Reinforcement Learning-based Traffic Signal Control for Signalized Intersections

by

Dunhao Zhong

Thesis submitted to the University of Ottawa
in partial Fulfillment of the requirements for the
Master of Computer Science

School of Electrical Engineering and Computer Science (EECS)
Faculty of Engineering
University of Ottawa

Abstract

Vehicles have become an indispensable means of transportation to ensure people's travel and living materials. However, with the increasing number of vehicles, traffic congestion has become severe and caused a lot of social wealth loss. Therefore, improving the efficiency of transport management is one of the focuses of current academic circles. Among the research in transport management, traffic signal control (TSC) is an effective way to alleviate traffic congestion at signalized intersections.

Existing works have successfully applied reinforcement learning (RL) techniques to achieve a higher TSC efficiency. However, previous work remains several challenges in RL-based TSC methods. First, existing studies used a single scaled reward to frame multiple objectives. Nevertheless, the single scaled reward has lower scalability to assess the controller's performance on different objectives, resulting in higher volatility on different traffic criteria. Second, adaptive traffic signal control provides dynamic traffic timing plans according to unforeseeable traffic conditions. Such characteristic prohibits applying the existing eco-driving strategies whose strategies are generated based on foreseeable and pre-fixed traffic timing plans. To address the challenges, in this thesis, we propose to design a new RL-TSC framework along with an eco-driving strategy to improve the TSC's efficiency on multiple objectives and further smooth the traffic flows.

Moreover, to achieve effective management of the system-wide traffic flows, current researches tend to focus on the design of collaborative traffic signal control methods. However, the existing collaboration-based methods often ignore the impact of transmission delay for exchanging traffic flow information on the system. Inspired by the state-of-the-art max-pressure control in the traffic signal control area, we propose a new efficient RL-based cooperative TSC scheme by improving the reward and state representation based on the max-pressure control method and developing an agent that can address the data transmission delay issue by decreasing the discrepancy between the real-time and delayed traffic conditions. To evaluate the performance of our proposed work more accurately, in addition to the synthetic scenario, we also conducted an experiment based on the real-world traffic data recorded in the City of Toronto. We demonstrate that our method surpassed the performance of the previous traffic signal control methods through comprehensive experiments.

Acknowledgements

I would first like to thank my supervisor, Professor Azzedine Boukerche, whose expertise, patient guidance, and financial support were invaluable in completing my thesis work. His insightful feedback steered me in the right direction and brought my work to a higher level.

I wish to express my deepest gratitude to Dr. Peng Sun for his patient guidance throughout my studies. He provided me with beneficial comments and suggestions on my research and the completion of the manuscript.

In addition, I would like to thank my parents for their love and support throughout my life. Thank you both for giving me the strength and bravery to chase my goal. Also, I wish to express my deepest gratitude to Jade, who provides me with unending inspiration and encouragement in my life.

Publications

1. Azzedine Boukerche, Dunhao Zhong, and Peng Sun. FECO: An Efficient Deep Reinforcement Learning-based Fuel-Economic Traffic Signal Control Scheme. Accepted by *IEEE Transactions on Sustainable Computing*, 2020. Manuscript number: TSUSC-2020-05-0058.

2. Azzedine Boukerche, Dunhao Zhong, and Peng Sun, "A Novel Reinforcement Learning-based Cooperative Traffic Signal System through Max-pressure Control," in *IEEE Transactions on Vehicular Technology* (Early Access), 2021.

3. Dunhao Zhong, Peng Sun, and Azzedine Boukerche. Empirical study and analysis of the impact of traffic flow control at road intersections on vehicle energy consumption. In *Proceedings of the 18th ACM Symposium on Mobility Management and Wireless Access*, pages 21–28, 2020.

4. Dunhao Zhong and Azzedine Boukerche. Traffic signal control using deep reinforcement learning with multiple resources of rewards. In *Proceedings of the 16th ACM International Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks*, pages 23–28, 2019.

