Actor* and *Critic* at each learning step. This corresponds to updating the parameters of the fuzzy membership functions and the weights of the fully connected feed-forward neural network in Bingham's study. The TD error defined in Equation (26) has three components. In Bingham's study, r_{t+1} was defined as minus total vehicle delay between two consecutive decision points. $V(s_t)$ and $V(s_{t+1})$ were the outputs of the fully connected feed-forward neural network when the environment was in state s_t and s_{t+1}, respectively. The detailed updating algorithm is fairly complicated and can be found in (*68,75*).

Bingham compared the control performance of the original and the updated fuzzy logic controllers (ASN in her study) using a simulation program called HUTSIM. Three different traffic demand levels were tested, which were 300, 500, and 1000 vehicles per hour. The results showed that for traffic demand of 300 vehicles per hour, the original fuzzy logic controller performed better than the updated one; while for the other two traffic demand levels, the updated fuzzy logic controllers slightly outperformed the original one.

Other Signal Control Using Reinforcement Learning

Choy et al. (*76,77*), and Srinivasan and Choy (*78*) modeled a regional traffic signal control problem using reinforcement learning. In their studies, each intersection was controlled by a pre-timed controller. Reinforcement learning was used mainly to dynamically update the cycle lengths and other parameters of the pre-timed controllers in response to changing traffic flow conditions. The methods they proposed are similar to those investigated in the UTCS projects, and are not truly demand-responsive adaptive control recommended by Gartner (*8,35*).

PROBLEMS WITH THE EXISTING METHODS

Existing studies applying reinforcement learning to intersection traffic control provide useful information benefiting future research in this area. However, there are still several important issues that need to be investigated.

First, reinforcement learning is based on the MDP framework. In cases where the state space dimension is large, reinforcement learning will suffer from the curse of dimensionality (*42,34*) problem. For example, for an isolated four-approach intersection with eight movements (each through movement and its associated right-turn movement are combined as one movement) as shown in Figure 2, if one uses the numbers of queuing vehicles of each movement as state variables and the maximum queue length for each movement is five vehicles, then the total number of states is $6^8 \approx 1.68 \times 10^6$, which means the Q-Table will have around 1.68×10^6 rows. If some continuous state variables such as length of green time are introduced, the state space could theoretically be infinite. The huge state space first makes the storage of the Q-Table very difficult. It also requires demanding computation time to fill the Q-Table accurately (*16*). Yu and Recker (*34*) tried to solve this difficulty by setting a threshold for each movement, such that each movement only had two states, namely congested and non-congested. This method significantly reduced the total number of states to $2^8 = 256$, however, an obvious problem incurred is that this probably will degrade the control performance.

Secondly, the coordination of different control agents has not been adequately investigated in previous studies. Bingham (*5*) and Abdulhai et al. (*73*) only reported results for isolated intersections. Although Thorpe (*70*) did apply his proposed method to a 4×4 network, coordination was not explicitly considered or discussed in his study.

Thirdly, most previous studies used isolated intersections and networks with very simple structures for testing. Thorpe (*70*) tested his reinforcement learning control method on a network without considering left-turn phases. Abdulhai et al. (*73*) evaluated their truly adaptive reinforcement learning traffic control method on an isolated intersection without turning vehicles. Bingham (*5*) evaluated an ACRL traffic controller on an isolated intersection of two one-way streets. In all these studies, there were only two phases to be considered for each intersection. In reality, most intersections have eight movements and are typically controlled using three or four phases.

Finally, most of the previous studies did not use a commonly-accepted traffic simulation platform for algorithm evaluations. Thorpe (*70*) used a simulation program developed by himself. Bingham (*5*) used the HUTSIM developed by the Helsinki University of Technology. In the study by Abdulhai et al. (*73*), they did not mention which simulation program was used.

SUMMARY

This chapter focused on introducing reinforcement learning methods and their recent applications in intersection traffic control. Markov property and MDP were first discussed, which were the modeling bases of reinforcement learning methods. After that, dynamic programming and three reinforcement learning methods were introduced and compared. The three reinforcement learning methods discussed were SARSA, Q-Learning, and ACRL. Both dynamic programming and reinforcement learning can be used for solving MDP problems, and dynamic programming has been applied to solve adaptive traffic signal control modeled as a MDP problem (*34*). Comparison in this chapter showed that reinforcement learning has certain advantages over dynamic programming for intersection traffic control problems based on MDP framework. This is mainly because reinforcement learning does not need to have perfect models of the systems to be controlled, and can implicitly learn the state transition probability function from the interactions between environments and agents.

Some recent applications of reinforcement learning to traffic signal control were reviewed. Several problems with these existing applications were identified and discussed. Those problems include the following:

1. Only very simple two-phase signal control was considered.
2. The curse of dimensionality problem was not well addressed in most previous studies.
3. Coordination among intersection control agents was not explicitly considered. And
4. No comprehensive tests have been conducted using commonly-accepted microscopic traffic simulation tools.

Despite of these limitations, the existing studies provide much useful information for this dissertation and future research in this area. In the next chapter, a new reinforcement learning signal control method based on neural networks and fuzzy logic will be developed, and details about how to apply this new signal control method to both intersection and arterial control are also presented.

CHAPTER IV

DEVELOPMENT OF AN ARTERIAL TRAFFIC SIGNAL CONTROL SYSTEM BASED ON NEURAL FUZZY ACTOR-CRITIC REINFORCEMENT LEARNING

INTRODUCTION

In Chapters II and III, a comprehensive review of intersection traffic signal control, reinforcement learning, and reinforcement learning for adaptive traffic signal control is presented. The review shows that adaptive traffic signal control is conceptually more efficient than pre-timed and actuated control. Many adaptive traffic signal control methods have been developed. Compared to traditional adaptive traffic control methods such as OPAC and RHODES, modeling adaptive traffic control as a MDP problem can better account for the uncertainty in state transition by introducing a state transition probability matrix. The review also shows the advantages of using reinforcement learning over dynamic programming for adaptive intersection traffic control modeled as a MDP problem. In the meantime, problems with reinforcement learning and its applications to adaptive traffic control are also discussed in details. To address these problems, in this chapter a Neuro-Fuzzy Actor-Critic Reinforcement Learning (NFACRL) method is developed for both intersection and arterial traffic control. The NFACRL method is designed to consider more practical traffic signal control problems with more than two phases and left-turn movements. Compared to the traditional reinforcement learning methods such as Q-Learning, the NFACRL method can better handle the curse of dimensionality and generalization problems. Coordination of intersection traffic control agents is also taken into account. In addition, the NFACRL method will be compared with optimized pre-timed and actuated control strategies using a commonly-accepted microscopic traffic simulation tool.

In the following sections, a concise description of fuzzy logic control and neural networks is presented first. The NFACRL method is then introduced and two implementation schemes for isolated intersection traffic control using the NFACRL are proposed. Following that are the discussions of coordination strategies and the development of an arterial adaptive traffic control system using the NFACRL.

FUZZY LOGIC CONTROL AND NEURAL NETWORKS

Fuzzy Logic Control

Fuzzy Sets and Fuzzy State Representation

Before introducing fuzzy sets and fuzzy state representation, an example of discrete state representation is presented. Discrete state representation has been used in several previous studies (*34,70,73*). It uses crisp boundaries to partition observed state values into different categories. For example, if a set of boundary values shown in Table 2 is used for partitioning state values, then a queue of 6 vehicles will be classified as "Uncongested", while a queue of 7 vehicles will be classified as "Congested". Although the difference between queues of 6 and 7 vehicles is almost negligible, these two queues belong to distinctly different states according to the discrete state representation. Also for queues of 1 vehicle and 6 vehicles, although a queue of 6 vehicles is six times as long as a queue of only 1 vehicle, they all belong to state "Uncongested" and are treated the same. Obviously, it is problematic to use such partition method for categorizing input state values. One way to address this problem is to use smaller partition intervals, but this will considerably increase the number of states and make the reinforcement learning problem intractable.

TABLE 2 Threshold Values for Each Category

	Uncongested	Congested
Threshold values	<=6 vehicles	>=7 vehicles

This problem can be better solved by using fuzzy sets and fuzzy set representation. In the fuzzy set representation, each category in Table 2 will have a membership function associated with it. For given queue length, there are two membership function values showing the degrees that the given queue belongs to each category. Using membership function values can avoid classifying a queue into a category absolutely. To explain how this works, first the concept of fuzzy sets is formally defined below (*55*).

$$A = \{(x_i, \mu_A(x_i)) \mid x_i \in X\} \tag{30}$$

where

A = fuzzy set;

X = a collection of values, which can be discrete or continuous and is often referred to as universe of discourse;

x_i = values that belong to set *X*; and

$\mu_A(x_i)$ = membership function for fuzzy set *A*. Its values are always between 0 and 1 and represent the degrees that each x_i belongs to the current fuzzy set.

There are many types of membership functions, including Triangular, Trapezoidal, and Gaussian membership functions as defined in Equations (31) through (33). Examples of these three types of fuzzy membership functions are also shown in Figure 20.

Triangular membership function (*55*)

$$\mu_A(x) = \begin{cases} (x-a)/(b-a) & x \in [a,b] \\ (c-x)/(c-b) & x \in [b,c] \\ 0 & \text{else} \end{cases}, \text{ where } a < b < c \tag{31}$$

Trapezoidal membership function (*55*)

$$\mu_A(x) = \begin{cases} (x-a)/(b-a) & x \in [a,b] \\ 1 & x \in [b,c] \\ (d-x)/(d-c) & x \in [c,d] \\ 0 & \text{else} \end{cases}, \quad \text{where} \quad a < b \le c < d \tag{32}$$

Gaussian membership function (55)

$$\mu_A(x) = \exp\left\{ -\frac{(x-a)^2}{2\sigma^2} \right\} \tag{33}$$

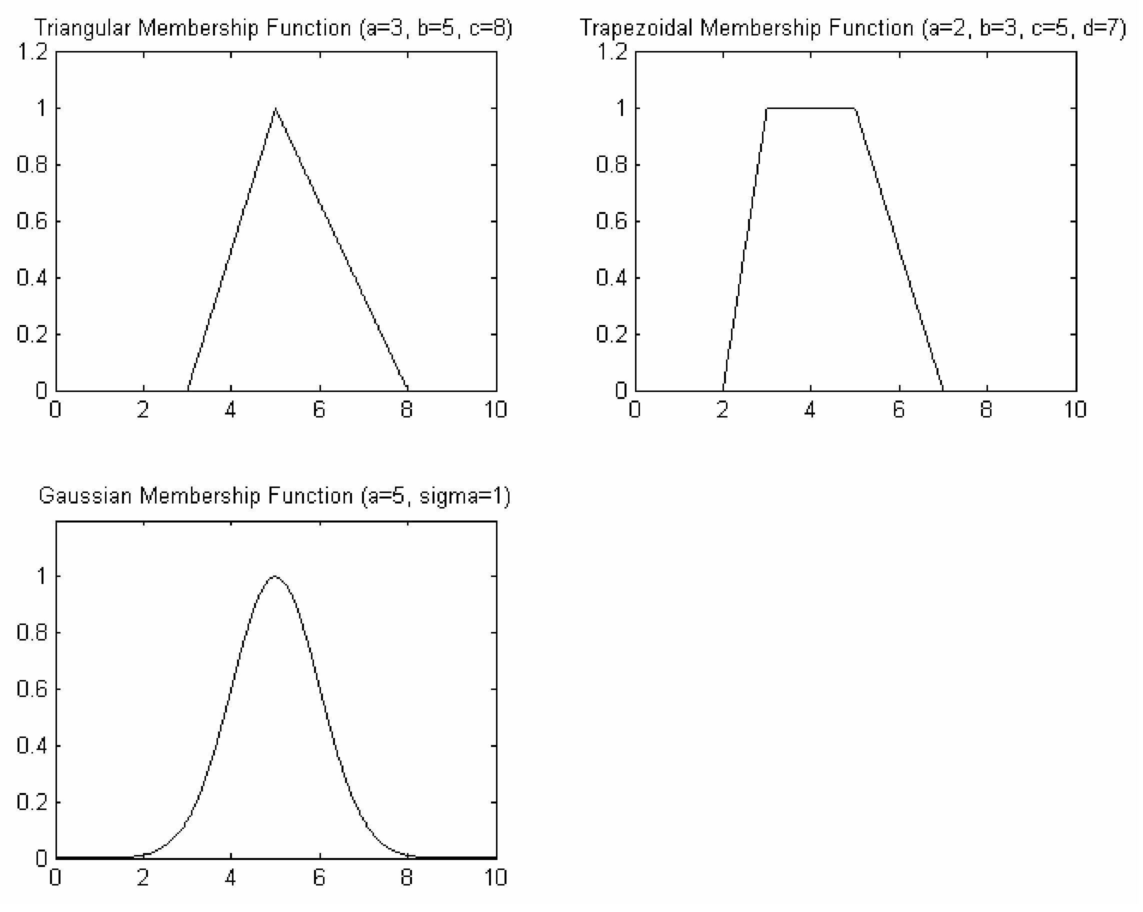


FIGURE 20 Fuzzy membership function examples.

A number of operations are defined for fuzzy sets, including union and intersection. The union of fuzzy sets *A* and *B* is denoted as $A \cup B$, and the membership function for the resulted new fuzzy set is defined as (*55,79*)

$$\mu_{A \cup B}(x) = \mu_A(x) \vee \mu_B(x) = \max\{\mu_A(x), \mu_B(x)\} \tag{34}$$

The intersection of fuzzy sets *A* and *B* is denoted as $A \cap B$, for which the new membership function is defined as (*55,79*)

$$\mu_{A \cap B}(x) = \mu_A(x) \wedge \mu_B(x) = \min\{\mu_A(x), \mu_B(x)\} \tag{35}$$

If the fuzzy sets and fuzzy set representation is used to classify a queue into two categories as shown in Table 2, then x_i represents the queue length; *X* denotes all possible discrete queue length values; and there are two fuzzy sets *U* and *C*, which stand for "Uncongested" and "Congested" conditions, respectively. For fuzzy set *C*, if the membership function is a Triangular function with parameters *a*=5, *b*=7, and *c*=9, then given queue lengths 6 and 7, their corresponding membership function values are 0.5 and 1, respectively. Compared with the results from the discrete state representation presented at the beginning of this section, the results from the fuzzy set representation are more rational. Moreover, the number of states can be kept within a reasonable range. The process of applying the fuzzy set representation and calculating the membership function values is often called fuzzification.

Fuzzy Rules and Reasoning

Using fuzzy sets, state variables can be written in the following linguistic term

- ♦ Current Queue Length is {*A*}

where "Current Queue Length" is a state variable and also called a linguistic variable in this case. *A* is a linguistic value corresponding to a fuzzy set that could denote "Uncongested" or "Congested" condition. For each observed value of "Current Queue Length", there is a fuzzy membership function value associated with the linguistic term "Current Queue Length is {*A*}", and this membership function value is also called degree of compatibility. Action variables can also be expressed in the same way by using linguistic term. For instance,

- ♦ Green Time Extension is {*G*}

where "Green Time Extension" is an action variable (also a linguistic variable) and *G* is a linguistic value corresponding to a fuzzy set that could denote "Short" or "Long".

Using linguistic terms, traffic control can be realized using fuzzy rules that consist of linguistic terms as in the following examples:

- ♦ **IF** Current Queue Length (q) is {Short} **AND** Arrival (a) is {Low} **AND** Conflicting Queue Length (c) is {Medium}, **THEN** Extension (e) is {Short}
- ♦ **IF** Current Queue Length (q) is {Medium} **AND** Arrival (a) is {High} **AND** Conflicting Queue Length (c) is {Short}, **THEN** Extension (e) is {Long}

A fuzzy rule usually has two components: antecedent and consequence. In the first fuzzy rule presented above, linguistic terms "Current Queue Length (q) is {Short}", "Arrival (a) is {Low}", and "Conflicting Queue Length (c) is {Medium}" are antecedents, while the last linguistic term "Extension (e) is {Short}" is a consequence (*55*). Each antecedent or consequence has a degree of compatibility, which in fact is the fuzzy membership function value for the corresponding linguistic term.

Each fuzzy rule has a numerical value associated with it. This value is called firing strength. Firing strength is calculated based on the degrees of compatibility of antecedents. For the first fuzzy rule in the previous paragraph, the degrees of compatibility are $\mu_{Short}(q)$, $\mu_{Low}(a)$, and $\mu_{Medium}(c)$. There are basically two methods to calculate the firing strength (*55*). The first one is to calculate it as the intersection of the degrees of compatibility of all antecedents, which is in Equation (36).

$$FS_{Rule\,1} = \mu_{Short}(q) \wedge \mu_{Low}(a) \wedge \mu_{Medium}(c) \tag{36}$$

The other method is to calculate it as the product of the degrees of compatibility of all antecedents (*68*) as shown in Equation (37)

$$FS_{Rule\,1} = \mu_{Short}(q) \times \mu_{Low}(a) \times \mu_{Medium}(c) \tag{37}$$

Firing strength is used for calculating the output of a fuzzy rule, and the output is an induced consequent fuzzy set. The entire process from fuzzy rules to the induced consequent fuzzy set is called fuzzy reasoning and is illustrated in Figure 21, in which there are two fuzzy rules. The first step of fuzzy reasoning is to calculate the degree of compatibility of each antecedent. This step is also called fuzzification. Based on these degrees of compatibility, the second step is to calculate the firing strength of each fuzzy rule. As discussed before, there are mainly two different methods for calculating the firing strength. For the example in Figure 21, Equation (36) is used to calculate firing strengths. Each fuzzy rule has a consequence. The consequences in this example are two fuzzy sets: $\mu_{Short}(e)$ and $\mu_{Long}(e)$. The calculated firing strengths are then applied to these two consequences to obtain induced consequent fuzzy sets. The two induced consequent fuzzy sets are represented by the shaded areas in Figure 21. For fuzzy reasoning problems with two or more fuzzy rules, a union operation in Equation (34) is usually used to merge all induced consequent fuzzy sets to obtain a combined induced consequent fuzzy set. For the example shown in Figure 21 with two fuzzy rules, the combined consequent fuzzy set is $\mu_{Extension}(e)$.

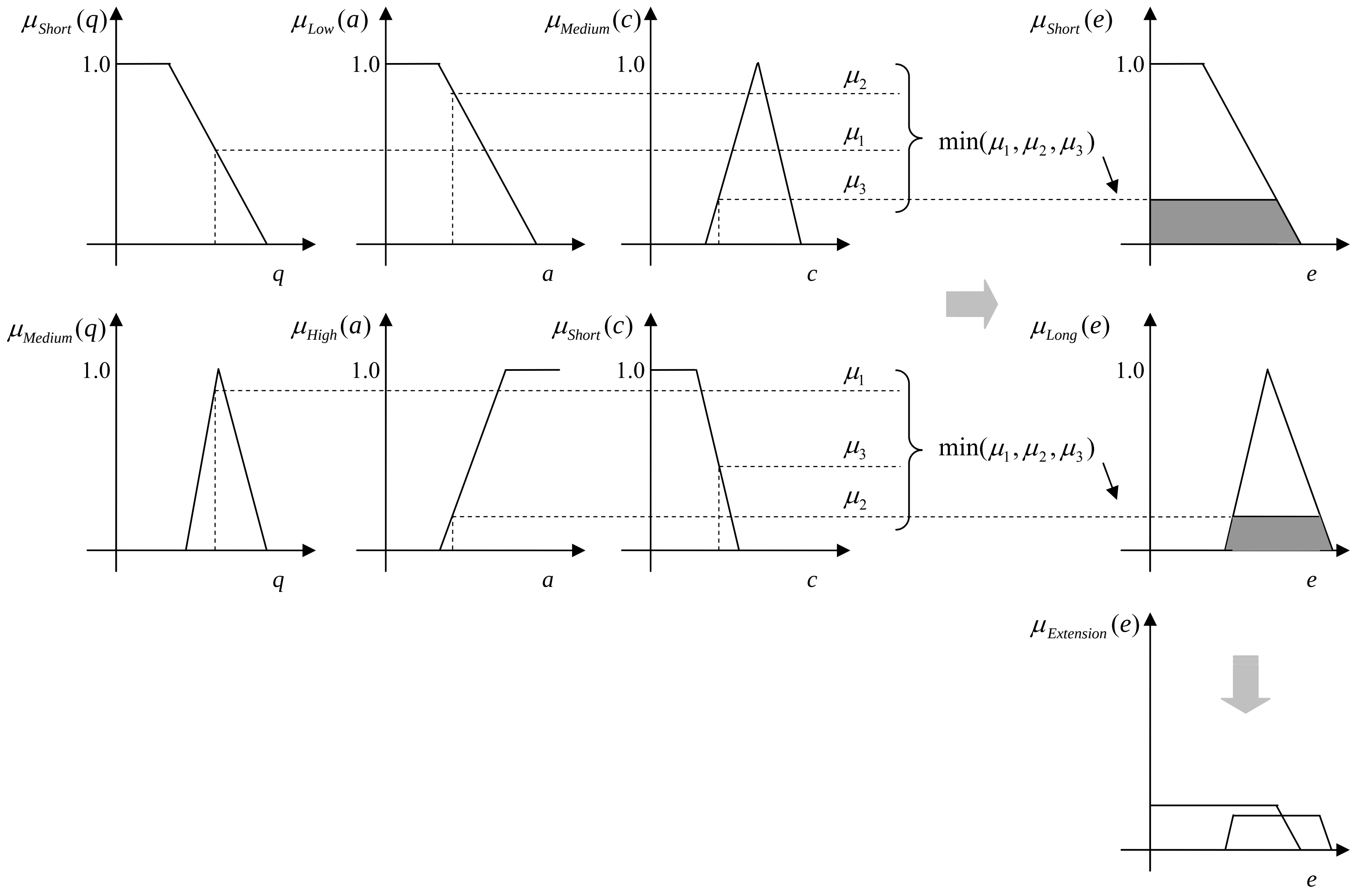


FIGURE 21 Example of fuzzy reasoning.

Fuzzy Logic Controller

A typical fuzzy logic controller has five major components, which are shown in Figure 22. The fuzzification process is to obtain the degrees of compatibility of each antecedent in fuzzy rules. The fuzzy inference component includes fuzzy rules and fuzzy reasoning, which are discussed in the previous section. As shown in Figure 21, the result from fuzzy inference is a combined induced consequent fuzzy set. To apply fuzzy logic controllers to practical control problems, a meaningful and crisp value usually needs to be obtained from the combined induced consequent fuzzy set.

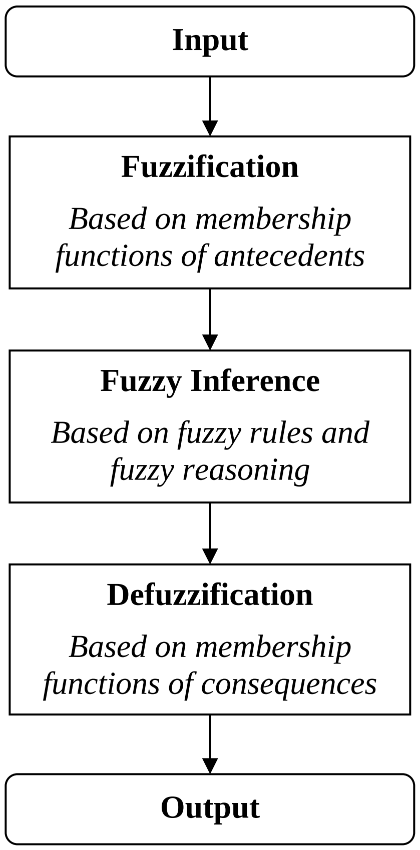

Input
Fuzzification
Based on membership functions of antecedents
Fuzzy Inference
Based on fuzzy rules and fuzzy reasoning
Defuzzification
Based on membership functions of consequences
Output

FIGURE 22 Structure of a typical fuzzy logic controller.

The process of obtaining a crisp value from the output of fuzzy inference, a combined induced consequent fuzzy set, is called defuzzification. A number of methods are available for this purpose, including Centroid of Area (COA), Bisector of Area (BOA), Mean of Max (MOM), Smallest of Max (SOM), and Largest of Max (LOM) (*55*). Using the combined induced consequent fuzzy set shown in Figure 21 as an example, the COA method is defined in Equation (38).

$$Output_{COA} = \frac{\int_{e} \mu_{Extension}(e) \cdot e de}{\int_{e} \mu_{Extension}(e) de} \tag{38}$$

Compared with other defuzzification methods such as SOM and LOM, the COA method can give a more reasonable output, especially for fuzzy sets with irregular shapes. However, the COA method requires more computation time as integrals are involved. After defuzzification, a crisp value can be obtained and used for practical applications. For the example in Figure 21, the output crisp value represents the amount of green time extension that should be given to the current green phase.

There are several different types of fuzzy logic controllers. The one just introduced is called Mamdani fuzzy logic controller. Other well-known fuzzy logic controllers include Sugeno and Tsukamoto fuzzy logic controllers. Detailed information about them can be found in (*55*).

Neural Networks

In traditional reinforcement learning methods such as Q-Learning, a Q-Table is usually used for storing the learned control policy in the form of action values. As discussed in Chapter III, this Q-Table method has certain limitations when the state or action space is large. In recent studies, neural networks are often used instead of the Q-Table in reinforcement learning for storing learned policies (*16,74,75*), to improve generalization ability and better handle the curse of dimensionality problem. In the proposed NFACRL method, neural networks are combined with fuzzy logic control to approximate the best control policy. For better understanding of the proposed NFACRL method, a feed-forward back-propagation neural network is briefly described here.

Figure 23 shows the structure of a typical feed-forward back-propagation neural network. This network has three layers. The first layer is the input layer that takes inputs and sends them to the second layer. Each node in the first layer represents an input variable. The second layer is the hidden layer that consists of a number of hidden neurons, and each hidden neuron has a transfer function. The input to each transfer function is the summation of the weighted outputs from the first layer. The third layer is the output layer. In this example, it consists of only one neuron. In fact, there could be more than one neuron in the output layer depending on the problems to be solved. Similar to the hidden layer, the neuron in the output layer also has a transfer function. Its input is the summation of the weighted outputs from the hidden layer. The output of this transfer function is also the output of this neural network. In addition to these neurons, there are a number of weights and biases in the network. Before a neural network can be used to solve problems, these weights and biases have to be calibrated through a process called training.

In the sample network shown in Figure 23. The transfer functions for the hidden layer are chosen to be a Tangent (tanh) function and a linear function is used as the transfer function for the output layer. Assume there are n pairs of observed input and output data $\{(x_1, y_1),...,(x_i, y_i),...,(x_n, y_n)\}$. The prediction output $\hat{y}_i$ using this sample neural network is given by Equation (39).

$$\hat{y}_i = f(x_i, \psi) = b2 + \sum_{j=1}^{M}\left[w2(j) * \tanh\left(\sum_{k=1}^{P} w1(j,k) * x_{ik} + b1(j) \right)\right] \tag{39}$$

where

P = number of input neurons;

M = number of hidden neurons;

$b1(j)$ and $b2$ = biases;

$w2(j)$ = weights connecting hidden layer and output layer;

$w1(j,k)$ = weights connecting input layer and hidden layer;

x_{ik} = the k^{th} element of the i^{th} input;

$x_i = [x_{i1},..., x_{ik},..., x_{iP}]$, the i^{th} input;

ψ = a vector contains all the network parameters ($b1(j)$, $b2$, $w1(j,k)$, and $w2(k)$);

$i = 1, 2,\ldots, n$; $j = 1, 2,\ldots, M$; and $k = 1, 2,\ldots, P$.

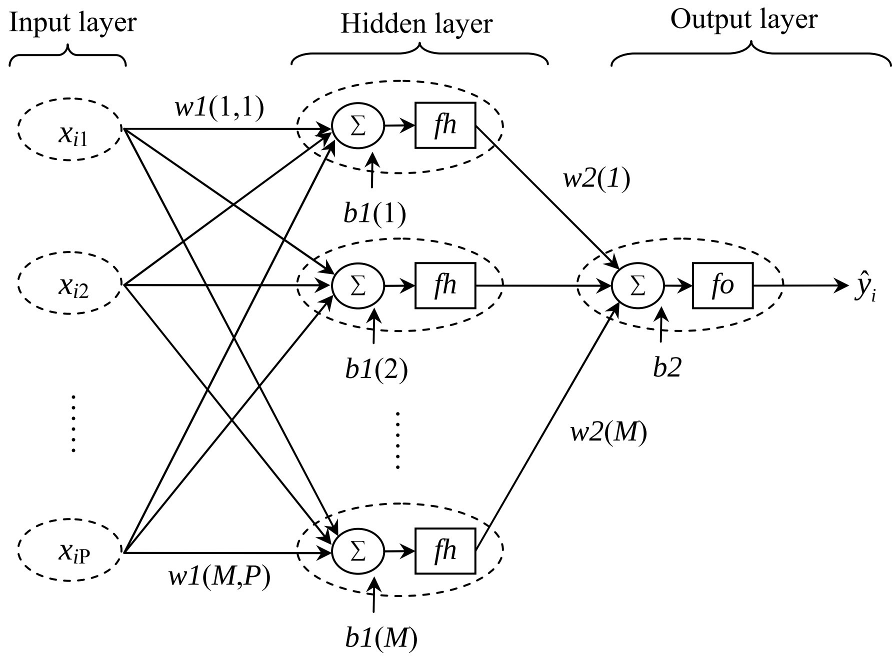


FIGURE 23 A typical feed-forward back-propagation neural network.

The goal of training the sample neural network is to minimize the error term defined in Equation (40) by fine tuning weights and biases. An often used method for minimizing the error term is the back-propagation training, which is detailed in (*55,80*)

$$E = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \quad (40)$$

When applying neural networks to store the learned policy of a traffic control problem, the input to the network shown in Figure 23 could be the queue lengths and the output could be the amount of green time extension. Depending on the problems under study, neural networks can have multiple output units and each of them stands for a specific control action. The value of each output unit represents the preference that the corresponding action should be chosen.

NEURO-FUZZY ACTOR-CRITIC REINFORCEMENT LEARNING (NFACRL)

Introduction

Most existing reinforcement learning traffic control studies are for oversimplified two-phase controlled intersections, and there are only two control actions. However, intersections in real world usually have four approaches and eight movement combinations as shown in Figure 2. There could be as many as eight control actions, which are shown in Figure 24. Due to this significant difference, the methods developed in most existing reinforcement learning traffic control studies (*5,70,73*) are not directly applicable to controlling a realistic intersection shown in Figure 2 for the following two major reasons.

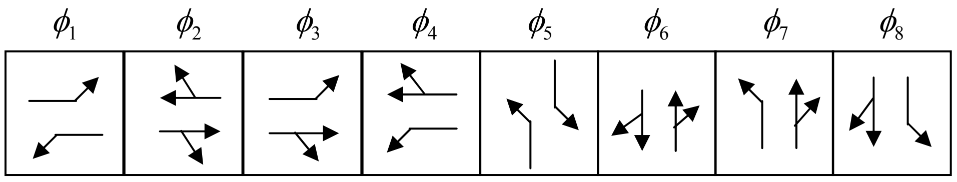


FIGURE 24 Possible control actions for a four-approach intersection.

First, for controlling realistic intersections with more than two phases, phase sequence is expected to have significant effects on traffic control performance, and it is necessary to examine the possibility of applying reinforcement learning to phase sequence selection. While in most previous reinforcement learning traffic control studies

(*5,70,73*), there were only two actions and phase sequence optimization was not considered at all. In these studies, reinforcement learning was used mainly for determining when to switch phases.

Secondly, when there are eight movement combinations, the state space will be very large if the discrete state representation method is used. Large input state space makes the reinforcement learning process very slow and also brings up the generalization problem (*16*). Thorpe (*70*) did not consider large state space problem in his study. Abdulhai et al. (*73*) used Cerebellar Model Articulation Controller (CMAC) to store the Q-Table. However, it is not clear whether this method can handle large state space or not. Although fuzzy logic adopted by Bingham (*5*) can help solve the large state space problem, determining the fuzzy rules is difficult, especially when there are many state variables. To solve the aforementioned two major problems, the NFACRL method is introduced.

NFACRL Structure

The NFACRL method developed by Jouffe (*81,82*) is also an actor-critic type of reinforcement learning, but it is different from the GARIC method used by Bingham (*5*). The NFACRL method takes the form of neural networks and also incorporates fuzzy logic control into it. The structure of the NFACRL method is shown in Figure 25. The symbols used in Figure 25 are described below.

$S_i =$ the i^{th} input variable;

$K =$ the total number of input variables;

$NM_i =$ the number of fuzzy sets or membership functions for the i^{th} input variable;

$M_i^{a(i)} =$ the $a(i)^{th}$ fuzzy set or membership function for the i^{th} input variable;

$R_j =$ the j^{th} fuzzy rule;

$N =$ the total number of nodes in the third layer;

$\lambda^j =$ the weight connecting the j^{th} fuzzy rule and the critic output;

$w_q^j =$ the weight connecting the j^{th} fuzzy rule and the q^{th} action output;

$V =$ the critic output;

$A_q =$ the q^{th} action output;

$P =$ the total number of actions;

$a(i) \in \{1,\ldots,NM_i\}$

$i = 1,\ldots,K$;

$j = 1,\ldots,N$; and

$q = 1,\ldots,P$.

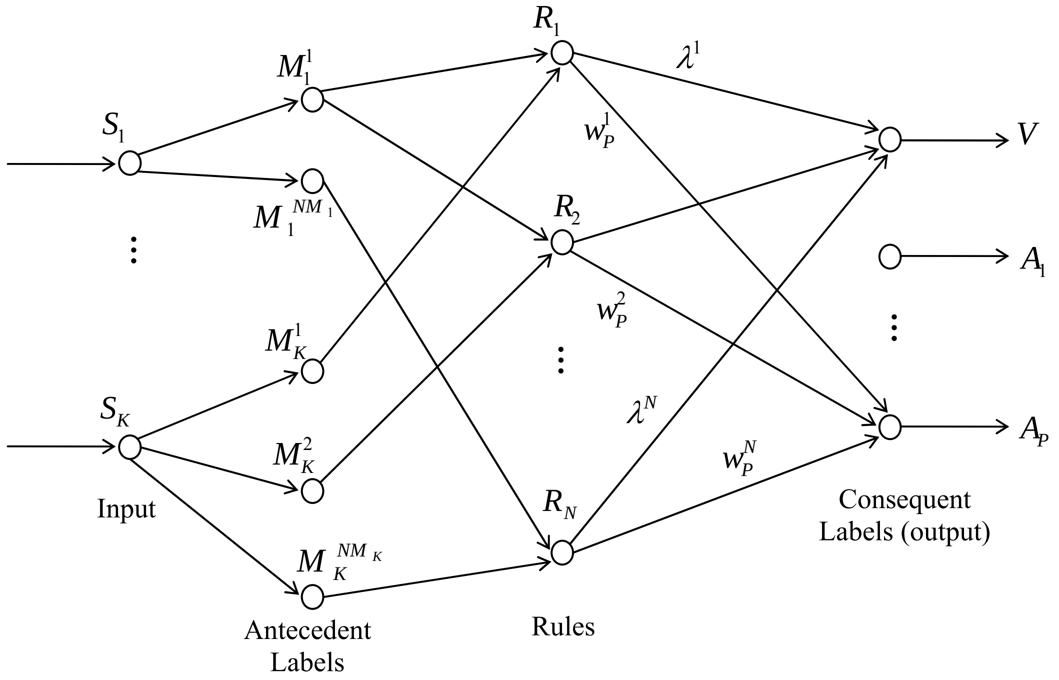


FIGURE 25 Example of the NFACRL (*81*).

Similar to neural networks, the NFACRL method has four layers as shown in Figure 25. The first layer is the input layer. It receives state variable values and sends them to different fuzzy membership functions in the second layer. Each node in the first layer represents an input (state) variable. Each node in the second layer is a fuzzy set with a fuzzy membership function associated with it. The inputs to the second layer are the state variable values, and the outputs of the second layer are fuzzy membership function values. The inputs and fuzzy sets of the second layer constitute many linguistic terms such as "Queue is {Short}" and "Queue is {Long}". Thus, the outputs of the second layer can also be considered as degrees of compatibility. The third layer corresponds to fuzzy rules in a fuzzy logic controller, and the outputs of the third layer can be considered as firing strengths. The fourth layer is a collection of nodes representing consequences. The first node stands for the *Critic* (see Figure 19), and its output value shows how good the current state is. The remaining nodes correspond to the available actions that can be taken, and their output values are the preferences to choose each action given the current state inputs.

There are three major differences between the architectures of the NFACRL method and the GARIC method used by Bingham (*5*).

1. First, the NFACRL method has multiple outputs that are crisp values representing *Critic* and actions, while the GARIC method only has one continuous output. Multiple outputs can be more useful for modeling phase sequence optimization than the single continuous output. Since GARIC only has a continuous output, it can only be used to decide weather and how to extend the current green phase. While for the NFACRL method, the multiple action outputs can be used to decide which control phase should be chosen for the next step.
2. Bingham (*5*) used GARIC mainly for fine tuning the parameters of fuzzy membership functions. The fuzzy rules in her study needed to be prespecified. If the control problem has many input variables, specifying the

fuzzy rules could be cumbersome and prone to error. It will be shown later that the NFACRL dos not need to specify the fuzzy rules.

3. GARIC uses a neural network as the *Critic* and a fuzzy logic controller as the *Actor*, and *Critic* and *Actor* are relatively independent of each other. For the NFACRL, *Critic* and *Actor* are closely related. Both of them use the same fuzzy membership function values as the inputs.

For the GARIC method, fuzzy rules need to be prespecified based on users' experience. If a control problem has many state variables, the fuzzy rules will become very complicated as the example shown below, and are difficult to specify even for very experienced experts.

- ♦ **IF** S_1 is {1} **AND** S_2 is {2} **AND** S_3 is {1} **AND** S_4 is {1} **AND** … **AND** S_K is {2}, **THEN** Action Output is { A_t }

Specifying fuzzy rules in the GARIC method is similar to determining how to connect the nodes in the second, third, and fourth layers in Figure 25. The maximum possible number of fuzzy rules is

$$Nmax = P\prod_{i=1}^{K} NM_i \qquad (41)$$

For control problems with many state variables, there could be several hundreds of complicated fuzzy rules need to be specified manually if the GARIC method is used. In the NFACRL method, by introducing the weights between the third and fourth layers, one can simply use $Nmax$ fuzzy rules. Through fine tuning the weights between the third and fourth layers, the best fuzzy rules can be found automatically even though the number of fuzzy rules can still potentially be large.

Calculation Procedure of the NFACRL

For the NFACRL control, the input and fuzzification parts are the same as the typical fuzzy logic control. Given the input and fuzzy membership functions, fuzzy membership function values are generated and fed into the third layer of NFACRL. The fuzzy inference method used in the NFACRL is a little different from what is shown in Figure 21. Assuming the j^{th} fuzzy rule has the following *K* antecedents

$$S_1 \in M_1^{a(1)}, S_2 \in M_2^{a(2)}, \ldots, S_K \in M_K^{a(K)} \tag{42}$$

then the firing strength of the j^{th} fuzzy rule is

$$FS_{R_j} = \prod_{i=1}^{K} \mu_{a(i)}^{j}(S_i) \tag{43}$$

where

$a(i) \in \{1, \ldots, NM_i\} =$ one of the fuzzy sets for the i^{th} input variable; and

$\mu_{a(i)}^{j}(S_i) =$ the membership function value of the $a(i)^{th}$ fuzzy set for the i^{th} input variable, and this value is used in the j^{th} fuzzy rule.

Some of the firing strengths may be zeroes, which means the corresponding fuzzy rules will not affect the final control output. After the firing strengths of each fuzzy rule are obtained, the next step is to calculate the preference of choosing each action using Equation (44).

$$\text{Pref}(A_q) = \sum_{j=1}^{N} FS_{R_j} w_q^j \tag{44}$$

where

$\mathrm{Pref}(A_q) =$ preference of choosing the q^{th} action; and

$q = 1,...,P$.

w_q^j in Equation (44) is also referred to as action weight. If the following two row vectors are used to represent firing strengths and action weights,

$$FS = \{FS_{R_1},..., FS_{R_N}\} \tag{45}$$

$$w_q = \{w_q^1,..., w_q^N\} \tag{46}$$

then Equation (44) can be rewritten as

$$\mathrm{Pref}(A_q) = FS(w_q)^T \tag{47}$$

In Equation (47), T means transpose. Similarly, the critic output of the NFACRL is defined in Equation (48).

$$V = \sum_{j=1}^{N} FS_{R_j} \lambda^j = FS(\lambda)^T \tag{48}$$

where

$\lambda = \{\lambda^1,..., \lambda^N\}$; and

$\lambda^j =$ the i^{th} critic weight connecting the i^{th} fuzzy rule and the critic output.

Learning Procedure of the NFACRL

The previous subsection describes how to calculate the outputs of NFACRL for given action and critic weights. In this subsection, the process of fine tuning the action and critic weights will be introduced.

Let $\lambda(t) = \{\lambda^1(t), \lambda^2(t), ..., \lambda^N(t)\}$ represent the critic weights at time step *t*, and $w_q(t) = \{w_q^1(t), w_q^2(t), ..., w_q^N(t)\}$ denote the action weights at time step t for the q^{th} action output. If the state variables at time step t are $S(t) = \{S_1(t), ..., S_K(t)\}$, then the critic and action outputs for state *S*(*t*) using weights at time step *t* are

$$V_t(S(t)) = FS(S(t))[\lambda(t)]^T \tag{49}$$

$$\text{Pref}_t(A_q, S(t)) = FS(S(t))[w_q(t)]^T \tag{50}$$

where

$FS\ (S\ (t)) =$ firing strengths calculated based on state variables at time step *t*;

$V_t(S(t)) =$ critic output calculated based on state variables at time step *t* and weights at time step *t*; and

$\text{Pref}_t(A_q, S(t)) =$ preference for the q^{th} action calculated based on state variables at time step *t* and weights at time step *t*.

After all the action outputs have been calculated, an ε-greedy algorithm is used to choose an action based on the calculated preferences of each action. If action j is selected and executed, and the resulted new state at time step *t*+1 is $S(t+1)$, then the critic output for state $S(t+1)$ using weights at time step *t* is

$$V_t(S(t+1)) = FS(S(t+1))[\lambda(t)]^T \tag{51}$$

where $V_t(S(t+1))$ is calculated based on input $S(t+1)$ using weights at time step t. The transition from states $S(t)$ to $S(t+1)$ also results in a reward r_{t+1} at time step $t+1$. Based on $V_t(S(t))$, $V_t(S(t+1))$, and r_{t+1}, A TD error is calculated using Equation (52).

$$\delta_t = r_{t+1} + \gamma V_t(S(t+1)) - V_t(S(t)) \tag{52}$$

This TD error is used to update both the critic and action weights using Equations (53) and (54), respectively.

$$\lambda(t+1) = \lambda(t) + \beta\delta_t FS(S(t)) \tag{53}$$

$$w_j(t+1) = w_j(t) + \beta\delta_t FS(S(t)) \tag{54}$$

where β is a learning rate to be specified. Note that at each step only the action weights connecting to the chosen (j^{th}) action are updated.

If after certain updating steps the changes of critic and action weights are less than a prespecified small value, or the control performance tends to be stable, the learning process is terminated and the trained NFACRL is then used for real world control applications. After the learning process is terminated, a greedy action selection strategy should be used in lieu of the ε-greedy action selection method, such that the NFACRL method will not give irrational instructions during implementation. The entire learning process of the NFACRL method is summarized in Figure 26.

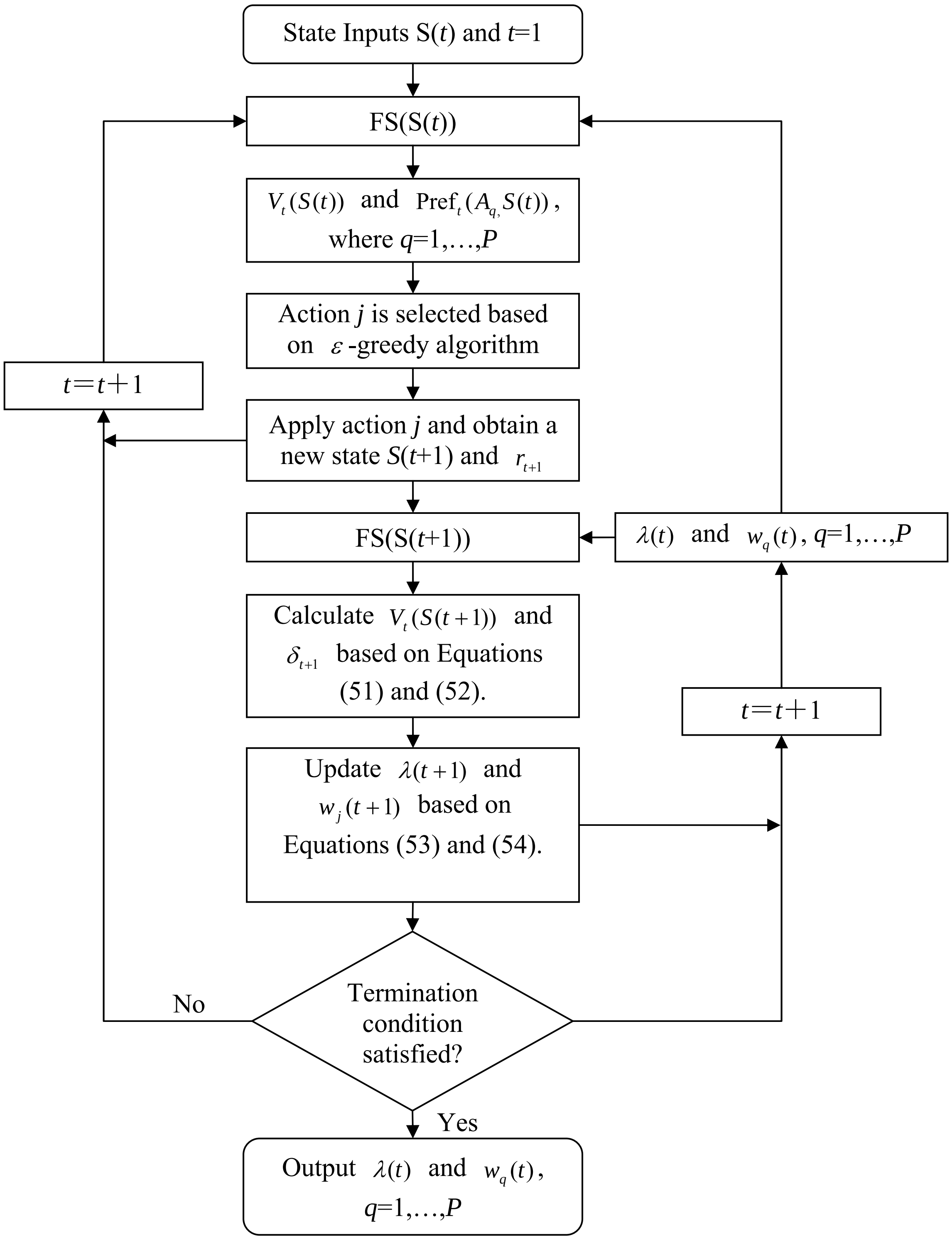

State Inputs S(t) and t=1
FS(S(t))
$V_t(S(t))$ and $\text{Pref}_t(A_{q,}S(t))$, where q=1,…,P
Action j is selected based on ε-greedy algorithm
Apply action j and obtain a new state S(t+1) and r_{t+1}
FS(S(t+1))
Calculate $V_t(S(t+1))$ and δ_{t+1} based on Equations (51) and (52).
Update $\lambda(t+1)$ and $w_j(t+1)$ based on Equations (53) and (54).
Termination condition satisfied?
No
Yes
Output $\lambda(t)$ and $w_q(t)$, q=1,…,P
t=t+1
$\lambda(t)$ and $w_q(t)$, q=1,…,P
t=t+1

FIGURE 26 Training process of the NFACRL method.

Summary of NFACRL

The NFACRL method is a combination of neural networks, fuzzy logic control, and actor-critic reinforcement learning, and is different from the GARIC method used by Bingham (*5*). It has the ability to handle phase sequence optimization of traffic signal control, large state space, generalization ability, and complicated fuzzy rules.

The following three problems can have significant effects on the performance of NFACRL. Before applying the NFACRL method to traffic signal control, these three problems need to be solved.

1. Choices of state variables and actions;
2. Definition of reward; and
3. Coordination of control agents.

In the following two sections, these three problems are addressed and two intersection and arterial control methods based on NFACRL are proposed.

ISOLATED INTERSECTION TRAFFIC CONTROL BASED ON NFACRL

Fixed Phase Sequence Control Based on NFACRL

There could be many different ways of applying the NFACRL method to intersection traffic control. One option is to consider a fixed phase sequence. In this case, the action space is to either extend the current green phase or terminate it, which is similar to what has been used in previous studies (*70,73*).

In this research, only four-approach and three-approach intersections are considered, as they are the most common types of intersections in real world. For a typical four-approach intersection in Figure 2, the following phase sequence shown in Figure 27 is used. The control logic starts with the first phase (ϕ_1), and then visits the remaining five phases one by one in order. After the last phase (ϕ_6) in the sequence has been visited, the control logic goes back to the first phase and repeats the entire process.

Similar phase sequences are also used in the pre-timed and actuated control strategies that are to be compared with the NFACRL control. These pre-timed and actuated control strategies are optimized by Synchro.

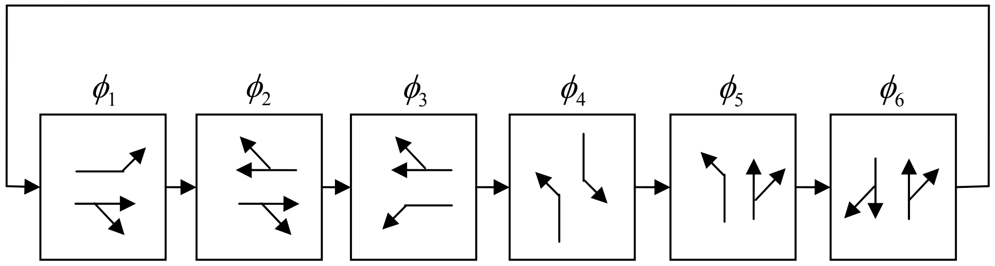


FIGURE 27 Phase plan for a four-approach isolated intersection.

For a three-approach isolated intersection with five movements as shown in Figure 28, the phase sequence shown in Figure 29 is used.

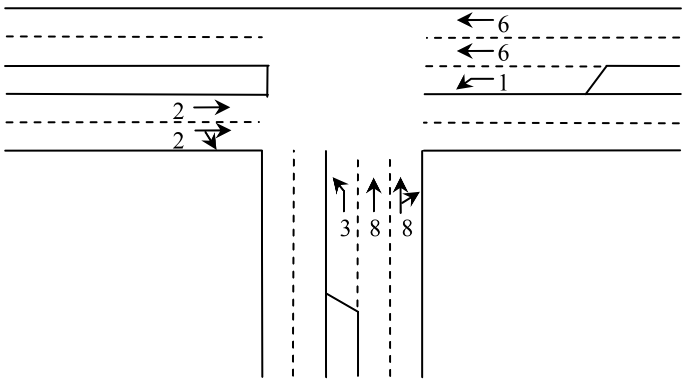


FIGURE 28 Layout of a typical three-approach intersection.

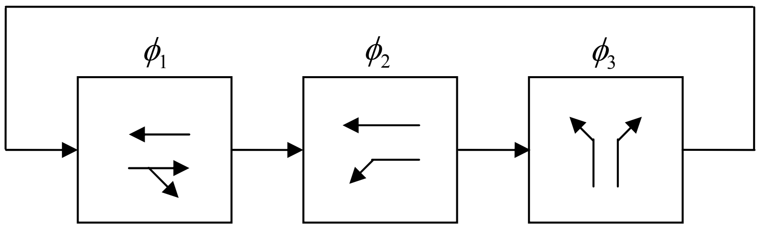


FIGURE 29 Phase plan for a three-approach isolated intersection.

Choices of State Variables

In most previous reinforcement learning traffic control studies, queue lengths were used as state variables (*5,70,73*). For the fixed phase sequence control based on NFACRL, queue lengths are also used as state variables. In addition to queue lengths, another state variable representing the current signal status is included.

For four-approach isolated intersections with eight movements (each through movement and its associated right-turn movement are combined as one movement) as in Figure 2, totally nine state variables are used, which means *K* in Figure 25 is equal to nine. The first eight state variables are used to represent the queue lengths and the last state variable is used to indicate the current signal state. More specifically, the first eight state variables are defined in Equation (55).

$$S_i = Q_i \tag{55}$$

where

$S_i =$ the i^{th} state variables;

$Q_i =$ queue length of the i^{th} movement (see Figure 2); and

$i = 1,...,8$.