5. Azzedine Boukerche and Dunhao Zhong. Traffic Signal Control: Reinforcement Learning-based Traffic Signal Control Algorithms: A Comprehensive Review. Submitted to IEEE Transactions on Intelligent Transportation Systems, 2021.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Traffic congestion is an ever-growing problem due to increasing volume of vehicles. The delay per auto commuter in large cities has increased from 30 hours in 1982 to around 63 hours in 2014 [1]. The problem also brings financial costs to drivers. According to the report of INRIX [2], the cost of a single driver due to traffic congestion in Boston was increased by $2,291 on average in 2017. Due to skyrocketing vehicle volume, the traffic congestion problem has become increasingly severe, leading to higher traffic delays, energy consumption and carbon dioxide emissions, and even more traffic accident events [3]. Therefore, transport requires an intelligent transport management system to alleviate the traffic congestion problem [4]. This chapter first introduces the thesis's motivation and then describes the problems that are focused on in this research. Further, the main contributions are highlighted, and the thesis outlines are described at the end of this chapter.

1.1 Motivation

Intelligent traffic signal control (ITSC) plays a pivotal role in an intelligent transport management system and has been receiving a surge of consideration as a potential solution to address the traffic congestion problem [5, 6]. ITSC is an effective way to improve traffic signal timing plan's efficiency by adjusting the signal timing plan dynamically according to detected traffic conditions [7–9]. Several early ITSC studies have focused on actuated

traffic signal control (TSC) systems at intersections, such as Split Cycle Offset Optimization Technique (SCOOT) [10], Sydney Coordinated Adaptive Traffic System (SCATS) [11] and Real-time Hierarchical Optimizing Distributed Effective System (RHODES) [12]. The designers of the actuated TSC systems design the rules and traffic patterns before applying the system to the intersections [13–16]. After installing the system, the signal controllers match the traffic patterns with the real-time collected traffic information and responsively control signals based on the designed rules. However, actuated TSC methods responsively control signals for the real-time traffic conditions. They do not consider the long-term traffic flows, such as the upstream platoons, resulting in a higher percentage of vehicles stopping [16–19].

To improve the efficiency of TSC for long-term traffic flows, strategies based on artificial intelligence (AI) methods have attracted research interest in recent years. Reinforcement learning (RL) algorithms [20], as a branch of artificial intelligence, have attracted research interest and seen increasing applications in addressing TSC problems [21–23]. The self-learning nature of RL makes it a powerful approach to formulating TSC problems. First, it does not depend on supervised learning labels, which require a large and reliable dataset to train the controller. Moreover, the online adaptive learning characteristic is embedded in the process of model learning [20]. This intrinsic characteristic of RL allows the signal controllers to update their models incrementally with new traffic observations in a dynamic traffic environment.

Several developments of techniques significantly promote the application of RL techniques in TSC problems. More traffic features (vehicle speed, positions, fuel consumption, and emissions, etc.) are available to be collected by detectors and analyzed by signal controllers due to the development of mobile communication techniques in recent years (e.g., GPS, vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication techniques) [24–27]. These substantial features significantly enhance the capabilities of signal controllers to learn the traffic environment and find the optimal global solutions for traffic conditions. Moreover, the new generation of telecommunications 5G mobile communications [28–30] improves the efficiency of data transformation, providing a solid fundamental for signal controllers to efficiently interact with traffic devices and moving vehicles [31–34]. Higher computing power enhances the computational ability of controllers to efficiently learn the environment and policies in a shorter period [35–37].

Therefore, the RL technique is among many essential benefits for solving TSC problems, and it is valuable to investigate the application of RL techniques in TSC to improve traffic control efficiency. In this thesis, we aim to address the problems of applying RL techniques toward TSC and improve TSC efficiency in a realistic traffic environment.

1.2 Problem Statement

1.2.1 Problem Statement for Isolated Intersection

The traffic signal control for an isolated intersection only measures the local intersection's traffic conditions, such as the traffic flows of the connecting lanes. The traffic conditions of the adjacent intersections are not considered in such traffic environments. Many studies have put considerable effort into designing reward models to improve RL-based TSC methods' performance on multi-objectives. The common structure of the reward models is scaling the reward values by combining a set of weighted rewards for different objectives. However, the agent is less sensitive to each individual's behaviours due to the scaled single reward. Furthermore, the pre-set weights of each reward have lower scalability to assess the actions for dynamic traffic states.