The last state variable is defined as

$$S_9 = \begin{cases} 1 & \phi_1 = \text{Green, and } \phi_{i\neq 1} = \text{Red} \\ 2 & \phi_2 = \text{Green, and } \phi_{i\neq 2} = \text{Red} \\ 3 & \phi_3 = \text{Green, and } \phi_{i\neq 3} = \text{Red} \\ 4 & \phi_4 = \text{Green, and } \phi_{i\neq 4} = \text{Red} \\ 5 & \phi_5 = \text{Green, and } \phi_{i\neq 5} = \text{Red} \\ 6 & \phi_6 = \text{Green, and } \phi_{i\neq 6} = \text{Red} \end{cases}, \; i=1,\ldots,6 \tag{56}$$

For three-approach isolated intersections as the one shown in Figure 28, six state variables are used. Consequently, the parameter *K* in Figure 25 is equal to six. Among the six state variables, the first five ones are used to represent the queue lengths and are defined as

$$S_1 = Q_1 \tag{57}$$

$$S_2 = Q_2 \tag{58}$$

$$S_3 = Q_3 \tag{59}$$

$$S_4 = Q_6 \tag{60}$$

$$S_5 = Q_8 \tag{61}$$

The last state variable is used to indicate the current signal state and is defined as

$$S_6 = \begin{cases} 1 & \phi_1 = \text{Green, and } \phi_2, \phi_3 = \text{Red} \\ 2 & \phi_2 = \text{Green, and } \phi_1, \phi_3 = \text{Red} \\ 3 & \phi_3 = \text{Green, and } \phi_1, \phi_2 = \text{Red} \end{cases} \tag{62}$$

Fuzzy Membership Functions

To apply the NFACRL method, a set of fuzzy membership functions needs to be defined for the state variables. For each queue length state variable, two fuzzy sets are defined, which are {Short, Long}. The membership function for fuzzy set {Short} is defined in Equation (63).

$$\mu_{Short}(x) = \begin{cases} 1 & x \le 0 \\ (10 - x)/10 & x \in (0,10) \\ 0 & x \ge 10 \end{cases} \tag{63}$$

The membership function for fuzzy set {Long} is defined in Equation (64).

$$\mu_{Long}(x)=\begin{cases}0 & x\le 0\\ x/10 & x\in(0,10)\\ 1 & x\ge 10\end{cases} \quad \textbf{(64)}$$

The value 10 in both Equations (63) and (64) is a subjective number selected for this study. These fuzzy membership functions are illustrated in Figure 30.

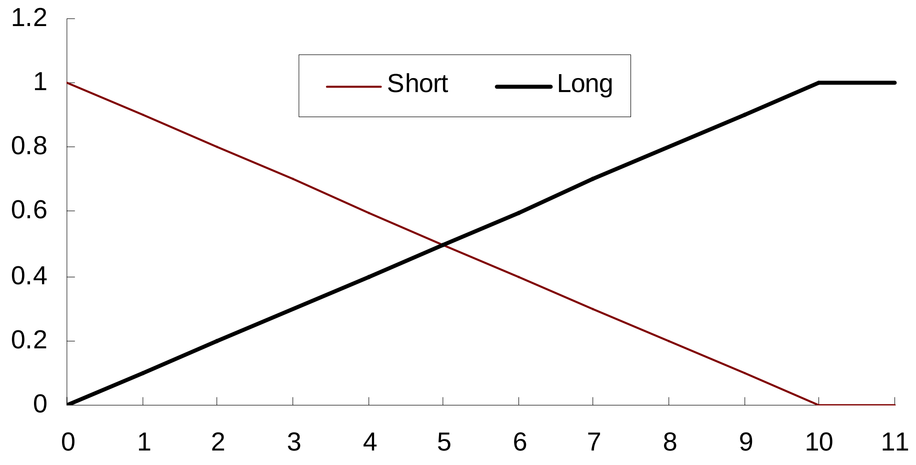


FIGURE 30 Fuzzy membership functions for queue length state variables.

For the state variable representing signal status, the definition of its fuzzy membership function is a little different. Using the three-approach intersection shown in Figure 28 as an example, the state variable S_6 has three fuzzy sets, which are $\{\phi_1,\phi_2,\phi_3\}$. The corresponding fuzzy membership functions are defined in Equations (65) through (67).

$$\mu_{\phi_1}(S_6)=\begin{cases}1 & S_6=1\\ 0 & \text{else}\end{cases} \quad \textbf{(65)}$$

$$\mu_{\phi 2}(S_6) = \begin{cases} 1 & S_6 = 2 \\ 0 & \text{else} \end{cases} \tag{66}$$

$$\mu_{\phi 3}(S_6) = \begin{cases} 1 & S_6 = 3 \\ 0 & \text{else} \end{cases} \tag{67}$$

Using the same principle, a set of fuzzy membership functions is defined for state variable S_9 for four-approach intersections.

Fuzzy Rules

For this fixed phase sequence control scheme, the third and fourth layers of the NFACRL (Figure 25) are assumed to be fully connected. Each node in the third layer has *K* connections with the second layer, one for each input state variable. Taking the three-approach intersection control as an example, a sample fuzzy rule is presented below

- ♦ **IF** S_1 is {Long} **AND** S_2 is {Short} **AND** S_3 is {Long} **AND** S_4 is {Short} **AND** S_5 is {Short} **AND** S_6 is {ϕ_1}, **THEN** Next Action is {Extension}

Since each of the five queue length state variables has two categories and the signal state variable has three values, totally there are 96 nodes in the third layer of the NFACRL (see Figure 25).

Similarly, for the four-approach intersection control, each of the eight queue length state variables has two fuzzy sets associated with it, and the signal state variable has six possible states. Therefore, for the four-approach intersection control there are a total of 1536 nodes in the third layer of the NFACRL (see Figure 25).

Definition of Reward

As shown in Equation (16), the objective of reinforcement learning is to find an optimal policy π^* (a mapping from states to actions) to maximize the reward of each state, and it is equivalent to maximizing the summation of discounted rewards shown in Equation (68).

$$\begin{aligned} V^*(s) &= \max_{a \in A(s)} E\left\{ \sum_{k=1}^{\infty} \gamma^{k-1} r_{t+k} \mid s_t = s \right\} \\ &= \max_{a \in A(s)} E\{ r_{t+1} + \gamma V^*(s') \mid s_t = s \} \end{aligned} \tag{68}$$

This is similar to the DYPIC method based on dynamic programming, whose optimization goal is in Equation (69).

$$f_i(j) = \min_{a_i} \{ C_{jk} + f_{i+1}(k) \}, \quad i = 1, \ldots, N, \quad j \in S_i, \quad k \in S_{i+1} \tag{69}$$

where C_{jk} is the total delay associated with transition from state *j* at stage *i* to state *k* at stage *i*+1. Comparing Equations (68) and (69) suggests that the minus delay between two decision points can be used as the reward.

Thorpe (*70*) used a linear combination of discharged and stopped vehicles as the reward. Bingham (*5*) used minus delay as the reward. Abdulhai et al. (*73*) also used minus total delay between two decision points as the reward, and the total delay was calculated by counting queue lengths every 1 second. It makes sense to use minus total delay as the reward, as minimizing delay is often used as the objective of traffic signal control. However, simply using queue length to represent delay in the reward function may not be enough, as queue length can not accurately reflect the delay caused by acceleration and deceleration maneuvers. Also, sometimes it is desirable to consider minimizing number of stops. Thus, in this research, the following reward definition is used.

$$r = \beta_1 x_1 - \beta_2 x_2 - \beta_3 x_3 + \beta_4 x_4 - \beta_5 x_5 \qquad \textbf{(70)}$$

where

$x_1 =$ number of vehicles that have passed the intersection from approaches being given green signal;

$x_2 =$ number of vehicles in queues;

$x_3 =$ number of vehicles newly added to queues;

$x_4 =$ number of vehicles in approaches being given green signal;

$x_5 =$ number of vehicles being stopped when signal is switched from green to red; and

$\beta_i =$ nonnegative coefficients for each variable. $i = 1,...,5$

x_1 encourages moving more vehicles through the intersection during two decision points; x_2 represents stopped delay; x_3 is used to account for deceleration delay; x_4 is to have more vehicles in the current green phase; and x_5 is used to penalize switching green signal to red while there are many vehicles being served by this green signal.

Variable Phase Sequence Control Based on NFACRL

Fixed phase sequence control based on NFACRL can significantly reduce the dimension of state and action spaces, consequently reducing the number of action and critic weights. However, the fixed sequence NFACRL control may lack the flexibility to fully adapt to traffic flow fluctuations due to the fixed phase sequence constraint. In this section, a variable phase sequence control method based on NFACRL is proposed. The variable phase sequence NFACRL control also uses queue lengths and signal states as inputs. But the decision output is not extension or termination. Instead, the decision output is any of the available control actions. For the three-approach intersection in Figure 28, the decision output could be ϕ_1, ϕ_2, or ϕ_3 shown in Figure 29. In this case, a sample fuzzy rule is

- **IF** S_1 is {Long} **AND** S_2 is {Short} **AND** S_3 is {Long} **AND** S_4 is {Short} **AND** S_5 is {Short} **AND** S_6 is {ϕ_1}, **THEN** Next Action is {ϕ_3}

For the four-approach intersection in Figure 2, the decision output could be any of the eight phases in Figure 24.

For the three-approach intersection, five queue length state variables and one signal state variable are used in the variable phase sequence NFACRL control. These variables are defined exactly the same as in the fixed phase sequence NFACRL control. Namely, each queue length state variable has two fuzzy sets and each signal state variable has three fuzzy sets. Therefore, the variable phase sequence NFACRL has 96 nodes in the third layer (see Figure 25).

For the four-approach intersection in Figure 2, eight queue length state variables and one signal state variable are used in the variable phase sequence NFACRL control. The eight queue length state variables are defined the same as in the fixed phase sequence NFACRL control. The signal state variable is defined a little differently, which is shown in Equation (71).

$$S_9 = j,\ \text{if}\ \phi_j = \text{Green and}\ \phi_{i \neq j} = \text{Red},\ (i,j = 1,\ldots,8) \tag{71}$$

Therefore, for four-approach intersection control with variable phase sequence, the NFACRL method has 2048 nodes in the third layer (see Figure 25).

For the variable phase sequence NFACRL control, the same fuzzy membership functions in Figure 30 are used for all queue length state variables. The fuzzy membership functions for the signal state variables are defined in the same way as in the fixed phase sequence NFACRL control. The reward definition used in the fixed phase sequence NFACRL control is also used in the variable phase sequence NFACRL control, but different coefficients for each variable are chosen.

ARTERIAL TRAFFIC CONTROL BASED ON NFACRL

Multiagent Reinforcement Learning

Isolated intersection control is a single agent decision problem. For a system that has more than one intersection, multiple control agents should be used. A system consists of several agents is usually referred to as multiagent system (MAS). As many practical control problems, such as arterial traffic control, can be modeled as MASs, multiagent reinforcement learning (MARL) has attracted considerable attention over the past two decades (*62,83,84,85,86,87,94,88,89,88,89*). In the following subsections, three major MARL methods are briefly reviewed.

MARL Based on Independent-Agent

Independent-agent is the simplest MARL method. It directly applies single-agent reinforcement learning to MAS. Each agent treats all other agents as part of the environment (*62*). One potential problem of this method is that the existence of other agents may affect the environment and invalidate the Markov property assumption (*90*).

MARL Based on SG

Many MARL studies have been focused on using stochastic game (SG) or Markov game (MG). SG is a natural extension of MDP to handle problems with multiple agents. Recall that in Chapter III a MDP is defined by a tuple (S, A, r, p). Similarly, a SG is defined by a more complicated tuple as $(n, S, A_1, ..., A_n, r_1, ..., r_n, p)$ (*62,91,92,93*), where

1. n is the total number of agents in the MAS;
2. S is a set of discrete states;

3. $A_i \quad (i = 1,...,n)$ is the action space for the i^{th} agent;
4. $r_i \quad (i = 1,...,n)$ is the reward function for the i^{th} agent, which is affected by the current system state and all actions that will be taken; and
5. p is a transition function, which gives the probability that the system will be in each state provided with the current system state and actions to be taken.

Under the framework of SG, the state transition is still assumed to satisfy the Markov Property.

Littman (*93*) appears to be the first researcher to use SG as the framework to solve MARL problems. He studied a two-agent zero-sum SG problem, and proposed a minimax-Q algorithm similar to Q-Learning for solving this problem. For the two-agent zero-sum SG problem, there are two competing agents. The gain of one agent always leads to the loss of another, and the summation of gains from both agents is equal to zero. For arterial traffic signal control, the gain of one control agent does not necessarily mean the loss of other agents. Therefore, the zero-sum SG framework is not suitable for modeling arterial traffic control problems.

Hu and Wellman (*62*) further researched the MARL problem under the framework of general-sum SG, in which different agents can increase their gains simultaneously. They developed a multiagent Q-Learning algorithm to solve n-agent general-sum SG problems. For ease of description, the following discussions only consider a two-agent general-sum SG problem. Different from the Q-Learning for MDP, the multiagent Q-Learning proposed by Hu and Wellman (*62*) requires each agent to keep two Q-Tables, one for itself and one for the other agent in the system. Using agent 1 as an example, during the learning process, it updates it own Q-Table using Equation (72).

$$V_{t+1}^1(s_t, a_t^1, a_t^2) = V_t^1(s_t, a_t^1, a_t^2) + \phi\left[r_{t+1}^1 + \gamma\pi^1(s_{t+1})V_t^1(s_t, a_t^1, a_t^2)\pi^2(s_{t+1}) - V_t^1(s_t, a_t^1, a_t^2)\right] \quad \textbf{(72)}$$

where

$V_{t+1}^{1}(s_t, a_t^1, a_t^2) =$ action function value for agent 1 at time step *t*+1;

$a_t^1 =$ action taken by agent 1 at time step *t*;

$a_t^2 =$ action taken by agent 2 at time step *t*;

$\pi^1(\cdot) =$ policy function of agent 1;

$\pi^2(\cdot) =$ policy function of agent 2;

$r_{t+1}^1 =$ reward for agent 1 at time step *t*+1; and

$\pi^1(s_{t+1})V_t^1(s_t, a_t^1, a_t^2)\pi^2(s_{t+1}) =$ is the expected reward of agent 1 under the mixed strategy Nash Equilibrium (*62*);

Note that updating agent 1's state action function (Q-Table) needs the policy function information of agent 2. This can be done by keeping track of agent 2's Q-Table using the following Equation (73). Detailed updating procedure can be found in **(***62***)**.

$$V_{t+1}^2(s_t, a_t^1, a_t^2) = V_t^2(s_t, a_t^1, a_t^2) + \phi\left[r_{t+1}^2 + \gamma\pi^1(s_{t+1})V_t^2(s_t, a_t^1, a_t^2)\pi^2(s_{t+1}) - V_t^2(s_t, a_t^1, a_t^2)\right] \quad (73)$$

There are two major difficulties in applying this multiagent Q-Learning method to arterial traffic control. First, with multiple intersections, the number of state variables will become very large and make the learning process extremely slow. Based on previous discussions on a four-approach intersection, there could be 9 state variables. If an arterial has four such intersections, then the total number of state variables is 36. Assuming each state variable has 2 categories, the total number of possible states is $2^{36} \approx 6.9 \times 10^{10}$. The huge number of possible states will not only considerably slow down the reinforcement learning process, but also give rise to the generalization problem.

MARL Based on Cooperative-Agent

MARL based on the SG framework is theoretically sound. However, it is not suitable for real world control applications due to its complexity and large state space. Tan (*94*) conducted a study to compare the performance of independent-agent and cooperative-agent in a MAS. For independent-agent method, agents treat each other as part of the environment. While for cooperative-agent method, agents share information with each other. For the cooperative-agent MARL method, Tan (*94*) experimented with the following three cooperation strategies:

1. The first strategy shared real-time state information among all agents. Although testing results showed that sometimes cooperative-agent method using this strategy could moderately outperform the independent-agent method, this strategy significantly increased the state space of each agent in the system and might not be suitable for arterial traffic signal control.
2. The second strategy shared experiences among all agents. These experiences were different from the instant information shared in the first strategy. They were past state, action, and reward information experienced by each agent. Tan reported that the second strategy improved the learning speed. However, it produced approximately the same performance as the independent-agent method did.
3. The third strategy was similar to the first one. But the author applied it to a new problem, in which two agents were designed to accomplish a common task. In addition to having the large state space problem of the first strategy, the third strategy required a lot of communications between the two agents.

Arterial Traffic Control Using Multiagent NFACRL

Review in previous section shows that there are basically three MARL methods:

1. MARL based on independent-agent;

2. MARL under the framework of SG; and
3. MARL based on cooperative-agent that shares experiences or information.

Due to the large state and action spaces problem, the second method under the framework of SG is ruled out for arterial traffic control in this research. In fact, this method so far has mainly been used in theoretical studies. The cooperative-agent method may also not be a good idea. As in this research, each intersection is controlled by an agent. Since different intersections may have different geometric settings, their environments are most likely different. Under this circumstance, sharing experience among different agents may not be useful. In addition, previous study by Tan (*94*) showed that sharing experience among agents only expedited the learning process and did not appear to improve the learning results.

For the independent-agent MARL method, agents are expected to learn how to coordinate implicitly. Although the first strategy is very simple, it can be very useful in practice. Compared to the other two more complicated MARL methods, it has the following nice properties:

1. No communication devices need to be installed between adjacent intersections.
2. Simplicity sometimes means robustness. In this case, the malfunction of other controllers will not directly affect the function of the current controller.

With all the above considerations, in this research the independent-agent method is chosen to coordinate different control agents.

SUMMARY

In this chapter, a neuro-fuzzy actor-critic reinforcement learning (NFACRL) method was introduced for adaptive traffic signal control. NFACRL uses a neuro-fuzzy network to store the actor and critic values of each state, such that the curse of dimensionality and generalization problems can be properly handled. It also has the ability to model discrete action outputs and can be used to optimize phase sequence of traffic signal control. To present the NFACRL method more clearly, fuzzy logic control and neural networks were also briefly discussed at the beginning of this chapter.

After the NFACRL method was introduced, two implementation schemes were proposed to apply the NFACRL method to isolated intersection traffic control. The first scheme considered a fixed phase sequence and the second one did not. For both implementation schemes, the implementation details such as the choice of state and action variables, fuzzy membership functions, fuzzy rules, and reward functions were discussed in details.

The two NFACRL control methods were further extended for the traffic control of an arterial consisting of several intersections. Each intersection was controlled by an agent and the arterial traffic signal control was modeled as a multiagent system. Various methods to coordinate different agents in this multiagent system were reviewed. Based on the review, a simple but robust independent-agent method was adopted for arterial adaptive traffic signal control.

CHAPTER V

EVALUATION OF THE NFACRL TRAFFIC CONTROL METHOD BASED ON MICROSCOPIC SIMULATION

INTRODUCTION

This chapter discusses in details the evaluation of the NFACRL traffic control using VISSIM microscopic traffic simulation. The evaluation is carried out at both isolated intersection and arterial levels based on simulation network created from real world data. The fixed and variable NFACRL control schemes for isolated intersection traffic control are evaluated first. Both NFACRL control schemes are then extended to arterial traffic control by using an independent-agent coordination method. For the isolated intersection evaluation, the two NFACRL control schemes are compared with optimized pre-timed and actuated control. For the arterial evaluation, the two NFACRL control schemes are compared with optimized coordinated pre-timed and coordinated actuated control.

The rest of this chapter is organized as the following: first, data used for setting up the simulation traffic network are described. Secondly, the VISSIM microscopic traffic simulation program used in this research is discussed. Details about how to code the simulation traffic network and various control algorithms are also presented. Thirdly, test design is described. Following that are the testing results at both intersection and arterial levels. The last section summarizes this chapter.

DATA DESCRIPTION

Data from a real world arterial network in College Station, Texas are used. The chosen arterial is a segment of FM 2818 (Harvey Mitchell Parkway), shown in Figure 31, which include three four-approach intersections and one three-approach intersection. The traffic volume data for each intersection in Figure 31 are listed in Tables 3 through 5. The morning peak period traffic data were collected on October 7, 2004 from 7:00 A.M.

to 8:00 A.M.; the noon peak period traffic data were collected on October 12, 2004 from 11:45 A.M. to 12:45 P.M.; and the afternoon peak period traffic data were also collected on October 12, 2004 but from 4:45 P.M. to 5:45 P.M.

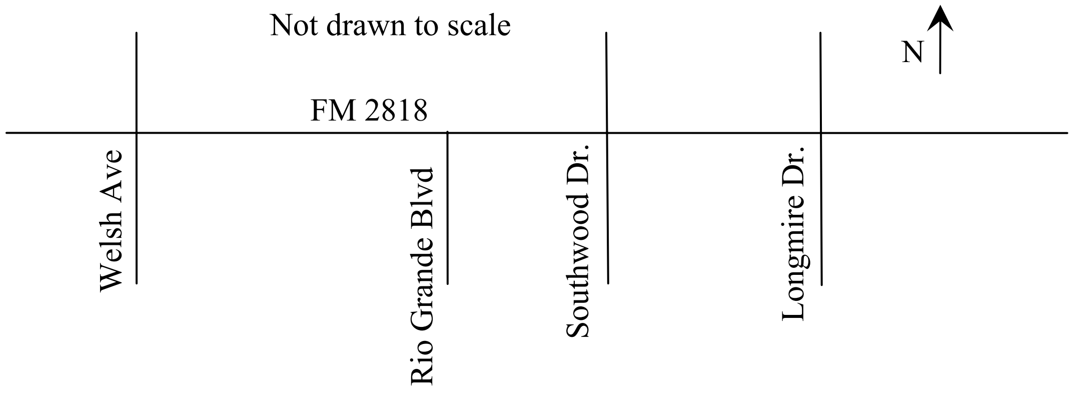


FIGURE 31 Testing arterial network.

MICROSCOPIC TRAFFIC SIMULATION

Microscopic traffic simulation has been used as a standard method for testing and comparing different traffic control strategies. Compared to evaluating traffic control strategies in the real world, using microscopic traffic simulation has the following advantages:

TABLE 3 Traffic Volume Data during Morning Peak Hour

Intersection	Time	Southbound				Westbound				Northbound				Eastbound			
		R	T	L	All	R	T	L	All	R	T	L	All	R	T	L	All
Longmire & FM 2818	7:15:00 AM	4	4	6	14	2	204	29	235	56	10	61	127	24	150	4	178
	7:30:00 AM	4	5	6	15	7	277	33	317	63	19	81	163	55	237	8	300
	7:45:00 AM	11	7	4	22	7	162	29	198	43	12	27	82	56	184	2	242
	8:00:00 AM	4	7	3	14	4	73	25	102	39	9	29	77	32	128	2	162
Southwood & FM 2818	7:15:00 AM	16	5	13	34	2	205	1	208	9	8	26	43	14	151	27	192
	7:30:00 AM	28	10	7	45	3	315	0	318	13	32	53	98	16	231	60	307
	7:45:00 AM	30	9	25	64	3	327	3	333	15	24	51	90	11	235	38	284
	8:00:00 AM	13	6	15	34	3	104	6	113	4	15	17	36	16	163	24	203
Rio Grande & FM 2818	7:15:00 AM	-	-	-	-	-	217	8	225	41	-	54	95	8	148	-	156
	7:30:00 AM	-	-	-	-	-	353	12	365	86	-	113	199	16	179	-	195
	7:45:00 AM	-	-	-	-	-	404	14	418	112	-	109	221	24	217	-	241
	8:00:00 AM	-	-	-	-	-	191	10	201	50	-	52	102	18	174	-	192
Welsh & FM 2818	7:15:00 AM	16	26	21	63	22	145	12	179	39	53	61	153	6	93	15	114
	7:30:00 AM	17	39	45	101	61	206	16	283	26	101	82	209	1	102	30	133
	7:45:00 AM	30	47	58	135	74	208	10	292	10	99	96	205	8	124	32	164
	8:00:00 AM	29	50	49	128	23	142	24	189	14	78	49	141	8	99	11	118

NOTE: L – Left-Turn Movement; T – Through Movement; R – Right-Turn Movement

TABLE 4 Traffic Volume Data during Noon Peak Hour

Intersection	Time	Southbound				Westbound				Northbound				Eastbound			
		R	T	L	All	R	T	L	All	R	T	L	All	R	T	L	All
Longmire & FM 2818	12:00:00 PM	10	11	12	33	11	118	64	193	82	9	42	133	38	106	6	150
	12:15:00 PM	3	13	11	27	5	131	69	205	88	19	58	165	34	106	8	148
	12:30:00 PM	3	12	8	23	6	92	49	147	74	6	61	141	46	114	8	168
	12:45:00 PM	8	18	6	32	7	80	41	128	57	20	39	116	48	149	8	205
Southwood & FM 2818	12:00:00 PM	12	10	13	35	4	144	6	154	10	8	17	35	32	147	18	197
	12:15:00 PM	29	17	8	54	7	168	3	178	8	13	24	45	24	138	24	186
	12:30:00 PM	15	10	13	38	6	143	6	155	7	12	17	36	17	133	15	165
	12:45:00 PM	18	8	5	31	6	121	5	132	11	11	20	42	24	187	26	237
Rio Grande & FM 2818	12:00:00 PM	-	-	-	-	-	161	19	180	29	-	21	50	18	165	-	183
	12:15:00 PM	-	-	-	-	-	184	22	206	23	-	20	43	17	170	-	187
	12:30:00 PM	-	-	-	-	-	168	24	192	21	-	14	35	7	148	-	155
	12:45:00 PM	-	-	-	-	-	145	14	159	31	-	11	42	14	202	-	216
Welsh & FM 2818	12:00:00 PM	5	17	23	45	10	115	39	164	21	12	21	54	15	146	1	162
	12:15:00 PM	4	14	16	34	8	150	25	183	22	25	28	75	23	118	3	144
	12:30:00 PM	1	20	20	41	27	108	26	161	21	25	29	75	14	105	4	123
	12:45:00 PM	3	26	20	49	13	104	23	140	46	20	29	95	26	128	8	162

NOTE: L – Left-Turn Movement; T – Through Movement; R – Right-Turn Movement

TABLE 5 Traffic Volume Data during Afternoon Peak Hour

Intersection	Time	Southbound				Westbound				Northbound				Eastbound			
		R	T	L	All	R	T	L	All	R	T	L	All	R	T	L	All
Longmire & FM 2818	5:00:00 PM	9	10	12	31	12	197	47	256	64	22	59	145	71	185	12	268
	5:15:00 PM	10	28	11	49	16	204	49	269	75	21	68	164	81	170	9	260
	5:30:00 PM	10	16	11	37	13	161	61	235	62	12	56	130	68	184	8	260
	5:45:00 PM	4	14	12	30	19	196	55	270	73	17	43	133	72	200	11	283
Southwood & FM 2818	5:00:00 PM	23	13	17	53	10	249	6	265	19	17	27	63	20	219	34	273
	5:15:00 PM	49	23	20	92	7	231	15	253	14	13	22	49	27	248	17	292
	5:30:00 PM	39	19	20	78	13	237	10	260	10	12	13	35	30	248	34	312
	5:45:00 PM	31	17	12	60	8	193	7	208	7	12	12	31	17	288	32	337
Rio Grande & FM 2818	5:00:00 PM	-	-	-	-	-	261	48	309	40	-	18	58	19	229	-	248
	5:15:00 PM	-	-	-	-	-	231	54	285	34	-	25	59	41	256	-	297
	5:30:00 PM	-	-	-	-	-	241	56	297	34	-	32	66	40	287	-	327
	5:45:00 PM	-	-	-	-	-	205	49	254	48	-	15	63	42	284	-	326
Welsh & FM 2818	5:00:00 PM	6	48	37	91	29	170	51	250	51	51	27	129	50	143	13	206
	5:15:00 PM	13	53	42	108	40	163	52	255	53	50	30	133	61	197	12	270
	5:30:00 PM	16	83	27	126	22	144	49	215	55	46	29	130	54	190	20	264
	5:45:00 PM	14	76	41	131	26	105	59	190	43	54	12	109	40	147	14	201

NOTE: L – Left-Turn Movement; T – Through Movement; R – Right-Turn Movement

1. It is cost effective. Testing a traffic control system using microscopic simulation is much easier than doing it in the real world. This saves a lot of efforts including installment of communication hardware, deployment of detectors, and related construction work.
4. It is safe. For new traffic control systems that are still in the testing stage, evaluating it in the real world may cause unexpected results such as serious traffic accidents.
5. It is fast. Implementing a traffic control system in microscopic simulation can be done in a few days, and the testing can usually be accomplished with a desktop computer.
6. It is very flexible. Traffic analysts can modify parameters or traffic network settings conveniently to suit different analysis purposes. Doing the same in the real world would be cumbersome or even impossible.
7. It is controllable. By using the same random number, traffic analysts can test different traffic control strategies under exactly the same traffic condition. While it is usually impossible to replicate the exact same conditions in the real world. Since different traffic control strategies will have to be tested during different time periods, there is no way to expect the traffic conditions during those time periods to be exactly the same. The difference in traffic conditions often makes the comparison results questionable, causing difficulties to draw valid and convincing conclusions from the results (*41*).

There are many microscopic traffic simulation packages being used, including VISSIM (*72*), CORSIM (*71*), AIMSUN (*95*), and Paramics (*96*). There have been studies comparing different traffic simulation programs (*97*), however, no universal consensus has been reached as to which program is the best one. In this research, VISSM is chosen mainly for the following reasons:

1. VISSIM is one of the most popular microscopic traffic simulation software being widely used around the world, and has been trusted by many traffic engineering researchers and practitioners. Using VISSIM as the testing platform makes it easy for other researchers to compare their traffic control methods with the one proposed in this research.
2. VISSIM provides a NEMA editor that can code actuated traffic signal control. Actuated traffic signal control is considered to be better than pre-timed control and is used as one of the baselines in this study.
3. VISSIM has a signal control DLL (Dynamic-Link Library) interface that can be used to code and test the proposed NFACRL control method.

TESTING DESIGN

Testing Procedure

The testing of the proposed NFACRL control method is conducted at both isolated intersection and arterial levels. The intersection at Welsh Avenue and the intersection at Rio Grande Boulevard (three-approach intersection) in Figure 31 are chosen for isolated intersection control testing, and the entire arterial network in Figure 31 is used for arterial control testing.

For testing on the two isolated intersections, the fixed and variable phase sequence NFACRL control schemes are evaluated and compared with pre-timed and actuated control. The pre-timed and actuated control plans are optimized by Synchro (*103*). The two NFACRL controllers are first trained using simulated traffic data and then applied to control the same simulated traffic. To make the evaluation and comparison results more convincing, each of the four control methods is tested 30 times using different random seeds.

The fixed and variable phase sequence NFACRL control schemes are extended to control the entire arterial using an independent-agent coordination method. Based on this coordination method, each intersection is controlled by one NFACRL controller.

These NFACRL controllers treat each other as part of the environment and learn how to coordinate implicitly. The NFACRL controllers based on the two schemes are trained and evaluated. Their performances on arterial control are then compared with those of coordinated pre-timed and coordinated actuated control. Again, the coordinated pre-timed and coordinated actuated control plans are optimized by Synchro (*103*). Each of the four control methods is tested 30 times independently using different random seeds.

Testing Under Different Flow Patterns

Using the intersection at Welsh Avenue in Figure 31 as an example, the northbound traffic volumes during morning, noon, and afternoon peak hours are plotted in Figure 32. Similarly, the southbound, eastbound, and westbound traffic volumes of this intersection are plotted in Figures 33 through 35. These figures clearly show that during different time periods of the day, traffic volumes of this intersection exhibit quite different patterns.

To better illustrate the difference among traffic flow patterns during morning, noon, and afternoon peak periods, the total entrance traffic volumes during each of these three peak periods are plotted in Figure 36. This figure shows that the total entrance traffic volumes during morning and afternoon peak periods are significantly larger than that during noon peak period.

To give a thorough evaluation of the proposed two NFACRL control schemes, they are tested using these three sets of traffic volume data at both the isolated intersections and the arterial.

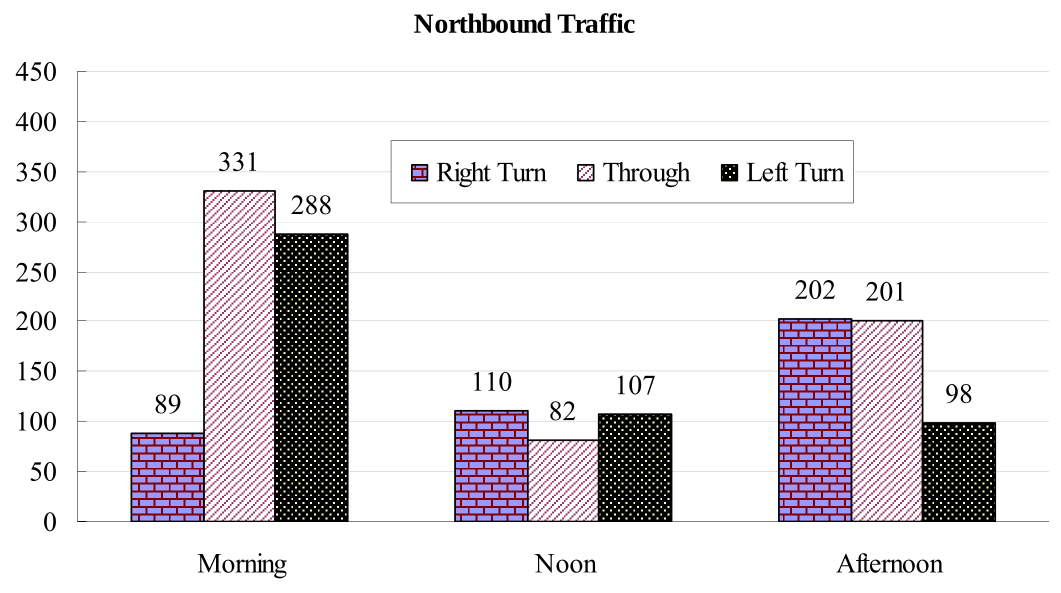


FIGURE 32 Northbound traffic flows of the intersection of FM 2818 and Welsh Avenue.

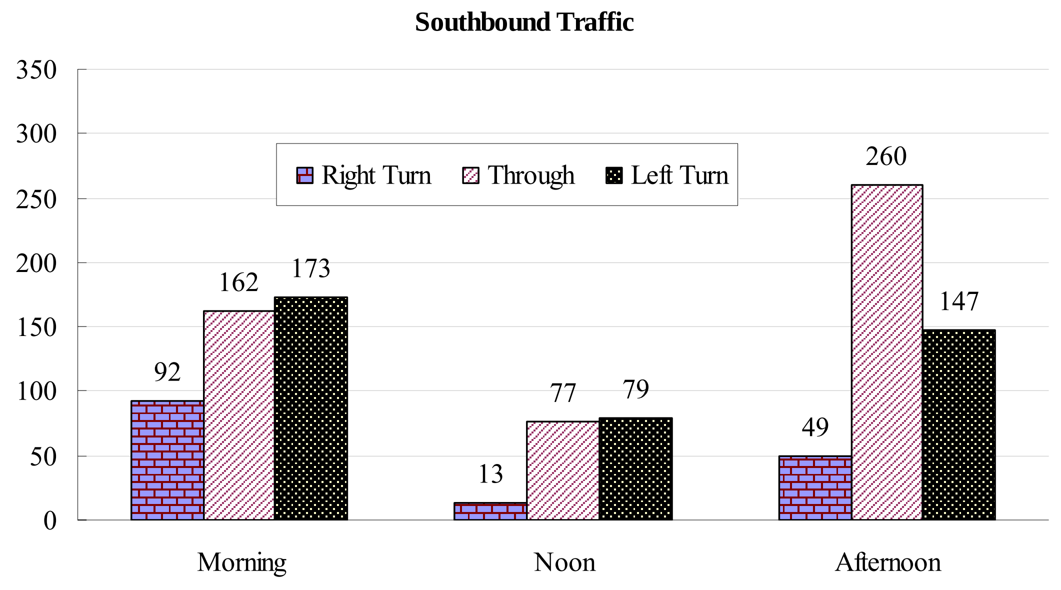


FIGURE 33 Southbound traffic flows of the intersection of FM 2818 and Welsh Avenue.

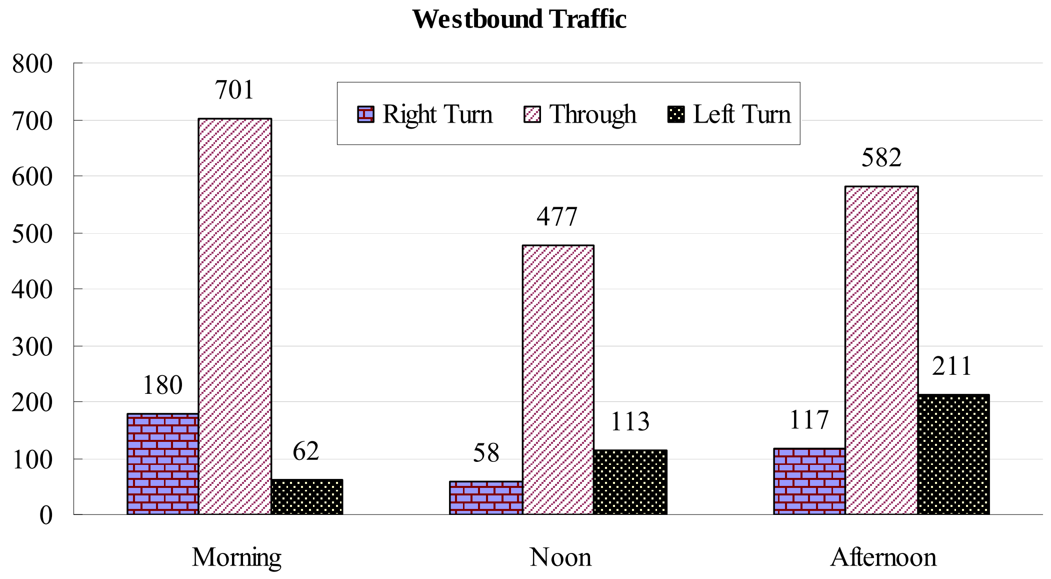


FIGURE 34 Westbound traffic flows of the intersection of FM 2818 and Welsh Avenue.

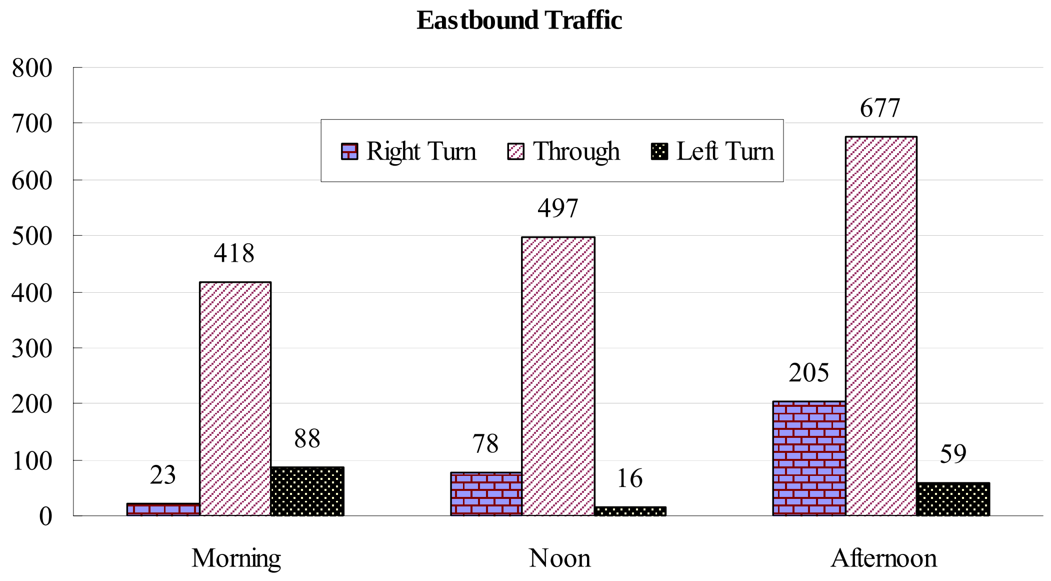


FIGURE 35 Eastbound traffic flows of the intersection of FM 2818 and Welsh Avenue.

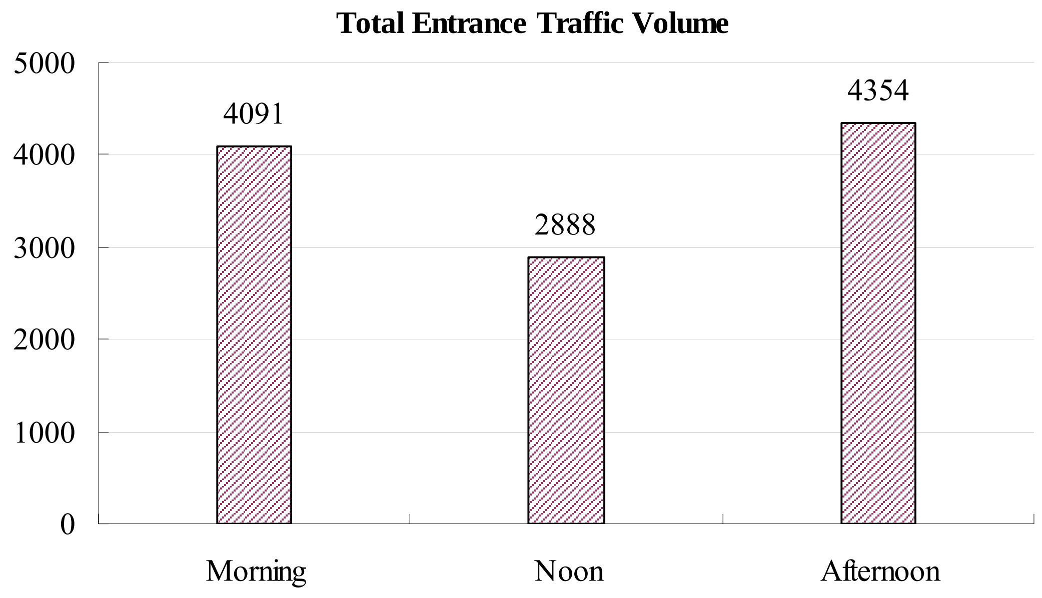


FIGURE 36 Total entrance traffic volumes.

Network Coding

GIS data from the website of the City of College Station (*98*) are used to code the arterial network. The coded arterial network in VISSIM is shown in Figure 37.

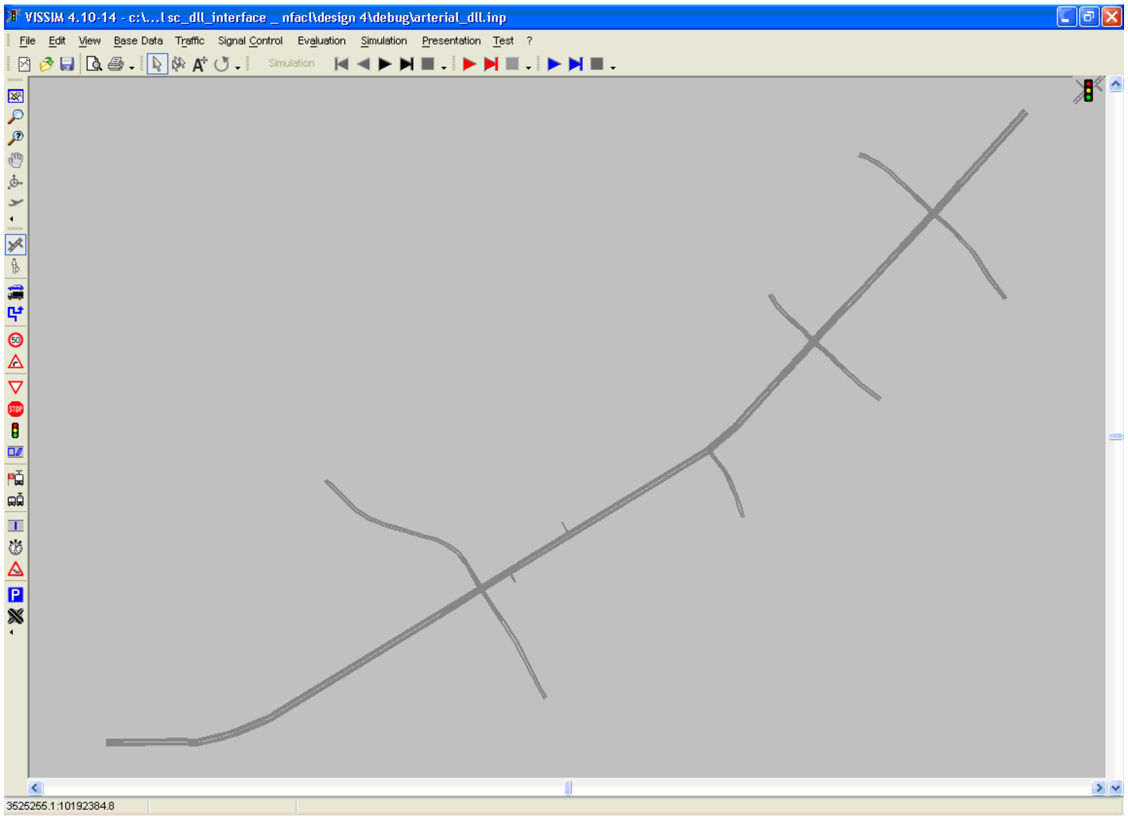

VISSIM 4.10-14 - c:\...\ sc_dll_interface _ nfacl\design 4\debug\arterial_dll.inp
File Edit View Base Data Traffic Signal Control Evaluation Simulation Presentation Test ?
Simulation
3525255.1:10192384.8

FIGURE 37 Coded arterial network.

Algorithm Implementation

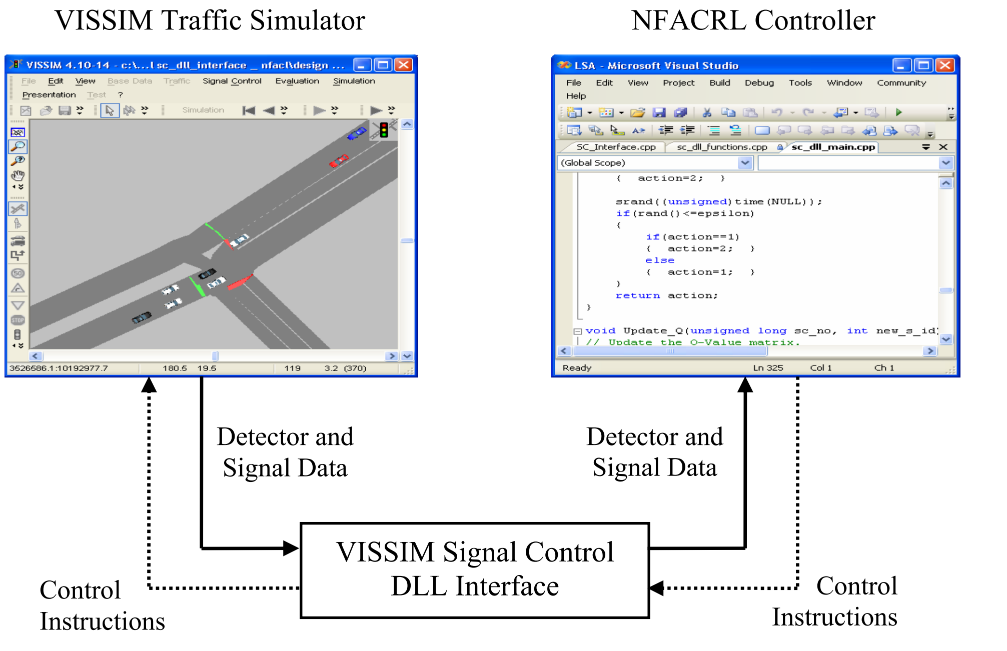
Pre-Timed and Actuated Control

Many software packages can be used to optimize pre-timed and actuated traffic control plans for both isolated intersections and arterials. Those packages include Synchro (*103*), PASSER II (*99*), PASSER V (*100*), and TRANSYT-7F (*101*). Synchro is chosen for this research as it has a very friendly user interface and its performance is also comparable with or better than other packages (*102*). Synchro is also more commonly used in practice than other packages.

The cycle length and phase duration optimization algorithm used in Synchro is based on the method in Highway Capacity Manual 2000 (*20*). In addition to cycle length and phase duration, the algorithm used in Synchro can also optimize phase sequence and offsets. More information on Synchro can be found in (*102,103*). The optimized pre-timed and actuated control plans are coded in VISSIM using the provided fix timed controller and the NEMA controller (*104*).

Reinforcement Learning Control

One reason for choosing VISSIM as the simulation platform is that VISSIM has a convenient DLL interface. With the help of this DLL interface, users can implement their own algorithms to control the simulated traffic. In this study, the two NFACRL control schemes are first coded as DLL files using the C++ language. The NFACRL control schemes in the form of DLL files communicate with the simulated traffic through the DLL interface. The entire idea of control flow using the DLL feature is illustrated in Figure 38.



```
        {   action=2;   }

        srand((unsigned)time(NULL));
        if(rand()<=epsilon)
        {
            if(action==1)
            {   action=2;   }
            else
            {   action=1;   }
        }
        return action;
    }

void Update_Q(unsigned long sc_no, int new_s_id)
// Update the Q-Value matrix.
```

FIGURE 38 DLL interface and implementation of NFACRL control schemes.

It can be seen from Figure 38 that the DLL interface functions as a relay. It obtains detector outputs and signal states from the VISSIM microscopic traffic simulator and sends them to the NFACRL controller. In return, the NFACRL controller gives control instructions back to the VISSIM simulator to control the simulated traffic. The DLL interface in this case can also be regarded as a translator. It interprets the instructions sent by the NFACRL controller and changes signal states in the VISSIM traffic simulator accordingly.

Performance Evaluation Criteria

VISSIM provides various outputs, including average delay per vehicle, average stopped delay per vehicle, and average number of stops per vehicle. Similar outputs have been used by several researchers for evaluating traffic control methods (*4,17,105*). In this research, all these three outputs are adopted as performance evaluation criteria for both isolated intersection and arterial control. For ease of description, these three performance criteria hereinafter are referred to as delay, stopped delay, and number of stops per vehicle.

For arterial control, three additional criteria are used, which are overall average speed, throughput, and average arterial travel time. Overall average speed is the average speed of all vehicles in the arterial system. Throughput is defined as the number of vehicles that have passed through the arterial system. Average arterial travel time is defined as the average travel time for vehicles traveling from one end of the arterial to the other end. In the rest of the dissertation, overall average speed and average arterial travel time are referred to as speed and arterial travel time, respectively.

PERFORMANCE EVALUATION ON ISOLATED INTERSECTIONS

Two isolated intersections are chosen for evaluating the proposed NFACRL methods. The first is the intersection of Welsh Avenue and FM 2818, which is a four-approach intersection. The second is the intersection of Rio Grande Boulevard and FM 2818, which is a three-approach intersection. For ease of description, hereinafter these two intersections are referred to as four-approach and three-approach intersections. Also, the two NFACRL methods with fixed and variable phase sequences are referred to as NFACRL-F and NFACRL-V, respectively.

NFACRL controllers are like neural networks. They need to be trained before they can actually be used. In this study, the NFACRL controllers were trained 90 runs based on the data from each selected intersection. The trained NFACRL controllers were then applied to the two intersections for performance evaluation. As described before, for each intersection under different traffic flow conditions, the performance evaluation process was repeated 30 runs with different random seeds.

Evaluation with Morning Data

Four-Approach Intersection

Table 6 and Figure 39 show the simulation results for the four-approach intersection based on the morning peak period traffic volume data (Table 3). It can be seen that for all performance criteria, the NFACRL-V control consistently outperforms the pre-timed control, actuated control, and NFACRL-F control. Also, the NFACRL-F control performs better than the pre-timed and actuated control in terms of delay and stopped delay. Compared to the pre-timed control, the NFACRL-F control reduces delay by 6.6 seconds per vehicle, which is 14.7% of the delay resulted from the pre-timed control. Although the NFACRL-F control performs slightly worse in terms of number of stops per vehicle, this could be solved by fine tuning the weight β_5 in Equation (70). For this scenario, actuated control has better performance than pre-timed control. This may be explained by actuated control's ability to adjust green signal length according to real-time traffic flow conditions.

It is expected that the NFACRL-V control has better performance than the NFACRL-F control, as NFACRL-V control is not constrained to any fixed phase sequences and has much greater flexibility to deal with changing traffic flow conditions.

The fact that for most performance criteria both NFACRL methods perform better than the optimized pre-timed and actuated control is very encouraging. It shows that it is feasible to use reinforcement learning in practical intersection traffic control problems with more than two phases, and reinforcement learning can be used to decide when to make phase switch as well as how to choose phase sequence.