Signalized intersections in urban areas increase the frequency of stop-go movements, leading to increased accelerating and decelerating times. Such accelerating and decelerating times consume extra fuel when drivers accelerate or decelerate vehicles hardly [38]. With the development of connected vehicle technology, researchers have been studying on eco-driving strategies to avoid hard acceleration and deceleration by taking the advances of vehicle-to-vehicle (V2V) and vehicle-to-infrastructure(V2I) technologies [39–41]. These eco-driving approaches aim to find the optimal speed profiles and trajectories for approaching vehicles based on the foreseeable traffic signal timing plans, such as pre-fixed traffic signal timing plans. Such characteristic prohibits their applications of the adaptive TSC methods whose traffic signal plans are dynamic.

To address these problems on the isolated intersection, we focus on improving traffic flow regulation stability for multi-objectives and finding an efficient way to control vehicle speed profiles based on the dynamic traffic signal timing plans.

1.2.2 Problem Statement for Multi-intersections

In a traffic road network, the traffic condition at an intersection depends on the signal control of the intersection and is affected by the signal controls and traffic flows of its adjacent intersections [42, 43]. Therefore, the cooperative traffic signal control method considering the associated traffic conditions of those inter-dependent intersections is essential to achieve the system-wide/global optimal signal control.

Many studies have put considerable effort into designing traffic state representations and reward models in formulating the cooperative TSC problem by RL algorithms [44]. However, the associated traffic conditions of multiple intersections have higher dimensions of traffic states than the traffic conditions of an isolated intersection. Concatenating the traffic states of multiple intersections increases exponentially the dimensions of traffic states. It becomes worse in the case of adding more traffic features in the traffic state representations. Furthermore, a higher traffic state dimension requires a higher learning cost in the process of agent learning.

Existing RL-based cooperative TSC methods, such as the Presslight method [45], ignored the data transmission delay issue in their traffic environment. However, the delay of the vehicular networks is inevitable in a real traffic environment [46–51]. The signal controllers are incapable of receiving all traffic conditions in real-time because of some delayed vehicle information. Owing to moving positions and dynamic directions of on-road vehicles, the traffic states generated by delayed information may not be identical to those in real-time traffic conditions [52]. The differences between delayed and real-time traffic states affect the reliability of signal control policies.

To address these problems on multi-intersections, we focus on finding an efficient way to cooperatively control traffic signals in a traffic road network and addressing the data transmission delay issue in a realistic traffic environment.

1.3 Contribution

This thesis comprises the following contributions to improve traffic signal control efficiency:

- We proposed a DRL-based framework for multi-objective using multi-rewards. Each

reward develops a traffic signal control model for the agent to achieve its corresponding objective. Moreover, we proposed an action voting policy for the agent to take the optimal action from a collection of candidates selected by different objectives.

- We proposed a vehicle speed profile control method that can optimize the vehicle's instantaneous velocity and acceleration/deceleration based on our adaptive TSC method.
- To reduce the dimensions of traffic states for multi-intersections, we adopted the max-pressure control method in this research to design a new traffic state representation and reward model.
- To address the data transmission delay issue in a traffic road network, we proposed a state prediction method predicting real-time traffic states based on the delayed vehicle information.
- We designed synthetic traffic scenarios and a realistic traffic scenario with the real-world dataset in Toronto to demonstrate the effectiveness and significance of the thesis work.

1.4 Thesis Outline

The remainder of this thesis is organized as follows: in Chapter 2, we present the literature review of RL-based traffic signal control methods. In Chapter 3, we introduce the preliminaries of this research for a better understanding of our proposed TSC methods. Chapter 4 describes the proposed DRL-based framework for the TSC at an isolated intersection and presents experimental results to demonstrate the effectiveness of the proposed method. Chapter 5 describes the proposed cooperative TSC scheme and presents experimental results to demonstrate the effectiveness in an arterial traffic road network. Finally, Chapter 6 concludes the thesis and points the future work.

Chapter 2

Related Work

This chapter introduces the literature research of reinforcement learning-based traffic signal control (RL-based TSC) methods. Section 2.1 presents the primary designs of RL methods for addressing TSC problems, including traffic state representation, action, and reward models. Section 2.2 summarizes the related RL-based TSC approaches in previous work. We first classify the RL-based TSC methods into model-based and model-free groups and then introduce the cooperative RL-based TSC methods for a traffic road network environment. The state-of-the-art RL-based TSC methods based on deep learning techniques are presented in this chapter for addressing TSC problems in a large scale traffic road network.