TABLE 6 Simulation Results for Four-Approach Intersection Based on Morning Peak Period Data

Model	Statistics	Delay (s/veh)	Stopped Delay (s/veh)	Number of Stops per Veh
NFACRL-F	Average	38.0	25.8	0.90
	Stdev	4.2	3.7	0.04
NFACRL-V	Average	**32.7**	**21.0**	**0.85**
	Stdev	2.7	1.8	0.05
Pre-Timed	Average	44.6	31.4	0.92
	Stdev	3.8	2.7	0.06
Actuated	Average	41.7	29.2	0.89
	Stdev	3.5	2.5	0.05
NFACRL-F vs. Pre-Timed	Improvement	6.6	5.7	0.02
	Percent. Improve. (%)	14.7	18.0	2.7
NFACRL-F vs. Actuated	Improvement	3.7	3.4	-0.01
	Percent. Improve. (%)	8.9	11.8	-0.9
NFACRL-V vs. Pre-Timed	Improvement	11.8	10.5	0.07
	Percent. Improve. (%)	26.6	33.3	7.9
NFACRL-V vs. Actuated	Improvement	9.0	8.2	0.04
	Percent. Improve. (%)	21.6	28.2	4.4

NOTE: $\text{Percent. Improve. (\%)} = \dfrac{\text{Improvement}}{\text{Average value of Pre – Timed or Actuated control}}$

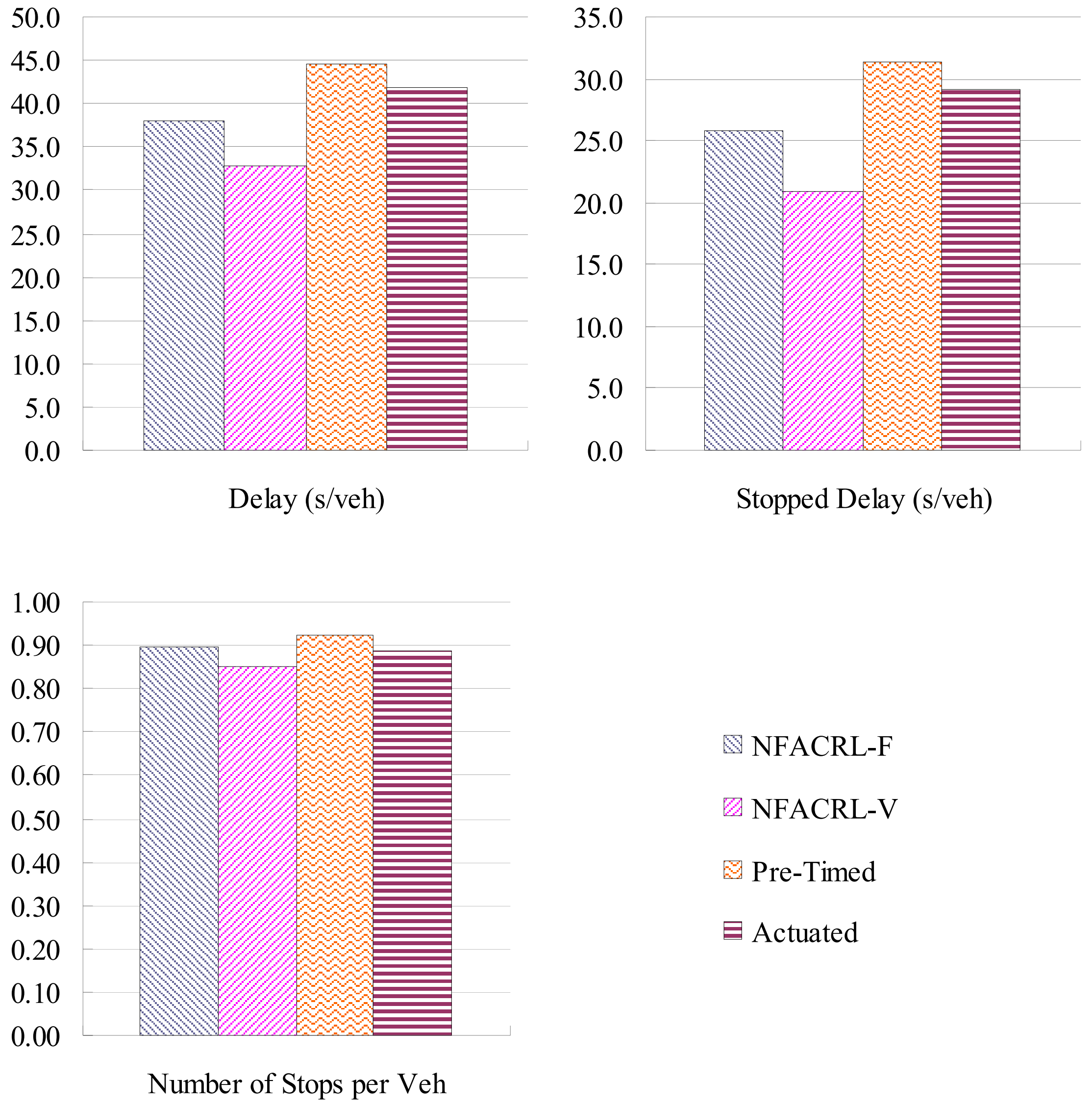


FIGURE 39 Simulation results for four-approach intersection based on morning peak period data.

For each of the 30 evaluation runs, the four control methods (Pre-timed, actuated, NFACRL-F, and NFACRL-V) used the same random seeds, and were evaluated under exactly the same traffic conditions. Paired-*t* tests were conducted to further compare the performance of the four control methods. One-tail paired-*t* test and a significance level of 0.05 were used in this study. The results of paired-*t* tests are presented in Table 7. Data in Tables 6 and 7 show that NFACRL-V control significantly outperforms the pre-timed and actuated control in terms of all three performance criteria. NFACRL-F control significantly reduces delay and stopped delay compared to the optimized pre-timed and actuated control. Although NFACRL-F control slightly increases the number of stops per vehicle compared to the optimized actuated control, the results in Table 7 indicate that this increase is statistically insignificant. Thus, the overall performance of the NFACRL-F control is better than that of the optimized pre-timed and actuated control.

TABLE 7 Paired-*t* Test for Four-Approach Intersection Based on Morning Peak Period Data

Model	Statistics	Delay (s/veh)	Stopped Delay (s/veh)	Number of Stops per Veh
NFACRL-F vs. Pre-Timed	*p*-value	0.000	0.000	0.011
	Comparison	Better	Better	Better.
NFACRL-F vs. Actuated	*p*-value	0.000	0.000	0.193
	Comparison	Better	Better	No Diff.
NFACRL-V vs. Pre-Timed	*p*-value	0.000	0.000	0.000
	Comparison	Better	Better	Better
NFACRL-V vs. Actuated	*p*-value	0.000	0.000	0.001
	Comparison	Better	Better	Better

Three-Approach Intersection

Table 8 and Figure 40 present the simulation results for the three-approach intersection based on the morning peak period traffic data (Table 3), and Table 9 shows the corresponding paired-*t* test results. The general trend shown in Tables 8 and 9 is similar to what has been suggested by the data in Tables 6 and 7. The only difference is that the

NFACRL-F control performs even better in this case, and both NFACRL control methods significantly outperform the optimized pre-timed and actuated control for all three performance criteria, indicated by the improved averages and the paired *t*-test results.

TABLE 8 Simulation Results for Three-Approach Intersection Based on Morning Peak Period Data

Model	Statistics	Delay (s/veh)	Stopped Delay (s/veh)	Number of Stops per Veh
NFACRL-F	Average	15.0	6.9	**0.47**
	Stdev	0.5	0.4	0.01
NFACRL-V	Average	**14.2**	**5.8**	**0.47**
	Stdev	0.8	0.7	0.02
Pre-Timed	Average	16.5	7.3	0.56
	Stdev	1.0	0.5	0.03
Actuated	Average	16.1	7.1	0.55
	Stdev	1.0	0.5	0.03
NFACRL-F vs. Pre-Timed	Improvement	1.6	0.4	0.08
	Percent. Improve. (%)	9.5	5.6	14.8
NFACRL-F vs. Actuated	Improvement	1.1	0.2	0.07
	Percent. Improve. (%)	7.1	2.6	13.3
NFACRL-V vs. Pre-Timed	Improvement	2.3	1.5	0.08
	Percent. Improve. (%)	14.2	20.8	14.9
NFACRL-V vs. Actuated	Improvement	1.9	1.3	0.07
	Percent. Improve. (%)	11.9	18.3	13.3

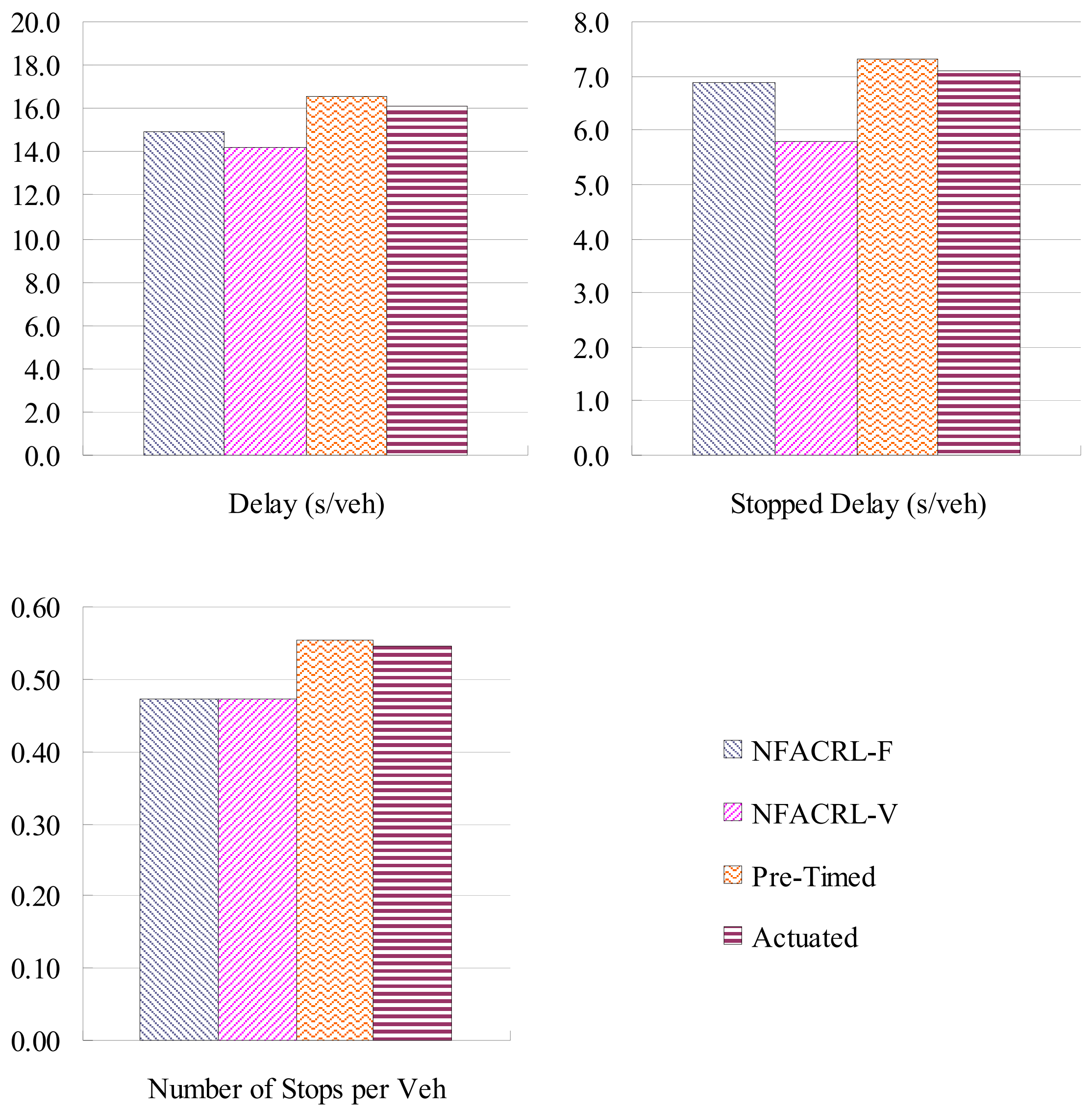


FIGURE 40 Simulation results for three-approach intersection based on morning peak period data.

TABLE 9 Paired-*t* Test for Three-Approach Intersection Based on Morning Peak Period Data

Model	Statistics	Delay (s/veh)	Stopped Delay (s/veh)	Number of Stops per Veh
NFACRL-F vs. Pre-Timed	*p*-value	0.000	0.000	0.000
	Comparison	Better	Better	Better
NFACRL-F vs. Actuated	*p*-value	0.000	0.024	0.000
	Comparison	Better	Better	Better
NFACRL-V vs. Pre-Timed	*p*-value	0.000	0.000	0.000
	Comparison	Better	Better	Better
NFACRL-V vs. Actuated	*p*-value	0.000	0.000	0.000
	Comparison	Better	Better	Better

Evaluation with Noon Data

Four-Approach Intersection

Table 10 and Figure 41 show the simulation results based on the noon peak period traffic data (Table 4) for the four-approach intersection. The NFACRL-V control produces the lowest delay and number of stops per vehicle. The NFACRL-F control also generates lower delay and number of stops per vehicle than the pre-timed and actuated control. In addition, paired-*t* test results in Table 11 show that these improvements on delay and number of stops per vehicle from the NFACRL methods are statistically significant.

While the delay and number of stops per vehicle results show that the two NFACRL methods outperform the optimized pre-time and actuated control, one inconsistency in Tables 10 and 11 indicates that the two NFACRL methods do not perform well on stopped delay. For isolated intersection control, the most important performance evaluation criteria are delay and number of stops per vehicle. Since in this example the delay and numbers of stops per vehicle for the two NFACRL control methods are significantly less than those for the optimized pre-timed and actuated control, the two NFACRL methods can still be considered to be better.

TABLE 10 Simulation Results for Four-Approach Intersection Based on Noon Peak Period Data

Model	Statistics	Delay (s/veh)	Stopped Delay (s/veh)	Number of Stops per Veh
NFACRL-F	Average	18.6	**11.1**	0.62
	Stdev	0.4	0.3	0.02
NFACRL-V	Average	**18.3**	11.9	**0.52**
	Stdev	0.7	0.6	0.02
Pre-Timed	Average	19.9	11.7	0.69
	Stdev	0.8	0.6	0.02
Actuated	Average	19.2	**11.1**	0.68
	Stdev	0.7	0.6	0.02
NFACRL-F vs. Pre-Timed	Improvement	1.3	0.6	0.06
	Percent. Improve. (%)	6.6	5.5	9.2
NFACRL-F vs. Actuated	Improvement	0.6	0.0	0.06
	Percent. Improve. (%)	3.2	0.4	8.6
NFACRL-V vs. Pre-Timed	Improvement	1.6	-0.2	0.17
	Percent. Improve. (%)	8.0	-1.4	24.4
NFACRL-V vs. Actuated	Improvement	0.9	-0.8	0.16
	Percent. Improve. (%)	4.6	-6.9	23.8

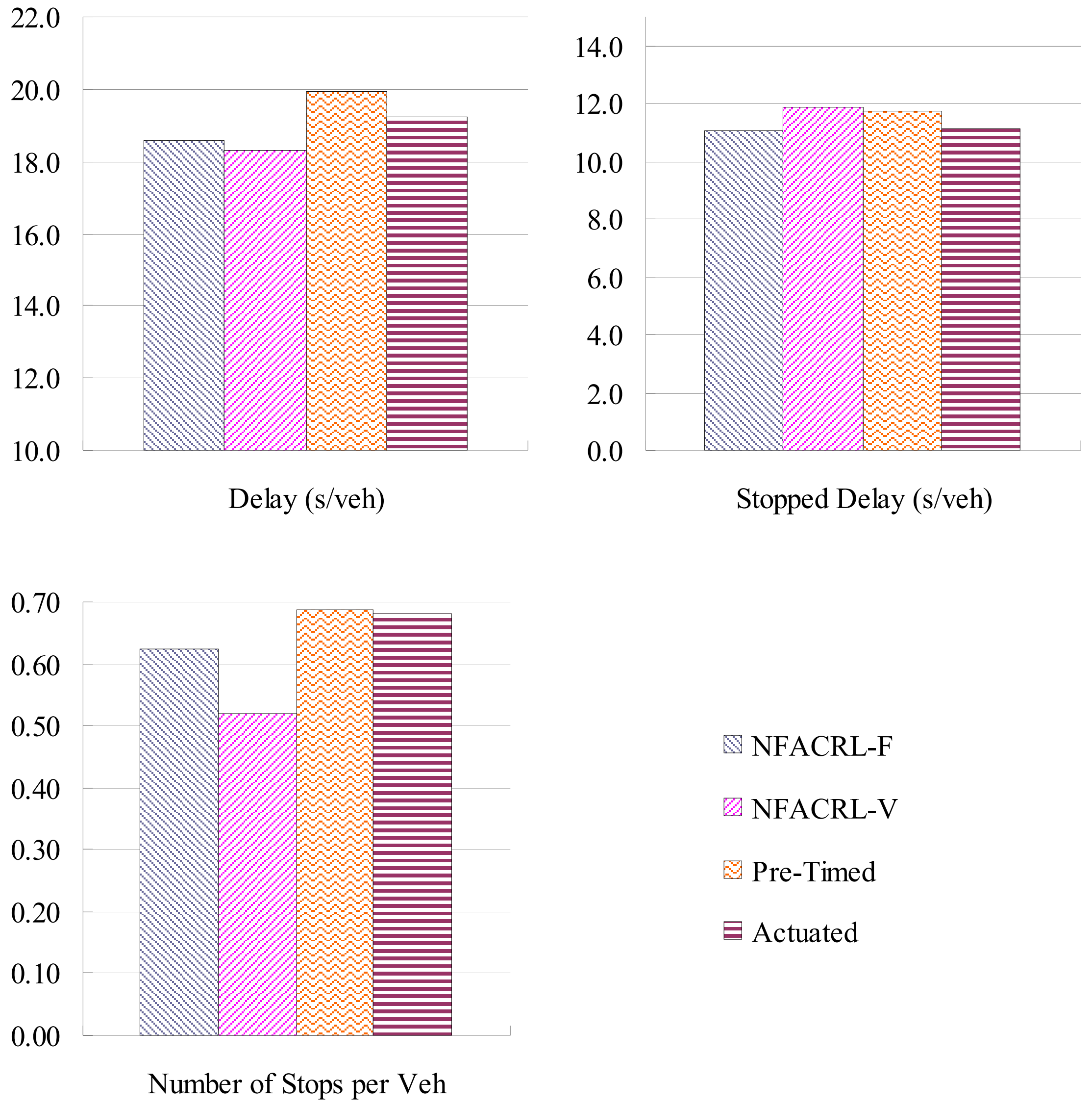


FIGURE 41 Simulation results for four-approach intersection based on noon peak period data.

TABLE 11 Paired-*t* Test for Four-Approach Intersection Based on Noon Peak Period Data

Model	Statistics	Delay (s/veh)	Stopped Delay (s/veh)	Number of Stops per Veh
NFACRL-F vs. Pre-Timed	*p*-value	0.000	0.000	0.000
	Comparison	Better	Better	Better
NFACRL-F vs. Actuated	*p*-value	0.000	0.314	0.000
	Comparison	Better	No Diff.	Better
NFACRL-V vs. Pre-Timed	*p*-value	0.000	0.116	0.000
	Comparison	Better	No Diff.	Better
NFACRL-V vs. Actuated	*p*-value	0.000	0.000	0.000
	Comparison	Better	Worse	Better

Three-Approach Intersection

Table 12 and Figure 42 show the simulation results based on the noon peak period traffic volume data (Table 4) for the three-approach intersection, and Table 13 shows the corresponding paired-*t* test results. The results in Tables 12 and 13 are very consistent and show that both NFACRL methods significantly outperform the optimized pre-timed and actuated control in terms of all performance criteria. Also, the results in Tables 12 and 13 show that the NFACRL-V control produces the best results for all performance criteria.

TABLE 12 Simulation Results for Three-Approach Intersection Based on Noon Peak Period Data

Model	Statistics	Delay (s/veh)	Stopped Delay (s/veh)	Number of Stops per Veh
NFACRL-F	Average	8.2	2.8	0.33
	Stdev	0.8	0.3	0.03
NFACRL-V	Average	**7.1**	**2.4**	**0.26**
	Stdev	0.6	0.3	0.03
Pre-Timed	Average	9.5	3.6	0.38
	Stdev	0.3	0.2	0.01
Actuated	Average	8.9	3.4	0.37
	Stdev	0.3	0.2	0.01
NFACRL-F vs. Pre-Timed	Improvement	1.3	0.9	0.06
	Percent. Improve. (%)	14.0	24.2	14.6
NFACRL-F vs. Actuated	Improvement	0.8	0.6	0.04
	Percent. Improve. (%)	8.7	18.3	10.5
NFACRL-V vs. Pre-Timed	Improvement	2.3	1.3	0.12
	Percent. Improve. (%)	24.7	35.5	31.0
NFACRL-V vs. Actuated	Improvement	1.8	1.0	0.10
	Percent. Improve. (%)	20.1	30.6	27.7

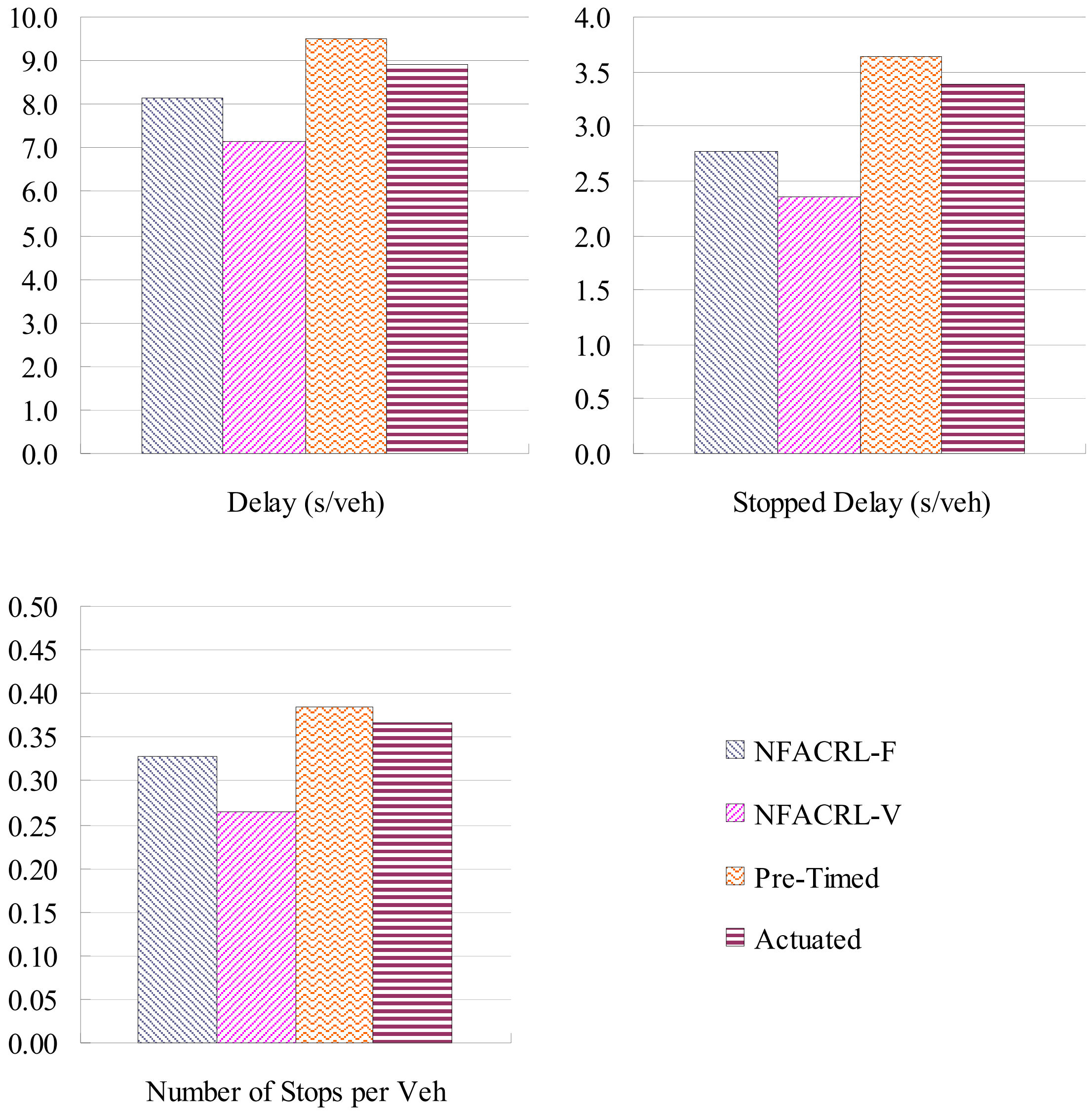


FIGURE 42 Simulation results for three-approach intersection based on noon peak period data.

TABLE 13 Paired-*t* Test for Three-Approach Intersection Based on Noon Peak Period Data

Model	Statistics	Delay (s/veh)	Stopped Delay (s/veh)	Number of Stops per Veh
NFACRL-F vs. Pre-Timed	*p*-value	0.000	0.000	0.000
	Comparison	Better	Better	Better
NFACRL-F vs. Actuated	*p*-value	0.000	0.000	0.000
	Comparison	Better	Better	Better
NFACRL-V vs. Pre-Timed	*p*-value	0.000	0.000	0.000
	Comparison	Better	Better	Better
NFACRL-V vs. Actuated	*p*-value	0.000	0.000	0.000
	Comparison	Better	Better	Better

Evaluation with Afternoon Data

Four-Approach Intersection

Table 14 and Figure 43 show the simulation results based on the afternoon peak period traffic volume data (Table 5) for the four-approach intersection. The data in Table 14 suggest that both NFACRL methods perform better than the optimized pre-timed and actuated control for all three criteria. More specifically, compared to the pre-timed and actuated control, the NFACRL-F control reduces delay by 9.0 and 6.7 seconds per vehicle, respectively. The corresponding improvements from the NFACRL-V control are even greater, at 12.3 and 9.9 seconds, respectively. The paired-*t* test results in Table 15 indicate that the improvements from the two NFACRL methods relative to the optimized pre-timed and actuated control are statistically significant.

TABLE 14 Simulation Results for Four-Approach Intersection Based on Afternoon Peak Period Data

Model	Statistics	Delay (s/veh)	Stopped Delay (s/veh)	Number of Stops per Veh
NFACRL-F	Average	36.6	25.2	**0.85**
	Stdev	1.5	1.3	0.02
NFACRL-V	Average	**33.4**	**21.6**	0.86
	Stdev	2.5	2.0	0.04
Pre-Timed	Average	45.7	31.1	1.01
	Stdev	5.3	3.3	0.10
Actuated	Average	43.3	29.5	0.97
	Stdev	4.7	3.1	0.08
NFACRL-F vs. Pre-Timed	Improvement	9.0	5.9	0.15
	Percent. Improve. (%)	19.8	18.9	15.3
NFACRL-F vs. Actuated	Improvement	6.7	4.2	0.11
	Percent. Improve. (%)	15.4	14.3	11.8
NFACRL-V vs. Pre-Timed	Improvement	12.3	9.5	0.14
	Percent. Improve. (%)	26.9	30.5	14.0
NFACRL-V vs. Actuated	Improvement	9.9	7.8	0.10
	Percent. Improve. (%)	22.9	26.6	10.5

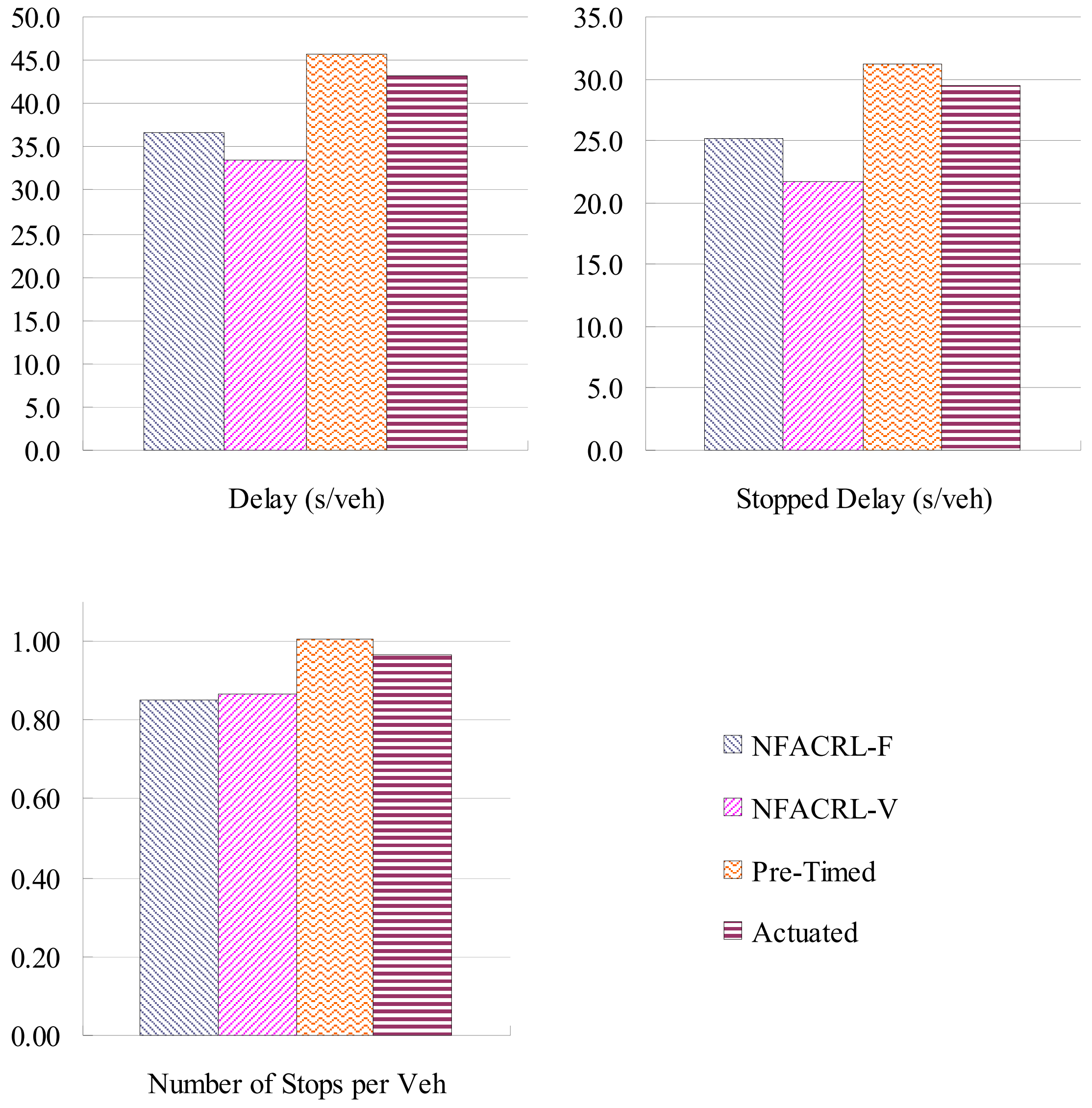


FIGURE 43 Simulation results for four-approach intersection based on afternoon peak period data.

TABLE 15 Paired-*t* Test for Four-Approach Intersection Based on Afternoon Peak Period Data

Model	Statistics	Delay (s/veh)	Stopped Delay (s/veh)	Number of Stops per Veh
NFACRL-F vs. Pre-Timed	*p*-value	0.000	0.000	0.000
	Comparison	Better	Better	Better
NFACRL-F vs. Actuated	*p*-value	0.000	0.000	0.000
	Comparison	Better	Better	Better
NFACRL-V vs. Pre-Timed	*p*-value	0.000	0.000	0.000
	Comparison	Better	Better	Better
NFACRL-V vs. Actuated	*p*-value	0.000	0.000	0.000
	Comparison	Better	Better	Better

Three-Approach Intersection

Table 16 and Figure 44 present the simulation results based on the afternoon peak period traffic volume data (Table 5) for the three-approach intersection. The results show that for all three evaluation criteria the NFACRL-F control outperforms the optimized pre-timed and actuated control, and the NFACRL-V control again performs better than the NFACRL-F control in terms of all three criteria. The paired-*t* test results in Table 17 show that compared to the optimized pre-timed and actuated control, all the improvements from the two NFACRL methods are statistically significant.

TABLE 16 Simulation Results for Three-Approach Intersection Based on Afternoon Peak Period Data

Model	Statistics	Delay (s/veh)	Stopped Delay (s/veh)	Number of Stops per Veh
NFACRL-F	Average	11.1	4.5	0.37
	Stdev	0.4	0.4	0.02
NFACRL-V	Average	**10.4**	**4.3**	**0.32**
	Stdev	0.6	0.7	0.01
Pre-Timed	Average	13.3	5.9	0.45
	Stdev	0.7	0.5	0.02
Actuated	Average	13.3	6.0	0.45
	Stdev	0.9	0.6	0.02
NFACRL-F vs. Pre-Timed	Improvement	2.2	1.4	0.07
	Percent. Improve. (%)	16.6	24.2	16.6
NFACRL-F vs. Actuated	Improvement	2.2	1.5	0.08
	Percent. Improve. (%)	16.3	24.6	16.9
NFACRL-V vs. Pre-Timed	Improvement	3.0	1.6	0.13
	Percent. Improve. (%)	22.2	27.3	29.0
NFACRL-V vs. Actuated	Improvement	2.9	1.7	0.13
	Percent. Improve. (%)	22.0	27.7	29.2

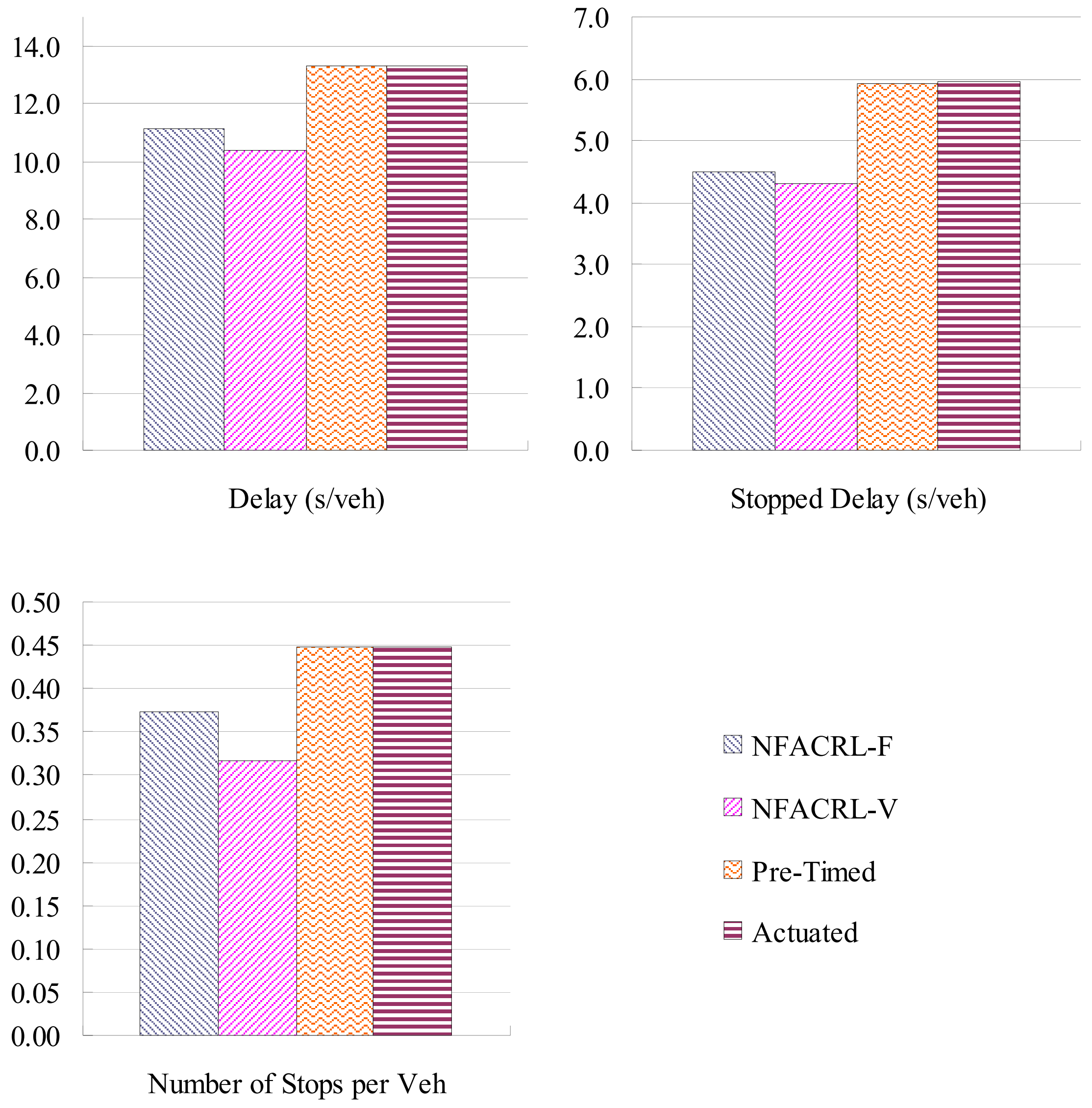


FIGURE 44 Simulation results for three-approach intersection based on afternoon peak period data.

TABLE 17 Paired-*t* Test for Three-Approach Intersection Based on Afternoon Peak Period Data

Model	Statistics	Delay (s/veh)	Stopped Delay (s/veh)	Number of Stops per Veh
NFACRL-F vs. Pre-Timed	*p*-value	0.000	0.000	0.000
	Comparison	Better	Better	Better
NFACRL-F vs. Actuated	*p*-value	0.000	0.000	0.000
	Comparison	Better	Better	Better
NFACRL-V vs. Pre-Timed	*p*-value	0.000	0.000	0.000
	Comparison	Better	Better	Better
NFACRL-V vs. Actuated	*p*-value	0.000	0.000	0.000
	Comparison	Better	Better	Better

Summary and Comparison of Performance during Morning, Noon, and Afternoon Peak Periods

The evaluation results on isolated intersections show that in general the NFACRL-F and NFACRL-V control perform significantly better than the optimized pre-timed and actuated control, and the NFACRL-V control outperforms the NFACRL-F control in most cases. For isolated intersection control, delay and number of stops per vehicle are the two most critical performance criteria. For all six scenarios considered in this dissertation, the NFACRL-V control produces lower delay and lower number of stops per vehicle than the optimized pre-timed and actuated control. The NFACRL-F control generates lower delay and lower number of stops than the optimized pre-timed and actuated control in all cases except for the morning peak period for the four-approach intersection. In this case, the NFACRL-F control generates approximately the same number of stops per vehicle compared to the optimized actuated control.

As delay is one of the most critical criteria for isolated intersection performance evaluation, it is interesting to further examine the relationship between the delay reductions from the two NFACRL methods and traffic volume levels. The delay reductions from the two NFACRL methods relative to the pre-timed control are plotted in Figures 45 and 46. Also plotted are the corresponding total traffic volumes during morning, noon, and afternoon peak periods. Figure 45 shows that for the four-approach intersection, the total traffic volumes during morning and afternoon peak periods are much higher than that of noon peak period. Interestingly, the delay reductions from the two NFACRL methods relative to the pre-timed control during the morning and afternoon periods are also more significant than that during the noon peak period.

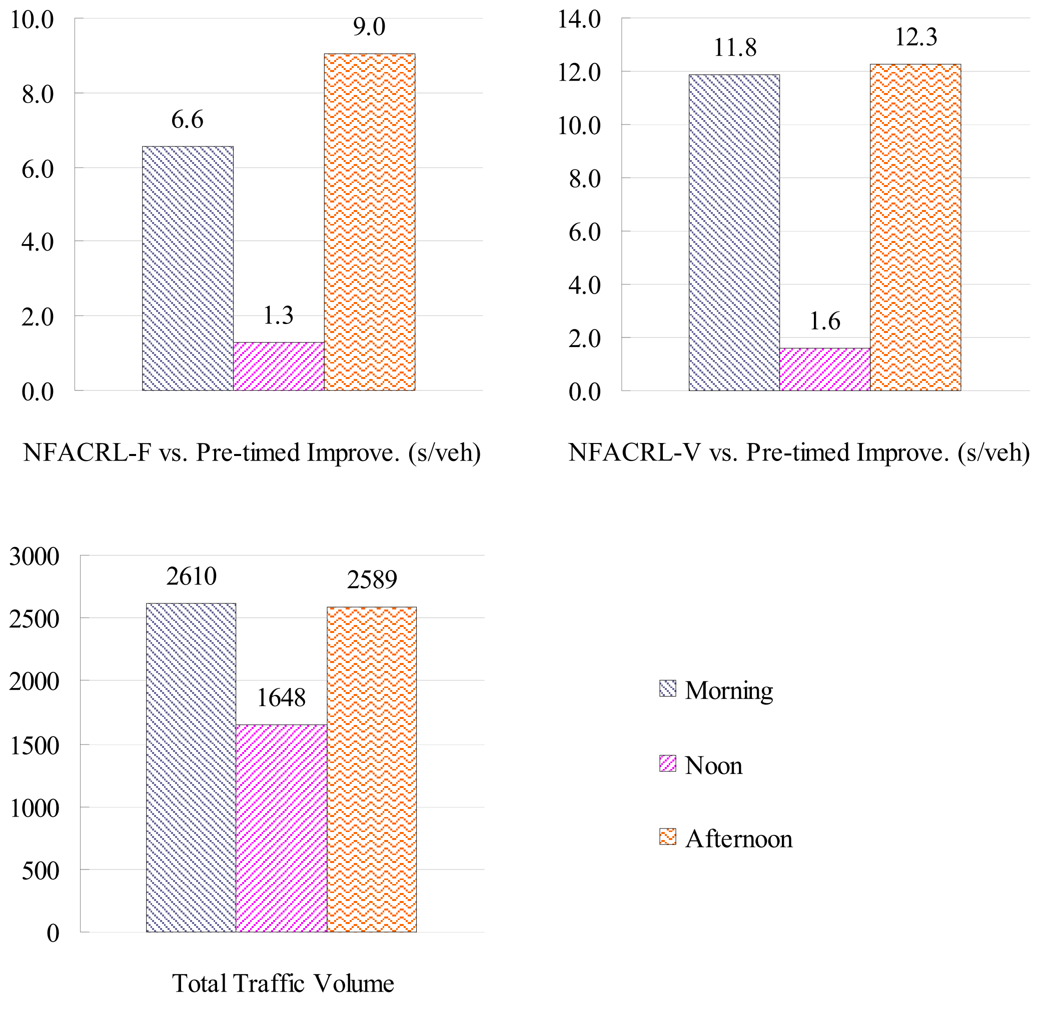


FIGURE 45 Delay improvements from the NFACRL methods relative to the pre-timed control and corresponding traffic volumes for the four-approach intersection.

Figure 46 shows the comparison results for the three-approach intersection. The traffic volume data in Figure 46 show similar trend as the traffic volume data in Figure 45 suggest. Again, the delay reductions from the two NFACRL methods relative to the pre-timed method in general are larger during morning and afternoon peak periods, though for the NFACRL-V method the relative delay reductions during morning and noon peak periods are virtually the same.

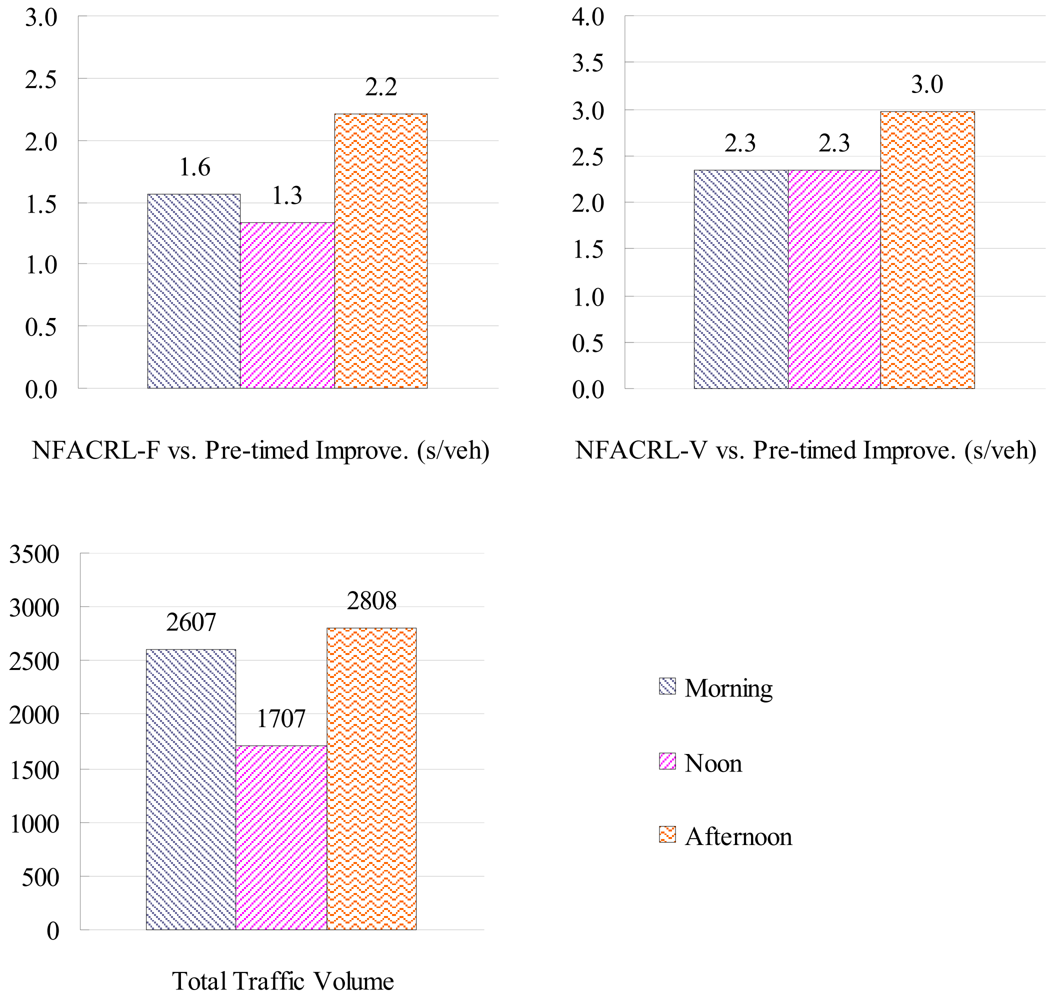


FIGURE 46 Delay improvements from the NFACRL methods relative to the pre-timed control and corresponding traffic volumes for the three-approach intersection.

The results in Figures 45 and 46 seem to suggest that for isolated intersection traffic control the benefits of using the proposed NFACRL methods are more significant when traffic volume is higher.

PERFORMANCE EVALUATION ON ARTERIAL

The proposed NFACRL-F and NFACRL-V methods were also evaluated on the entire arterial network shown in Figure 31. Again, the NFACRL-F and NFACRL-V controllers were trained 90 runs based on the arterial traffic data during morning, noon, and afternoon peak periods. The trained NFACRL controllers were then applied to the arterial for performance evaluation. For each of the three different traffic flow conditions (morning, noon, and afternoon), the performance evaluation process was repeated 30 runs with different random seeds. In addition to delay, stopped delay, and number of stops per vehicle, three new performance criteria were considered for performance evaluations on arterial. These three criteria were speed, throughput, and arterial travel time.

Evaluation with Morning Data

Table 18 and Figure 47 list the simulation results for the arterial based on the morning peak period traffic data. Paired *t*-test was also performed and the results are provided in Table 19. The results show that the NFACRL-V method performs the best. It significantly improves all performance criteria over the pre-timed control and significantly improves all performance criteria but throughput over the actuated control.

Results in Table 18 and Figure 47 suggest that the NFACRL-F method can effectively reduce delay and increase speed. However, it does not perform well on number of stops per vehicle and arterial travel time compared to the optimized coordinated pre-timed and actuated control. One reason for this phenomenon is that the coordinated pre-timed and actuated control strategies use fixed cycle length and offset to maximize the green signal bandwidth along the arterial. This is like giving vehicles traveling along the arterial higher priority. Thus the number of stops for vehicles traveling along the arterial and the arterial travel time can be significantly reduced. While for the NFACRL-F method, vehicles from arterial and cross streets are treated the same. Another possible reason is that the fixed phase sequence restricts NFACRL-F method's ability to minimize number of stops. It may often happen that a platoon of vehicles is coming from the arterial and there are only a few vehicles waiting in the cross streets, and the arterial directions cannot be given green signal immediately due to the phase sequence restriction and minimum green times that have to be served for the cross-street movements.

TABLE 18 Simulation Results for Arterial Based on Morning Peak Period Data

Model	Statistics	Speed (mph)	Delay (s/veh)	Stopped Delay (s/veh)	Number of Stops per Veh	Throughput (# of Veh.)	Arterial Travel Time (s)
NFACRL-F	Average	25.1	55.9	**31.5**	1.52	4104	212.1
	Stdev	0.4	2.3	1.7	0.05	10	3.1
NFACRL-V	Average	**25.6**	**52.9**	32.1	**1.25**	4104	**186.1**
	Stdev	0.4	1.9	1.5	0.04	9	2.5
Pre-Timed	Average	23.8	63.6	42.2	1.46	4093	211.3
	Stdev	0.5	3.1	2.0	0.06	11	3.0
Actuated	Average	24.7	58.3	38.4	1.35	**4112**	194.4
	Stdev	0.5	2.8	1.9	0.05	11	2.5
NFACRL-F vs. Pre-Timed	Improvement	1.3	7.7	10.7	-0.06	11	-0.8
	Percent. Improve. (%)	5.4	12.1	25.4	-4.2	0.3	-0.4
NFACRL-F vs. Actuated	Improvement	0.4	2.4	6.9	-0.17	-8	-17.7
	Percent. Improve. (%)	1.6	4.1	18.0	-12.6	-0.2	-9.1
NFACRL-V vs. Pre-Timed	Improvement	1.8	10.7	10.1	0.21	11	25.2
	Percent. Improve. (%)	7.7	16.8	24.0	14.2	0.3	11.9
NFACRL-V vs. Actuated	Improvement	1.0	5.4	6.3	0.10	-8	8.3
	Percent. Improve. (%)	3.9	9.3	16.5	7.3	-0.2	4.3

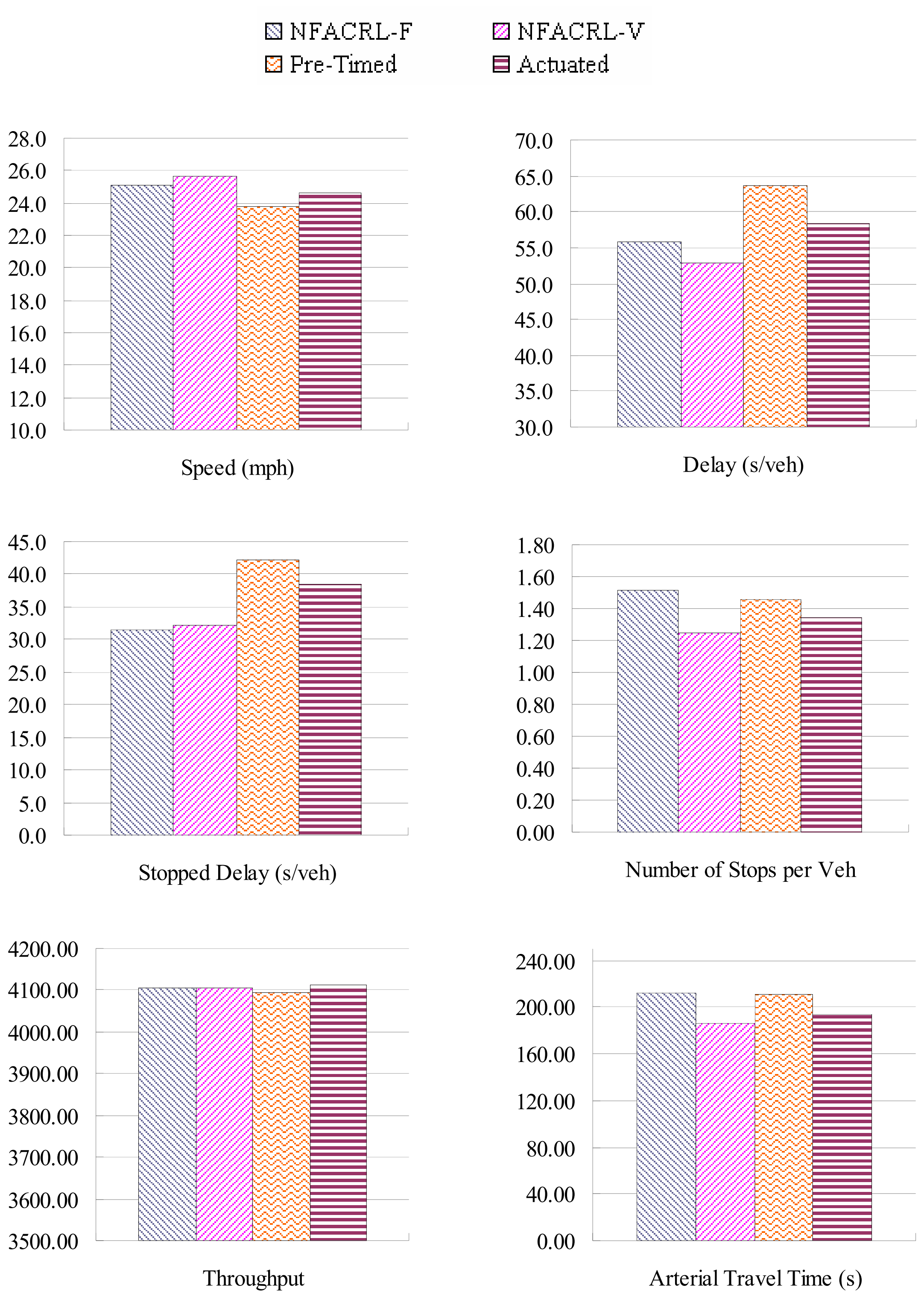


FIGURE 47 Simulation results for arterial based on morning peak period data.

TABLE 19 Paired-*t* Test for Arterial Based on Morning Peak Period Data

Model	Statistics	Speed (mph)	Delay (s/veh)	Stopped Delay (s/veh)	Number of Stops per Veh	Throughput (# of Veh.)	Arterial Travel Time (s)
NFACRL-F vs. Pre-Timed	*p*-value	0.000	0.000	0.000	0.000	0.000	0.139
	Comparison	Better	Better	Better	Worse	Better	No Diff.
NFACRL-F vs. Actuated	*p*-value	0.000	0.000	0.000	0.000	0.000	0.000
	Comparison	Better	Better	Better	Worse	Worse	Worse
NFACRL-V vs. Pre-Timed	*p*-value	0.000	0.000	0.000	0.000	0.000	0.000
	Comparison	Better	Better	Better	Better	Better	Better
NFACRL-V vs. Actuated	*p*-value	0.000	0.000	0.000	0.000	0.000	0.000
	Comparison	Better	Better	Better	Better	Worse	Better

Evaluation with Noon Data

Table 20 and Figure 48 present the simulation results for the arterial based on the noon peak period traffic data. It is easy to see that the NFACRL-V method outperforms the coordinated pre-timed and coordinated actuated control for all performance criteria. The paired-*t* test results in Table 21 also indicate that compared to the coordinated pre-timed and coordinated actuated control all improvements from the NFACRL-V method are statistically significant.