2.1 Designs of Reinforcement Learning-based Traffic Signal Control

State, action, and reward designs are critical parts of RL-based traffic signal control methods. The state transforms the traffic environment into a representation that can be identified and learned by the controller agent. The action defines the action space that the agent can execute to impact on the traffic environment. The reward frames the objectives of the agent by evaluating the executed actions. A higher reward indicates that the agent has a higher probability of repeating the same action for a similar state in the future steps. This section presents state, action, and reward literature designs for solving traffic signal control problems.

2.1.1 State Design

State design is the fundamental of RL-based TSC methods, since it describes the environment that can be learned by the controller agent. A state representation contains several traffic measurements, such as vehicle motion data and traffic signal condition, to let the agent know the condition of the traffic environment. Below, we list several critical measurements that are used to represent the state of the traffic environment.

- *Queue Length (QL)*: QL denotes the number of vehicles queueing at a signalized intersection or represents the length of queueing vehicles in meters. A vehicle is counted into queue length when the speed is lower than 0.1m/s.
- *Vehicle Speed (VS)*: VS represents the instantaneous speed of an approaching vehicle.
- *Vehicle Position (VP)*: VP describes the physical positions of vehicles on lanes. VP is usually described as the distance from the vehicle to the stop line of an intersection.
- *Waiting Time (WT)*: WT denotes the total time of a vehicle queueing at an intersection. The time is counted when the speed of a vehicle is lower than 0.1m/s.
- *Traffic Signal Phase (TSP)*: TSP represents the current traffic signal phase of a signalized intersection. A traffic signal phase describes the time process of traffic signals to regulate non-conflict vehicular movements.
- *Traffic Signal Phase Elapsed Time (TSPET)*: TSPET calculates the elapsed time of a traffic signal phase. Longer elapsed time allows the lanes with green signals more time for free traffic flows but longer stop time on the other lanes.
- *Street Density (SD)*: SD denotes the number of vehicles that are inside a corresponding segment, or the percentage of a segment occupied by vehicles. A segment can be a partial area of a street or a whole street.
- *Vehicle Number (VN)*: VN describes the number of vehicles that enter an intersection or a traffic road network. VN can be the total number of moving vehicles running over a period or at the current time step.
- *Throughput*: Throughput denotes the number of vehicles that pass through an intersection or a traffic network during a period.

Based on the traffic measurements above, state representation design can be categorized into vector-based representation and image-based representation. Vector-based state representation consists of traffic measurements by using vectors. A state vector $s = [f_1, f_2, \cdots, f_n]$ consists of several different measurements f to represent the traffic environment. Each measurement f_i can be represented by a sub-vector or a value. Image-based state representation consists of features by a stack of matrices, each containing a single or multiple features. The following sections summarize these two state representations.

Vector-based State Representation

QL of connecting lanes at an intersection is a commonly used features for describing traffic conditions. As an example in [53], a state with the QL of all lanes L is represented as $s = [q_1, q_2, \cdots, q_L]$. In order to reduce the state dimensions, the QL of a lane can be discretized into multiple levels, such as low, medium, and high [54]. The state vector can be extended by adding more traffic measurements. For example, the feature of TSPET can be appended to the state vector to enhance the state representation [55]. The enhanced state is represented as $s = [q_1, q_2, \cdots, q_L, t_{1,red}, t_{2,red}, \cdots, t_{L,red}]$, where $t_{i,red}$ denotes the elapsed time of red signals on the lane i. Another state design is a vector with the size of $2 + P$ [56, 57]. The first two elements are TSP, TSPET, and the last P elements are maximum QL associated with the TSP. These representations collect the overall features of the intersection. It is flexible in removing or adding features to describe the traffic state, but the disadvantage of this design is that the state dimension's size exponentially increases when adding a new feature into the state vector.

Vehicle-based state design is another vector-based state representations [58, 59]. It describes the state of an individual vehicle queueing at an intersection instead of the overall traffic conditions. The state of a vehicle is represented as $[TSP, VP, V_{des}]$, where V_{des} denotes the destination of the vehicle. This representation design significantly reduces the state space from an infeasible size to the number of moving vehicles in the intersection.