The NFACRL-F control in this case produces better speed, delay, stopped delay, and throughput results than the coordinated pre-timed and coordinated actuated control, and these improvements are statistically significant. However, the NFACRL-F control generates larger number of stops per vehicle than the coordinated pre-timed control and longer arterial travel time than the coordinated actuated control. The result from the NFACRL-F control in this case shows the same trend as the result from the NFACRL-F control based on the morning data suggests. It seems that the optimized coordinated pre-timed and coordinated actuated control methods favor the arterial more than the cross streets. Thus, they generally produce good performance along the arterial such as less arterial travel time. However, the overall performance of the system may not be the best. The NFACRL methods give the arterial and the cross streets equal priority. They may not have the best performance for the arterial. However, they usually generate better performance for the entire system than the optimized coordinated pre-timed and coordinated actuated control.

TABLE 20 Simulation Results for Arterial Based on Noon Peak Period Data

Model	Statistics	Speed (mph)	Delay (s/veh)	Stopped Delay (s/veh)	Number of Stops per Veh	Throughput (# of Veh.)	Arterial Travel Time (s)
NFACRL-F	Average	28.3	42.0	**23.0**	1.38	**2881**	196.2
	Stdev	0.2	1.0	0.6	0.04	10	2.6
NFACRL-V	Average	**28.8**	**39.9**	25.0	**1.06**	2877	**168.4**
	Stdev	0.2	1.0	0.9	0.03	10	1.3
Pre-Timed	Average	26.8	49.0	31.0	1.35	2859	200.0
	Stdev	0.3	1.4	0.9	0.02	10	1.7
Actuated	Average	26.3	52.0	28.4	1.67	2855	180.4
	Stdev	0.8	4.1	2.0	0.11	14	1.6
NFACRL-F vs. Pre-Timed	Improvement	1.5	7.0	8.0	-0.03	22	3.8
	Percent. Improve. (%)	5.4	14.3	25.7	-2.3	0.8	1.9
NFACRL-F vs. Actuated	Improvement	2.0	10.0	5.4	0.29	26	-15.8
	Percent. Improve. (%)	7.6	19.3	19.0	17.3	0.9	-8.7
NFACRL-V vs. Pre-Timed	Improvement	1.9	9.0	6.0	0.29	18	31.6
	Percent. Improve. (%)	7.2	18.5	19.3	21.6	0.6	15.8
NFACRL-V vs. Actuated	Improvement	2.5	12.1	3.4	0.61	22	12.0
	Percent. Improve. (%)	9.4	23.2	12.0	36.6	0.8	6.7

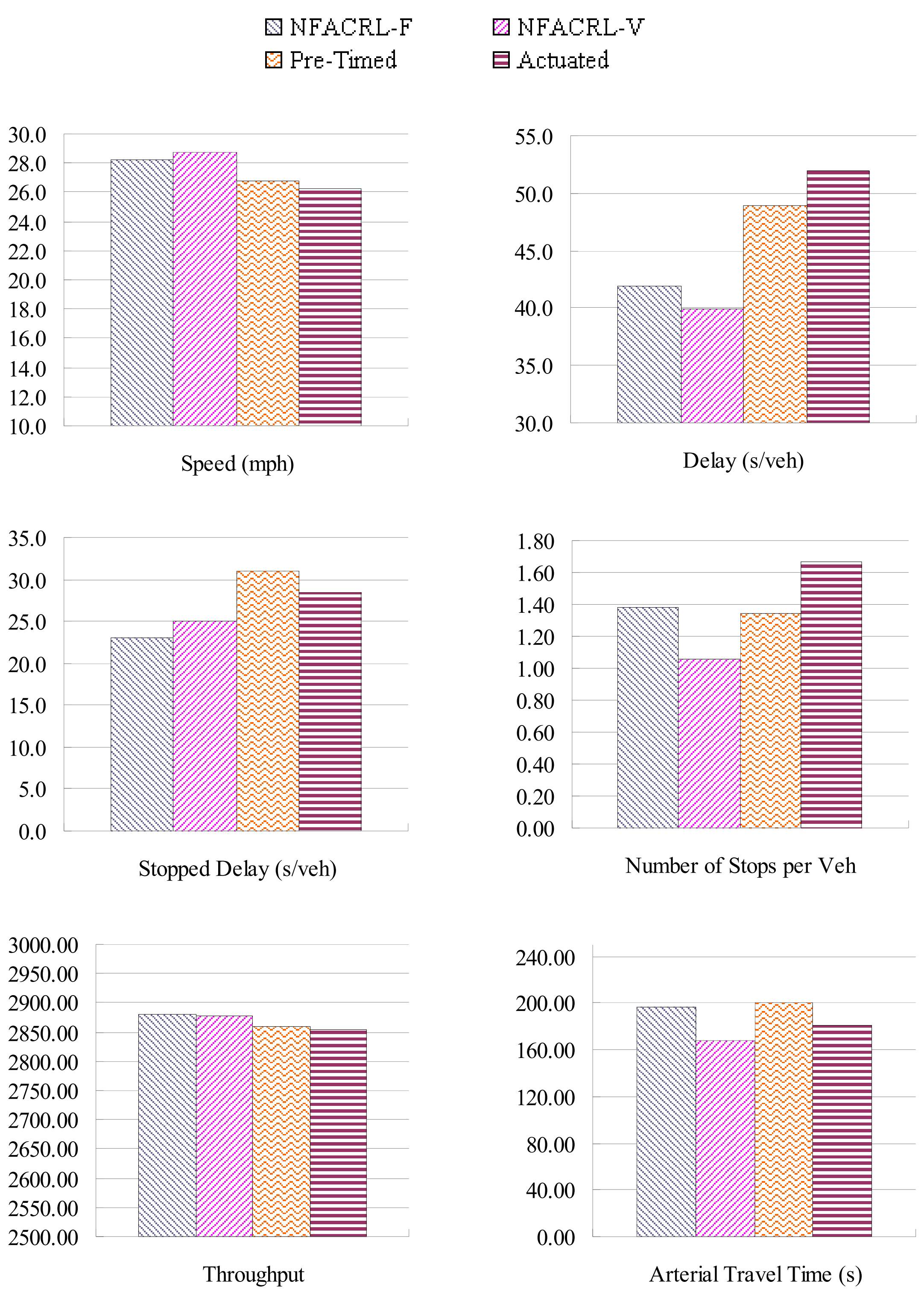


FIGURE 48 Simulation results for arterial based on noon peak period data.

TABLE 21 Paired-*t* Test for Arterial Based on Noon Peak Period Data

Model	Statistics	Speed (mph)	Delay (s/veh)	Stopped Delay (s/veh)	Number of Stops per Veh	Throughput (# of Veh.)	Arterial Travel Time (s)
NFACRL-F vs. Pre-Timed	*p*-value	0.000	0.000	0.000	0.000	0.000	0.000
	Comparison	Better	Better	Better	Worse	Better	Better
NFACRL-F vs. Actuated	*p*-value	0.000	0.000	0.000	0.000	0.000	0.000
	Comparison	Better	Better	Better	Better	Better	Worse
NFACRL-V vs. Pre-Timed	*p*-value	0.000	0.000	0.000	0.000	0.000	0.000
	Comparison	Better	Better	Better	Better	Better	Better
NFACRL-V vs. Actuated	*p*-value	0.000	0.000	0.000	0.000	0.000	0.000
	Comparison	Better	Better	Better	Better	Better	Better

Evaluation with Afternoon Data

Table 22 and Figure 49 present the simulation results for the arterial based on the afternoon peak period traffic data. In this scenario, the NFACRL-V method consistently performs the best in terms of all performance criteria. The paired-*t* test results in Table 23 also suggest that compared to the coordinated pre-timed and coordinated actuated control all improvements from the NFACRL-V method are statistically significant.

The data in Table 23 also show that, compared to the coordinated pre-timed and coordinated actuated control, the NFACRL-F control produces better or approximately the same results in terms of speed, delay, stopped delay, and throughput. However, the NFACRL-F control generates larger number of stops per vehicle and longer arterial travel time than both the coordinated pre-timed and coordinated actuated control.

TABLE 22 Simulation Results for Arterial Based on Afternoon Peak Period Data

Model	Statistics	Speed (mph)	Delay (s/veh)	Stopped Delay (s/veh)	Number of Stops per Veh	Throughput (# of Veh.)	Arterial Travel Time (s)
NFACRL-F	Average	24.1	62.3	37.5	1.66	4365	208.7
	Stdev	0.3	2.2	1.7	0.04	15	2.2
NFACRL-V	Average	**25.2**	**56.1**	**35.7**	**1.31**	**4368**	**179.6**
	Stdev	0.4	2.2	1.9	0.03	15	1.8
Pre-Timed	Average	23.9	63.9	40.4	1.45	4359	189.4
	Stdev	0.7	4.5	2.8	0.08	16	2.4
Actuated	Average	24.2	62.0	39.9	1.43	4359	181.5
	Stdev	0.7	4.3	2.8	0.08	15	2.1
NFACRL-F vs. Pre-Timed	Improvement	0.2	1.6	2.8	-0.20	7	-19.4
	Percent. Improve. (%)	1.0	2.5	7.1	-14.0	0.2	-10.2
NFACRL-F vs. Actuated	Improvement	-0.1	-0.3	2.4	-0.23	6	-27.2
	Percent. Improve. (%)	-0.3	-0.5	5.9	-16.0	0.1	-15.0
NFACRL-V vs. Pre-Timed	Improvement	1.3	7.8	4.7	0.14	10	9.8
	Percent. Improve. (%)	5.5	12.3	11.6	9.9	0.2	5.2
NFACRL-V vs. Actuated	Improvement	1.0	5.9	4.2	0.12	9	1.9
	Percent. Improve. (%)	4.1	9.6	10.6	8.4	0.2	1.1

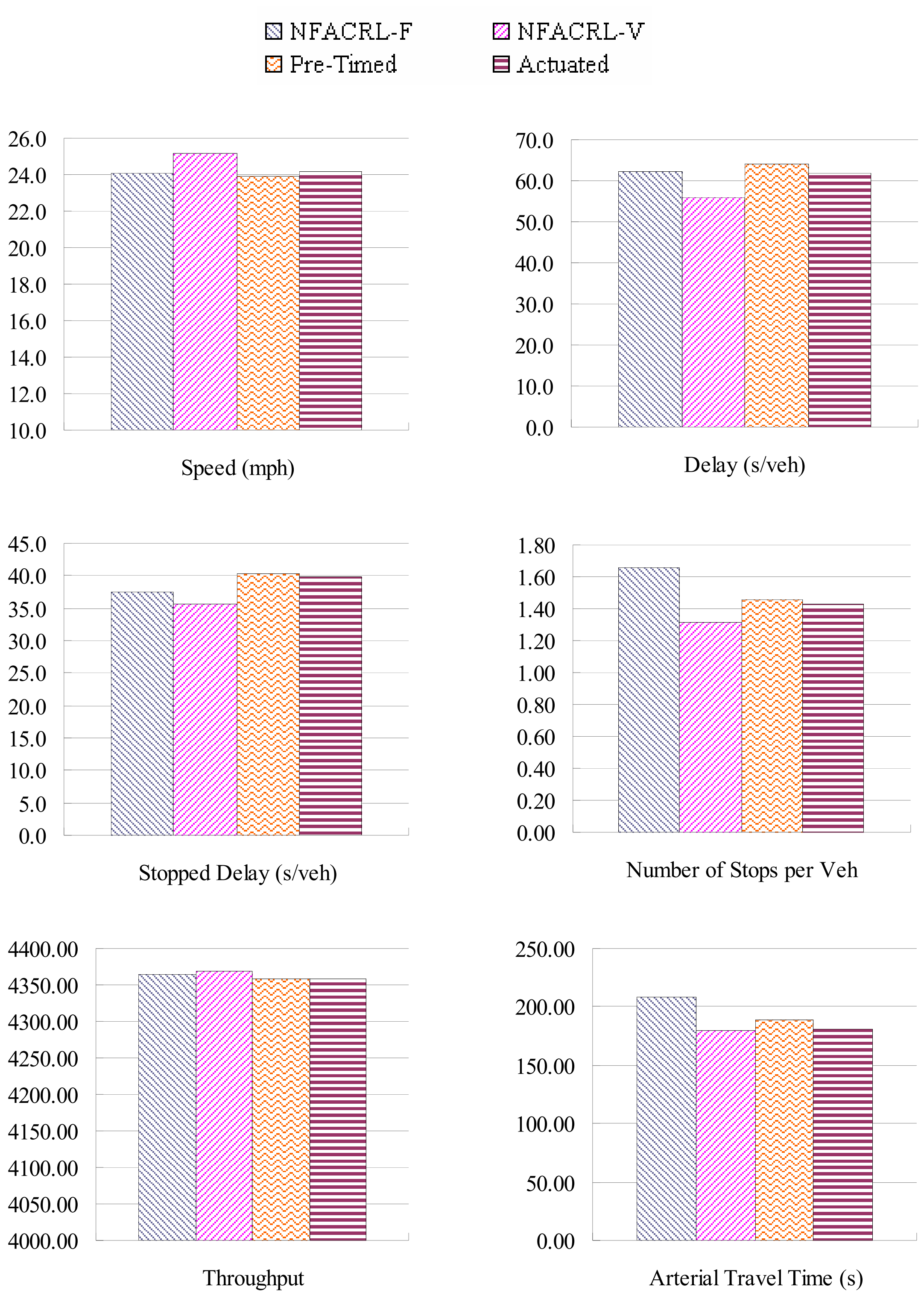


FIGURE 49 Simulation results for arterial based on afternoon peak period data.

TABLE 23 Paired-*t* test for Arterial Based on Afternoon Peak Period Data

Model	Statistics	Speed (mph)	Delay (s/veh)	Stopped Delay (s/veh)	Number of Stops per Veh	Throughput (# of Veh.)	Arterial Travel Time (s)
NFACRL-F vs. Pre-Timed	*p*-value	0.023	0.018	0.000	0.000	0.004	0.000
	Comparison	Better	Better	Better	Worse	Better	Worse
NFACRL-F vs. Actuated	*p*-value	0.268	0.328	0.000	0.000	0.008	0.000
	Comparison	No Diff.	No Diff.	Better	Worse	Better	Worse
NFACRL-V vs. Pre-Timed	*p*-value	0.000	0.000	0.000	0.000	0.001	0.000
	Comparison	Better	Better	Better	Better	Better	Better
NFACRL-V vs. Actuated	*p*-value	0.000	0.000	0.000	0.000	0.001	0.000
	Comparison	Better	Better	Better	Better	Better	Better

Summary and Comparison of Performance during Morning, Noon, and Afternoon Peak Periods

The paired-*t* test results in Tables 19, 21, and 23 show that, for all tests on the arterial, the NFACRL-V method produces better results than the coordinated pre-timed and coordinated actuated control for almost all performance criteria. The only exception is the throughput value for the test on morning peak period data, which is slightly less than that of the coordinated actuated control. The difference between the two throughputs is 8 vehicles, which is only 0.2% of the throughput resulted from the coordinated actuated control. In the meantime, the NFACRL-V control reduces delay by 5.4 seconds per vehicle, which is 9.3% of the delay resulted from the coordinated actuated control.

For the NFACRL-F control, the paired-*t* test results suggest that it generally performs better on delay, speed, stopped delay, and throughput compared to the coordinated pre-timed and coordinated actuated control. However, it does not perform well on number of stops per vehicle and arterial travel time.

To further compare the performance of the proposed NFACRL methods during morning, noon, and afternoon peak periods, the delay reductions from the two NFACRL methods relative to the coordinated pre-timed control during different peak periods are plotted in Figure 50. Also plotted are the total entrance traffic volume data during these three peak periods.

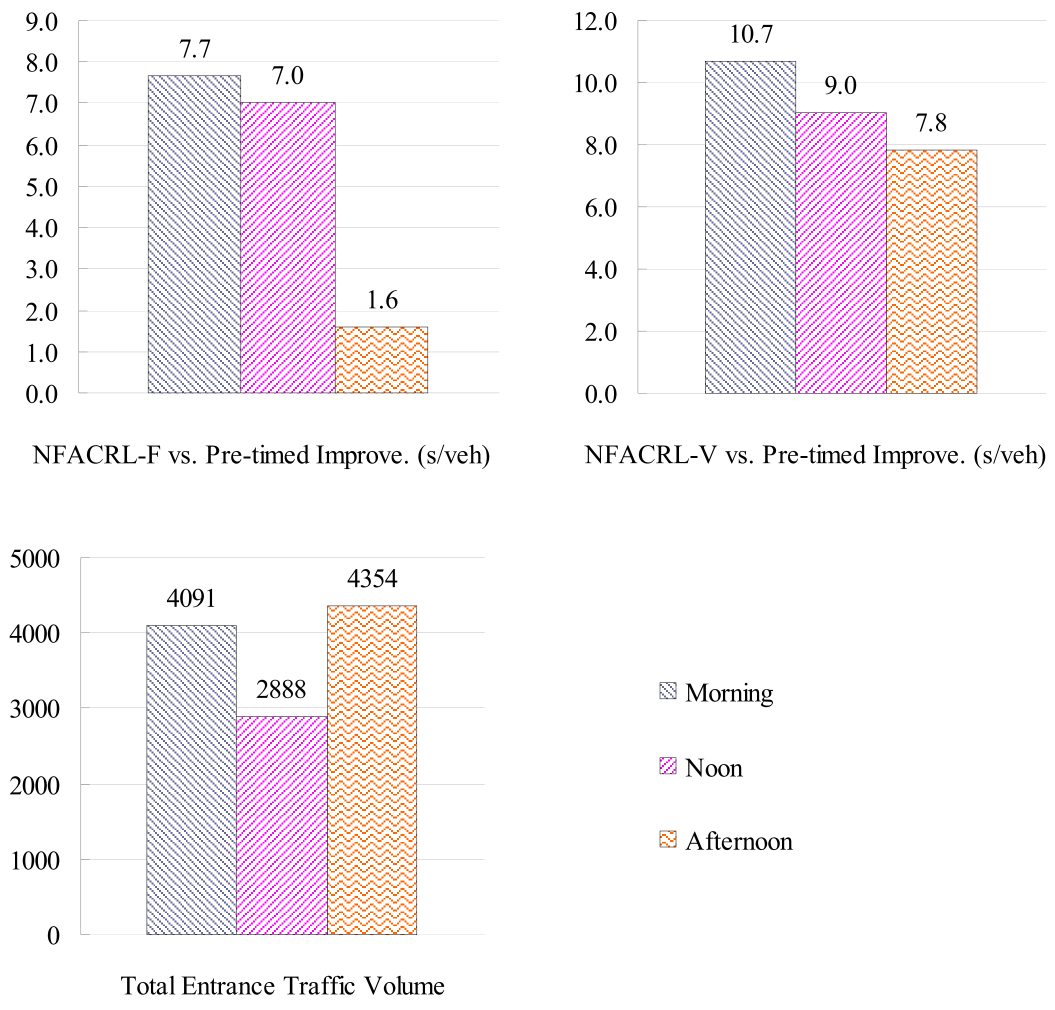


FIGURE 50 Delay improvements from the NFACRL methods relative to the coordinated pre-timed control and corresponding traffic volumes for the arterial.

From Figure 50, it seems that there is no direct connection between the total entrance traffic volume and the relative delay reductions from the two NFACRL methods.

A close look at the traffic volume data in Tables 3 through 5 suggests that the performance of the NFACRL methods relative to the coordinated pre-timed control may depend on the proportion of cross-street traffic and cross-street turning traffic. Cross-street traffic and cross-street turning traffic are defined in Figure 51 and Figure 52, respectively.

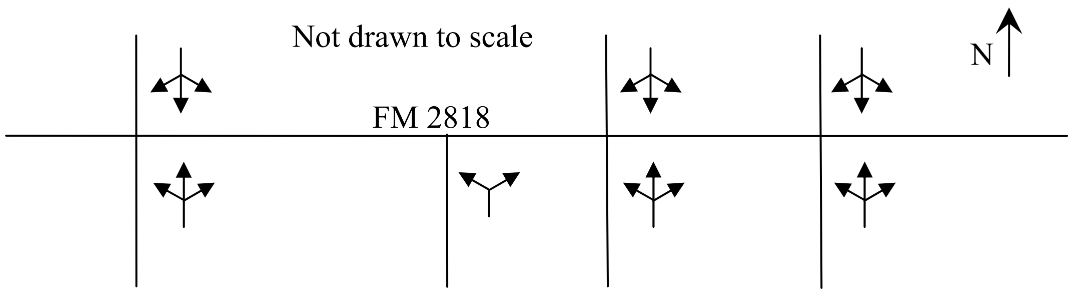


FIGURE 51 Cross-street traffic.

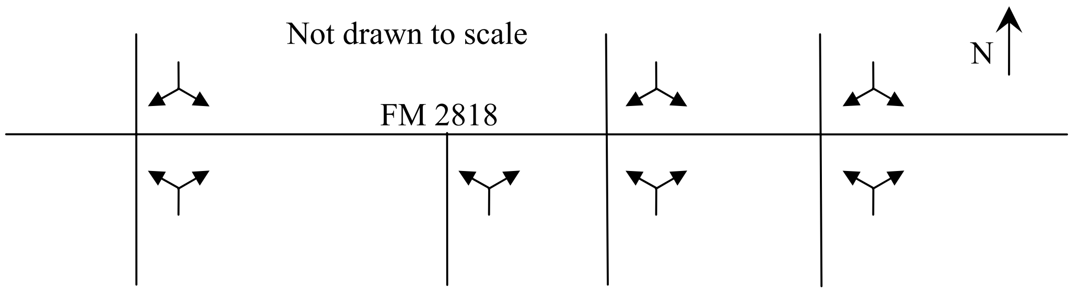


FIGURE 52 Cross-street turning traffic.

The cross-street and cross-street turning traffic data are listed in Table 24. It shows that during morning peak period 66.2% and 49.7% of the total entrance traffic are cross-street traffic and cross-street turning traffic, respectively. While for afternoon peak period only 54.7% and 38.0% of the total entrance traffic are cross-street traffic and cross-street turning traffic, respectively. Higher percentages of cross-street traffic and cross-street turning traffic may result in narrower green signal bandwidth along the arterial, thus the benefits of coordinating different traffic signal controllers may become smaller. Also, the large amount of traffic turning into the arterial can considerably affect the queue lengths of the arterial direction at each intersection. The initial queues may have negative impact on the performance of the coordinated pre-timed and coordinate actuated control. This probably is the reason why during the morning peak period the relative improvements from the two NFACRL methods are more significant.

TABLE 24 Cross Street Traffic during Morning, Noon, and Afternoon Peak Periods

	Morning	Noon	Afternoon
Total entrance traffic (A)	4091	2888	4354
Total cross-street traffic (B)	2710	1624	2383
(B/A)*100	66.2	56.2	54.7
Total cross-street turning traffic (C)	2035	1268	1656
(C/A)*100	49.7	43.9	38.0

SUMMARY

In this chapter, the proposed NFACRL-F and NFACRL-V control schemes were evaluated at two isolated intersections and on the entire arterial. The evaluations were performed using VISSIM simulation based on geometric and traffic data collected from a four-intersection arterial (FM 2818) in College Station, Texas. To better assess the performance of the proposed new methods under different traffic demand conditions, the evaluations were conducted using traffic data during morning, noon, and afternoon peak periods. Optimized pre-timed and actuated control methods were also evaluated to be compared with the two NFACRL methods. For each of the four control strategies (NFACRL-F, NFACRL-V, pre-timed, and actuated), the evaluation process was repeated 30 times with different random seeds.

A four-approach intersection and a three-approach intersection in the arterial were selected for isolated intersection testing. The testing results showed that in almost all cases, the two NFACRL control methods produced lower delay, stopped delay, and number of stops per vehicle compared to the optimized pre-timed and actuated control. Paired-*t* tests were also conducted and showed that all the improvements from the two NFACRL methods were statistically significant. Although the two NFACRL methods produced slightly larger stopped delay for the four-approach intersection using the noon peak period data, they were still considered to perform better than the optimized pre-timed and actuated control due to the considerably reduced delay and number of stops per vehicle. Further comparison of the evaluation results based on morning, noon, and afternoon peak periods showed that the performance of the two NFACRL methods relative to that of the pre-timed control seemed to be more significant when traffic volume was higher.

The NFACRL-F and NFACRL-V control schemes were also extended for arterial control. The results showed that the NFACRL-V control significantly outperformed the coordinated pre-timed and coordinated actuated control for speed, delay, stopped delay, number of stops per vehicle, and arterial travel time. The NFACRL-V control also produced good throughput results in most cases. The NFACRL-F control exhibited less flexibility than the NFACRL-V control. However, in most cases the NFACRL-F control still generated better delay, speed, stopped delay, and throughput results compared to the optimized coordinated pre-timed and coordinated actuated control. The results also showed that the NFACRL-F control did not perform well on number of stops per vehicle and arterial travel time compared to the coordinated pre-timed and coordinated actuated control. Possible reasons for this were discussed, and future studies are needed to address this problem. The performance of the NFACRL methods during the morning, noon, and afternoon peak periods were also compared and analyzed. The result suggested that the benefits of using the NFACRL methods for arterial control may be even larger with higher proportions of cross-street traffic and cross-street turning traffic.

CHAPTER VI

SUMMARY AND CONCLUSIONS

CONTRIBUTIONS

This research investigates the application of reinforcement learning to adaptive traffic signal control. A new adaptive traffic signal control method based on neuro-fuzzy actor-critic reinforcement learning (NFACRL) is developed and evaluated at both isolated intersection and arterial. Compared to previous studies using reinforcement learning for traffic signal control, this research has the following contributions:

1. A comprehensive review of existing traffic signal control methods and reinforcement learning is presented in this dissertation. This review systematically points out the connection between MDP, dynamic programming, and reinforcement learning. It also explains clearly the advantages of modeling traffic signal control as a MDP problem and using reinforcement learning methods to solve it.
2. By introducing the NFACRL, the curse of dimensionality and generalization problems associated with traditional reinforcement learning methods can be properly solved.
3. Bingham (*5*) also combined fuzzy logic and reinforcement learning for traffic signal control. In her study, the fuzzy rules need to be prespecified explicitly. When the state space is large, there could be several hundreds of fuzzy rules. Specifying so many fuzzy rules is very cumbersome. In the proposed NFACRL method, action weights are introduced and there is no need to define each fuzzy rule.
4. Most previous studies considered traffic signal control with only two phases, and phase sequence optimization was not investigated. In practice, typical intersections are controlled by four or even more phases. To show the potential

of applying reinforcement learning control methods in the real world, a fixed (NFACRL-F) phase sequence control strategy and a variable (NFACRL-V) phase sequence control strategy are proposed in this research. Four-phase control and three-phase control are used for the four-approach and three-approach intersections, respectively. This is the first time that complicated and realistic phase configuration is considered in truly adaptive traffic signal control based on reinforcement learning. Also, it is the first time that reinforcement learning is used for phase sequence selection.

5. Various strategies for coordinating agents in a multiagent system are reviewed and their pros and cons are discussed in details. Finally, a simple but robust independent-agent strategy is adopted to coordinate different NFACRL-F and NFACRL-V controllers.
6. A new reward function is proposed in this research. The new reward function takes into account multiple factors such as delay and number of stops, and has been shown to perform well in this research. Theoretical reasons for choosing this new reward function are also provided.
7. A comprehensive comparison of the proposed NFACRL control methods with pre-timed and actuated control is conducted based on VISSIM simulation. Most previous studies did not use a commonly-used microscopic traffic simulation platform for performance evaluations and did not compare their methods with optimized pre-timed or actuated control. Self-developed simulation platforms give users more control over the simulation process, but the simulated traffic environment may not be as close to the true traffic condition as those from commonly-used simulation tools such as VISSIM. Also, pre-timed and actuated control strategies are the two most widely used control methods in practice. It is necessary to show that the proposed new methods are better than them.

MAJOR FINDINGS

There are three major findings in this dissertation. First, this dissertation shows that it is feasible to apply reinforcement learning to adaptive isolated intersection control with more than two phases and complicated phase configurations. For all tests on isolated intersections, the proposed NFACRL-F and NFACRL-V methods produce considerably less delay than the optimized pre-timed and actuated control. In most cases, the two NFACRL methods generate significantly smaller stopped delay and number of stops per vehicle. The performance of the NFACRL-F and NFACRL-V control during morning, noon, and afternoon peak periods are also compared. The comparison result suggests that the performance of the NFACRL-F and NFACRL-V control relative to the optimized pre-timed control tend to be larger when traffic demand is higher.

Secondly, it is found in this dissertation that reinforcement learning can be used for phase sequence selection. In fact, the NFACRL-V method with phase sequence selection ability consistently outperforms the NFACRL-F method with fixed phase sequence for most performance evaluation criteria in this research. As only two phases were considered in most previous studies, none of them investigated the phase sequence selection using reinforcement learning.

Lastly, this research shows that reinforcement learning has the potential to be used for realistic arterial adaptive traffic signal control. The proposed NFACRL-F and NFACRL-V control strategies are applied to the control of a four-intersection arterial network based on VISSIM simulation. A simple but robust independent-agent coordination strategy is considered, in which control agents learn to coordinate with each other implicitly. The evaluation results show that both NFACRL methods can effectively increase overall network speed and reduce delay and stopped delay compared to the optimized coordinated pre-timed and coordinated actuated control. In addition, the NFACRL-V control exhibits more flexibility and produces significantly better performance in terms of number of stops per vehicle and arterial travel time, compared to the optimized coordinated pre-timed and coordinated actuated control. It is also found that the NFACRL-F method does not perform well for criteria such as number of stops per vehicle and arterial travel time. This could be the problem of the parameters used in the reward function or the definition of state variables. Overall, the test results show that the proposed NFACRL-F and NFACRL-V methods are promising tools for isolated intersection and arterial adaptive traffic signal control.

FUTURE RESEARCH

Although encouraging results are obtained in this research, further studies are still needed to address the following problems:

1. In this research, the decision interval is either 3 or 7 seconds. If a green extension decision is made, the next decision point is 3 seconds later. In other words, the green extension is 3 seconds; if a termination decision is made, the next decision point would be 7 seconds later. Because, in addition to 3-second minimum green time for the next phase, a 3-second yellow time and a 1-second all-red time need to be considered to ensure safety. In future studies, smaller green extensions such as 2 seconds can be considered, this should further improve the performance of the NFACRL methods.
2. The coordination strategy considered in this research is fairly simple. Even with this simple coordination strategy, arterial control using the NFACRL-V method still performs better than the coordinated pre-timed and coordinated actuated control in all cases. The NFACRL-F method also outperforms the coordinated pre-timed and coordinated actuated control in many cases in terms of speed, delay, and stopped delay. However, the NFACRL-F control does not perform well for number of stops per vehicle and arterial travel time. One possible solution to this problem is to add four new state variables representing upstream traffic arrival information. With advanced traffic arrival information, the NFACRL-F control should be able to better adjust its control strategy such that the number of stops and arterial travel time can be reduced.

3. A number of factors can affect the performance of the NFACRL control methods, for instance, the fuzzy membership function, learning rate β (Equations (53) and (54)), choice of state variables, ε in the action selection method (Equation (23)), and β_i in the reward function (Equation (70)). Due to limited computation resources and considerable amount of time spent on algorithm developing and debugging, a comprehensive evaluation of the effects of various factors on the NFACRL control performance is not conducted in this dissertation. In future studies, a comprehensive evaluation needs to be conducted.
4. In this dissertation, the proposed NFACRL methods are only applied to isolated intersection and arterial control. By using the same coordination strategy adopted in this research, it would be interesting to see how the NFACRL methods perform on a signalized street network.
5. In this research, it is assumed that queue lengths can be observed accurately, and there are no pedestrians. To apply the proposed NFACRL method in the real world, it is necessary to conduct future studies that take pedestrians into account. Also, an accurate and reliable queue detection method needs to be developed.

REFERENCES

1 Meyer, M. D. *A Toolbox for Alleviating Traffic Congestion and Enhancing Mobility*. Publication IR-054B. Institute of Transportation Engineers (ITE), Washington, D.C., 1997.

2 Schrank, D., and T. Lomax. *The 2005 Urban Mobility Report*. Texas Transportation Institute, College Station, Texas, 2005.

3 FHWA, Department of Transportation. *Improving Traffic Signal Operations: A Primer*. www.itsdocs.fhwa.dot.gov/JPODOCS/REPTS_TE/13466.pdf. Accessed May 28, 2007.

4 Li, H. *Traffic Adaptive Control for Isolated, Over-Saturated Intersections*. Ph.D. Dissertation. Department of Civil Engineering, University of Hawaii, Honolulu, Hawaii, 2002.

5 Bingham, E. Reinforcement Learning in Neurofuzzy Traffic Signal Control. *European Journal of Operational Research*, Vol. 131, No. 2, 2001, pp. 232-241.

6 Zhang, L., H. Li, and P. D. Prevedouros. Signal Control for Oversaturated Intersections Using Fuzzy Logic. Presented at 84th Annual Meeting of the Transportation Research Board. Washington, D.C., 2005.

7 Mirchandani, P., and L. Head. A Real-Time Traffic Signal Control System: Architecture, Algorithms, and Analysis. *Transportation Research Part C*, Vol. 9, No. 6, 2001, pp. 415-432.

8 Gartner, N. H. OPAC: A Demand-Responsive Strategy for Traffic Signal Control. In *Transportation Research Record: Journal of the Transportation Research Board, No. 906*, TRB, National Research Council, Washington, D.C., 1983, pp. 75-81.

9 Crites, R. H., and A. G. Barto. Elevator Group Control Using Multiple Reinforcement Learning Agents. *Machine Learning*, Vol. 33, No. 2-3, 1998, pp. 235-262.

10 Park, K. H., Y. J. Kim, and J. H. Kim. Modular Q-learning Based Multi-Agent Cooperation for Robot Soccer. *Robotics and Autonomous Systems*, Vol. 35, No. 2, 2001, pp. 109-122.

11 Swaminathan, J. M., S. F. Smith, and N. M. Sadeh. Modeling Supply Chain Dynamics: A Multiagent Approach. *Decision Sciences*, Vol. 29, No. 3, 1998, pp. 607-632.

12 Janssens, D., Y. Lan, G, Wets, and G. Chen. Allocating Time and Location Information to Activity–Travel Patterns through Reinforcement Learning. *Knowledge-Based Systems*, Vol. 20, No. 5, 2007, pp. 466-477.

13 Vengerov, D. A Reinforcement Learning Approach to Dynamic Resource Allocation. *Engineering Applications of Artificial Intelligence*, Vol. 20, No. 3, 2007, pp. 383-390.

14 Liu, F., G. S. Ng, and C. Quek. RLDDE: A Novel Reinforcement Learning-Based Dimension and Delay Estimator for Neural Networks in Time Series Prediction. *Neurocomputing*, Vol. 70, No. 7-9, 2007, pp. 1331-1341.

15 Vlassis, N. A Concise Introduction to Multiagent Systems and Distributed AI, 2003. staff.science.uva.nl/~vlassis/cimasdai/cimasdai.pdf. Accessed December 30, 2006.

16 Sutton, R. S., and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, Massachusetts. 1998.

17 Abbas, M. M. *A Real Time Offset Transitioning Algorithm for Coordinating Traffic Signals*. Ph.D. Dissertation. Department of Civil Engineering, Purdue University, West Lafayette, Indiana, USA, 2001.

18 Lan, C. J. New Optimal Cycle Length Formulation for Pretimed Signals at Isolated Intersections. *Journal of Transportation Engineering*, Vol. 130, No. 5, 2004, pp. 637-647.

19 Webster, F. V. *Traffic Signal Settings*. Road Research Technical Paper No. 39. Department of Scientific and Industrial Research, Road Research Laboratory, London, U.K., 1958.

20 *Highway Capacity Manual 2000*. Transportation Research Board, National Research Council, Washington, D. C., 2000.

21 Morgan, J. T., and J. D. C. Little. Synchronizing Traffic Signals for Maximal Bandwidth. *Operations Research*, Vol. 12, No. 6 (Special Transportation Science Issue), 1964, pp. 896-912.

22 Gartner, N. H., S. F. Assmann, F. Lasaga, and D. L. Hou. MULTIBAND – A Variable-Bandwidth Arterial Progression Scheme. In *Transportation Research*

Record: Journal of the Transportation Research Board, No. 1287, TRB, National Research Council, Washington, D.C., 1990, pp. 212-222.

23 Gartner, N. H., S. F. Assmann, F. Lasaga, and D. L. Hou. A Multi-Band Approach to Arterial Traffic Signal Optimization. *Transportation Research Part B*, Vol. 25, No. 1, 1990, pp. 55-74.

24 Stamatiadis, C., and N. H. Gartner. MULTIBAND-96: A Program for Variable-Bandwidth Progression Optimization of Multiarterial Traffic Networks. In *Transportation Research Record: Journal of the Transportation Research Board, No. 1554*, TRB, National Research Council, Washington, D.C., 1996, pp. 9-76.

25 Roess, R. P., E. S. Prassas, and W. R. McShane. *Traffic Engineering (3rd Edition)*. Pearson Education, Inc., Upper Saddle River, New Jersey, 2004.

26 Hunt, P. B. *A Traffic Responsive Method of Coordinating Signals*. Report No. LR1014. Transport and Road Research Laboratory, Crowthorne, Berkshire, England, 1981.

27 Sims, A. G., and K. W. Dobinson. The Sydney Coordinated Adaptive Traffic (SCAT) System Philosophy and Benefits. *IEEE Transactions on Vehicular Technology*, Vol. VT-29, No. 2, 1980, pp. 130-137.

28 Lowrie, P. R. The Sydney Co-ordinated Adaptive Traffic System – Principles, Methodology, Algorithms. In *Proceedings of the International Conference on Road Traffic Signaling*. London, United Kingdom, 1982, pp. 67-70.

29 Robertson, D. I., and R. D. Bretherton. Optimal Control of an Intersection for Any Known Sequence of Vehicle Arrivals. In *Proceedings of the 2nd IFAC-IFIP-IFORS Symposium of Traffic Control and Transportation Systems*. North-Holland, Amsterdam, Netherlands, 1974, pp. 3-17.

30 Mauro, V., and C. D. Taranto. UTOPIA. In *Proceedings of IFAC Control, Computers, Communications in Transportation*. Paris, France, 1989, pp. 245-252.

31 Donati, F., V. Mauro, G. Roncolini, and M. Vallauri. A Hierarchical-Decentralized Traffic Light Control System. In *Proceedings of IFAC 9th Triennial World Congress*. Budapest, Hungars, 1984, pp. 2853-2858.

32 Henry, J. J., J. L. Farges, and J. Tuffal. The PRODYN Real Time Traffic Algorithm. In *Proceedings of IFAC Control in Transportation Systems*. Baden-Baden, Federal Republic of Germany, 1983, pp. 305-310.

33 Porche, I. R. *Dynamic Traffic Control: Decentralized and Coordinated Methods*. Ph.D. Dissertation. Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, Michigan, 1997.

34 Yu, X. H., and W. W. Recker. Stochastic Adaptive Control Model for Traffic Signal Systems. *Transportation Research Part C*, Vol. 14, No. 4, 2006, pp. 263-282.

35 Gartner, N. H. Prescription for Demand-Responsive Urban Traffic Control. In *Transportation Research Record: Journal of the Transportation Research Board, No. 881*, TRB, National Research Council, Washington, D.C., 1982, pp. 73-76.

36 Gartner, N. H., C. Stamatiadis, and F. J. Tarnoff. Development of Advanced Traffic Signal Control Strategies for Intelligent Transportation Systems: Multilevel Design. In *Transportation Research Record: Journal of the Transportation Research Board, No. 1494*, TRB, National Research Council, Washington, D.C., 1995, pp. 98-105.

37 Shelby, S. G. Single-Intersection Evaluation of Real-Time Adaptive Traffic Signal Control Algorithms. In *Transportation Research Record: Journal of the Transportation Research Board, No. 1867*, TRB, National Research Council, Washington, D.C., 2004, pp. 183-192.

38 SCOOT - The World's Leading Adaptive Traffic Control System. Peek Traffic Limited, Siemens Traffic Controls and TRL Limited. www.scoot-utc.com/index.php. Accessed April 27, 2007.

39 Henry, R. D., R. A. Ferlis, and J. L. Kay. *Evaluation of UTCS Control Strategies—Executive Summary*. Publication FHWA-RD-76-149. FHWA, Department of Transportation, 1976.

40 Holroyd, J., and D. I. Robertson. *Strategies for Area Traffic Control Systems Present and Future*. Report No. LR569. Transport and Road Research Laboratory, Crowthorne, Berkshire, England, 1973.

41 Gartner, N. H., F. J. Pooran, and C. M. Andrews. Implementation of the OPAC Adaptive Control Strategy in a Traffic Signal Network. In *Proceedings of the 2001 IEEE Intelligent Transportation Systems Conference*. Oakland, California, 2001, pp. 195-200.

42 Sen, S., and K. L. Head. Controlled Optimization of Phases at An Intersection. *Transportation Science*, Vol. 31, No. 1, 1997, pp. 5-17.

43 Dell'Olmo, P., and P. B. Mirchandani. REALBAND: An Approach for Real-Time Coordination of Traffic Flows on a Network. In *Transportation Research Record:*

Journal of the Transportation Research Board, No. 1494, TRB, National Research Council, Washington, D.C., 1995, pp. 106-116.

44 Butenko, S., 2005. *Class Notes for INEN 623 – Nonlinear and Dynamic Programming*. Department of Industrial and Systems Engineering, Texas A&M University, College Station, Texas, 2005.

45 Davidsson, F., and C. D. Taranto. Priority for Public Transport Using Advanced Transport Telematics. In *Proceedings of the 3rd International Conference on Vehicle Navigation and Information Systems.* Oslo, Norway, 1992, pp. 530-536.

46 Henry, J. J., and J. L. Farges. PRODYN. In *Proceedings of IFAC Control, Computers, Communications in Transportation.* Paris, France, 1989, pp. 253-255.

47 Farges, J. L., L. Khoudour, and J. B. Lesort. PRODYN: On Site Evaluation. In *Proceedings of the 3rd International Conference on Road Traffic Control.* London, United Kingdom, 1990, pp. 62-66.

48 Gosavi, A. Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning. Kluwer Academic Publishers. Norwell, Massachusetts, 2003.

49 Barto, A. G., and S. Mahadevan. Recent Advances in Hierarchical Reinforcement Learning. *Discrete Event Dynamic Systems: Theory and Applications*, Vol. 13, No. 4, 2003, pp. 343-379.

50 Pappis, C. P., and E. H. Mamdani. A Fuzzy Logic Controller for a Traffic Junction. *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 7, No. 10, 1977, pp. 707-717.

51 Trabia, M. B., M. S. Kaseko, and M. Ande. A Two-Stage Fuzzy Logic Controller for Traffic Signals. *Transportation Research Part C,* Vol. 7, No. 6, 1999, pp. 353-367.

52 Murat, Y. S., and E. Gedizlioglu. A Fuzzy Logic Multi-Phased Signal Control Model for Isolated Junctions. *Transportation Research Part C,* Vol. 13, No. 1, 2005, pp. 19-36.

53 Chiu, S., and S. Chand. Adaptive Traffic Signal Control Using Fuzzy Logic. In *Proceedings of 2nd IEEE International Conferences on Fuzzy Systems*. San Francisco, California, 1993, pp. 1371-1376.

54 Niittymaki, J., and M. Pursula. Signal Control Using Fuzzy Logic. *Fuzzy Sets and Systems*, Vol. 116, No. 1, 2000, pp. 11-22.

55 Jiang, J. S. R, C. T. Sun, and E. Mizutani. *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Prentice Hall, Upper Saddle River, New Jersey, 1997.

56 Fuzzy Logic Toolbox 2 User's Guide. The MathWorks, Inc., Natick, MA, USA, 2007. www.mathworks.com/access/helpdesk/help/pdf_doc/fuzzy/fuzzy.pdf. Accessed May 6, 2007.

57 Lin, F. B. Comparative Analysis of Two Logics for Adaptive Control of Isolated Intersections. In *Transportation Research Record: Journal of the Transportation Research Board, No. 1194*, TRB, National Research Council, Washington, D.C., 1988, pp. 6-14.

58 Elahi, S. M., A. E. Radwan, and K. M. Goul. Knowledge-Based System for Adaptive Traffic Signal Control. In *Transportation Research Record: Journal of the Transportation Research Board, No. 1324*, TRB, National Research Council, Washington, D.C., 1987, pp. 115-122.

59 Owen, L. E., and C. M. Stallard. Rule-Based Approach to Real-Time Distributed Adaptive Signal Control. In *Transportation Research Record: Journal of the Transportation Research Board, No. 1683*, TRB, National Research Council, Washington, D.C., 1995, pp. 95-101.

60 Mitchell, T. M. *Machine Learning*. McGraw-Hill. New York, 1997.

61 Kaelbling, L. P., M. L. Littman, and A. W. Moore. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, Vol. 4, 1996, pp. 237-285.

62 Hu, J., and M. P. Wellman. Multiagent Reinforcement Learning: Theoretical Framework and an Algorithm. In *Proceedings of the Fifteenth International Conference on Machine Learning*. Madison, Wisconsin, USA, 1998, pp. 242-250.

63 Puterman, M. L. Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley & Sons, Inc., New York, 1994.

64 Barto, A. G., R. S. Sutton, and C. W. Anderson. Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems. *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 13, No. 5, 1983, pp. 834-846.

65 Gajjar, G. R., S. A. Khaparde, P. Nagaraju, and S. A. Soman. Application of Actor-Critic Learning Algorithm for Optimal Bidding Problem of a Genco. *IEEE Transactions on Power Systems*, Vol. 18, No. 1, 2003, pp. 11-18.

66 Bhatnagar, S., and J. R. Panigrahi. Actor-Critic Algorithm for Hierarchical Markov Decision Processes. *Automatica*, Vol. 42, No. 4, 2006, pp. 637-644.

67 Borkar, V. S. An Actor-Critic Algorithm for Constrained Markov Decision Processes. *System & Control Letters*, Vol. 54, No. 3, 2005, pp 207-213.

68 Berenji, H. R., and P. Khedkar. Learning and Tuning Fuzzy Controllers through Reinforcement. *IEEE Transactions on Neural Networks*, Vol. 3, No. 5, 1992, pp. 724-740.

69 Lin, C. T. and C. S. G. Lee. Reinforcement Structure/Parameter Learning for Neural-Network-Based Fuzzy Logic Control Systems. *IEEE Transactions on Fuzzy Systems*, Vol. 2, No. 1, 1994, pp. 46-63.

70 Thorpe, T. L. Vehicle Traffic Light Control Using SARSA. 1997. www.cs.colostate.edu/~anderson/pubs/thorpems.ps.gz. Accessed December 30, 2006.

71 *CORSIM User's Manual Version 1.03*. Federal Highway Administration, Mclean, VA, 1997.

72 *VISSIM User Manual, Version 4.30*. PTV Planung Transport Verkehr AG, Karlsruhe, Germany, 2007.

73 Abdulhai, B., R. Pringle, and G. J. Karakoulas. Reinforcement Learning for True Adaptive Traffic Signal Control. *Journal of Transportation Engineering*, Vol. 129, No. 3, 2003, pp. 278-285.

74 Berenji, H. R., and P. Khedkar. Learning and Tuning Fuzzy Controllers through Reinforcement. *IEEE Transactions on Neural Networks*, Vol. 3, No. 5, 1992, pp. 724-740.

75 Bingham, E. *Neurofuzzy Traffic Signal Control*. Master Thesis. Department of Engineering Physics and Mathematics, Helsinki University of Technology, Espoo, Finland, 1998.

76 Choy, M. C., 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, 2003, pp. 597-607.

77 Choy M. C., R. L. Cheu, D. Srinivasan, and F.Logi. Real-Time Coordinated Signal Control Using Agents with Online Reinforcement Learning. Presented at 82nd Annual Meeting of the Transportation Research Board. Washington, D.C., 2003.

78 Srinivasan, D., and M. C. Choy. Cooperative Multi-Agent System for Coordinated Traffic Signal Control. *IEE Proceedings - Intelligent Transport Systems*, Vol. 153, No. 1, 2006, pp. 41-50.

79 Ross, T. J. *Fuzzy Logic with Engineering Applications (2nd Edition)*. John Wiley & Sons, Ltd., England, 2004.

80 Rumelhart, D. E., G. E. Hinton, and R. J. Williams. Learning Representations by Back-Propagating Errors. *Nature*, Vol. 323, 1986, pp. 533-536.

81 Jouffe, L. Actor-Critic Learning Based on Fuzzy Inference System. In *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*. Beijing, China, Vol. 1, 1996, pp. 339-344.

82 Jouffe, L. Fuzzy Inference System Learning by Reinforcement Methods. *IEEE Transactions on Systems, Man, and Cybernetics Part C: Applications and Reviews*, Vol. 28, No. 3, 1998, pp. 338-355.

83 Littman, M. L. Friend-or-Foe Q-Learning in General-Sum Games. In *Proceedings of the 18th International Conference on Machine Learning*. Williamstown, Massachusetts, 2001, pp. 322-328.

84 Kalai, E., and E. Lehrer. Rational Learning Leads to Nash Equilibrium. *Econometrica*, Vol. 61, Issue 5, 1993. pp. 1019-1045.

85 Bowling, M., and M. Veloso. Rational and Convergent Learning in Stochastic Games. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, Seattle, Washington, 2001, pp. 1021-1026.

86 Price, B., and C. Boutilier. Implicit Imitation in Multiagent Reinforcement Learning. In *Proceedings of the 16th International Conference on Machine Learning*. Bled, Slovenia, 1999, pp. 325-334.

87 Panait, L., and S. Luke. Cooperative Multi-Agent Learning: The State of the Art. *Autonomous Agents and Multi-Agent Systems*, Vol. 11, 2005, pp. 387-434.

88 Claus, C., and C. Boutilier. The Dynamics of Reinforcement learning in Cooperative Multiagent Systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*. Madison, Wisconsin, 1998, pp.746-752.

89 Chalkiadakis, G., and C. Boutilier. Coordination in Multiagent Reinforcement Learning: A Bayesian Approach. In *Proceedings of the 2nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS-03)*. Melbourne, Australia, 2003, pp. 709-716.

90 Hu, J., and M. P. Wellman. Nash Q-Learning for General-Sum Stochastic Games. *Journal of Machine Learning Research*, Vol. 4, 2003, pp. 1039-1069

91 Bowling, M., and M. Veloso. Multiagent Learning Using a Variable Learning Rate. *Artificial Intelligence*, Vol. 136, No. 2, 2002, pp. 215-250.

92 Shoham, Y., R. Powers, and T. Grenager. Multi-Agent Reinforcement Learning: A Critical Survey. www.cs.ualberta.ca/~bowling/classes/cmput608/ShohamEtAl03.pdf. Accessed May 20, 2007.

93 Littman, M. L. Markov Games as A Framework for Multi-Agent Reinforcement Learning. In *Proceedings of the 11th International Conference on Machine Learning*. San Francisco, California, 1994, pp. 157-163.

94 Tan, M. Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents. In *Proceedings of the 10th International Conference on Machine Learning*. Amherst, Massachusetts, 1993, pp. 330-337.

95 *AIMSUN User Manual Version 4.1.4*. Transportation Simulation Systems, Barcelona, Spain, 2003.

96 Paramics Traffic Simulation Modeler Reference Manual Version 3.0. Quadstone, Ltd, Edinburgh, Scotland, 1999.

97 Birst, S., J. Baker, and E. Shouman. Comparison of Traffic Simulation Models to HCM 2000 Using Various Traffic Levels under Pre-timed Signal Control. Presented at 86th Annual Meeting of the Transportation Research Board. Washington, D.C., 2007.

98 City of College Station GIS files. City of College Station, Texas. www2.cstx.gov/gisdownloads/downloads/County_Streets.zip. Accessed May 20, 2007.

99 PASSER II-02 Version 1.0. Texas Transportation Institute, College Station, Texas. ttisoftware.tamu.edu/fraPasserII_02.htm. Accessed May 20, 2007.

100 PASSER V-03 Version 2.3. Texas Transportation Institute, College Station, Texas. ttisoftware.tamu.edu/fraPasserV_03.htm. Accessed May 20, 2007.

101 Traffic Network Study Tool: TRANSYT-7F, United States Version. McTrans Center, Gainesville, Florida. mctrans.ce.ufl.edu/featured/TRANSYT-7F/. Accessed May 20, 2007.

102 Zhang, Y., and Y. Xie. Comparison of PASSER V, Synchro, and TRANSYT-7F for Arterial Signal Timing Based on CORSIM Simulation. In *Proceedings of the 9th International Conference on Applications of Advanced Technology in Transportation*. Chicago, Illinois, 2006, pp. 479-484.

103 *Synchro 5 User Guide*. Trafficware, Albany, California, 2001.

104 *NEMA Editor Manual Version 5.* PTV America, Inc. ptvamerica.com/nse/manual/Manual_Nema.pdf. Accessed May 20, 2007

105 Zhang, Y. Optimal Traffic Control for a Freeway Corridor under Incident Conditions. Ph.D. Dissertation. Department of Civil Engineering, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 1996.

VITA

Name:	Yuanchang Xie
Address:	200 Charles Haltom Avenue, Apt 4D College Station, TX 77840
Email Address:	xieyc@hotmail.com
Education:	B.S., Transportation Engineering, Southeast University (China), 2000 M.S., Transportation Engineering, Southeast University (China), 2003 Ph.D., Civil Engineering, Texas A&M University, 2007