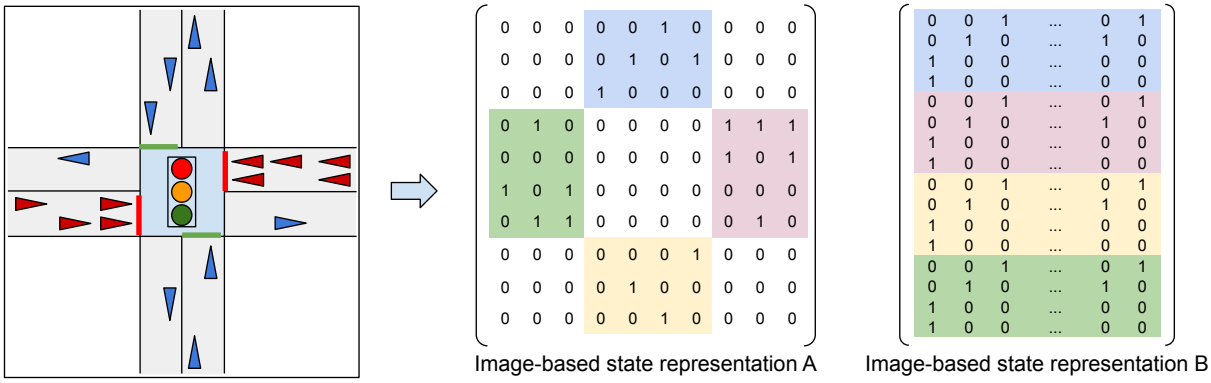


Figure 2.1: Image-based state representations. Representation A creates a matrix to represent the snapshot of the intersection and fills cells with value 1 if a vehicle is present in the cells. Representation B creates a matrix with size $r \times c$, where r denotes the number of connecting lanes, c denotes the maximum capacity of lanes.

Image-based State Representation

Image-based state representation is commonly used in deep RL methods since deep learning techniques can extract features from a broad state representation to a hyperspace. Such generalization helps the controller agent learn new states based on the similar encountered state, increasing its learning ability.

Matrix is the most common data structure to store state features for DRL-based TSC methods. The lanes of an intersection are discretized into small segments with a fixed size, normalized by the average length of vehicles and the length of lanes. The segments are put together into a matrix as the form of an image-like structure to represent the traffic environment. Fig. 2.1 shows two ways of image-based state representations using the matrix data structure. Representation A creates a snapshot of the intersection and fills cells with value 1 if vehicles are present in the cells [60]. Representation B removes all unused zeros in the representation A and stacks the cells of lanes [61].

Multiple matrices describing different features can be stacked to represent a traffic state. For example, Shabestary et al. [61] and Liang et al. [62] used two matrices to represent VP and VS features, respectively. A cell in VP matrix denotes the presence of a vehicle or the absence of a vehicle. Value 1 denotes that a vehicle is in the cell area. Otherwise, the cell value is 0. The VS matrix collects the velocities of the approaching vehicles at an intersection. The VS matrix cells are the speed values (m/s) of vehicles corresponding to

the cells with value 1 in the VP matrix. The value of the VS matrix can be normalized into range $[0, 1]$ by the speed limits of lanes to balance the weights of lanes due to different speed limits [63–66].

Using a raw pixel image of the intersection and its streets is another way to represent traffic state [67–69]. The image is taken by the cameras installed at an intersection or the snapshot of the intersection if the traffic environment is a graphical simulator (e.g., SUMO [70]) [67, 71–73]. This representation takes advantage of DL techniques. It detects vehicles' positions and possibly predicts the speeds and travel directions of vehicles according to the sequence of input images.

2.1.2 Action Design

Action defines an action space that an agent can take to change or impact on environment [20]. The action in the TSC problems describes the way of controlling the traffic signal timing plans to alleviate the congestion problem. Below summarizes three categories of action designs in RL-based traffic signal control methods.

The first category of the action designs is a TSP-based action space. The action space is defined as $\mathcal{A} = \{p_1, p_2, \cdots, p_n\}$, where p_i denotes a TSP of the traffic signal timing plans. The new selected TSP replaces the current TSP if they are different. Otherwise, the agent keeps the current TSP until a different TSP is selected [60, 67, 74]. In order to avoid TSP flickering issue, the duration of a TSP can be limited by a pre-set duration. For example, Ha-li and Ke [75] set minimum and maximum duration to each TSP, meaning that the agent cannot replace the current TSP until the TSPET is higher than the minimum threshold and the TSP must be replaced by another TSP when the elapsed time is higher than the maximum threshold.

The second category of the action designs is binary-based action space [65, 76]. The action is defined as $\mathcal{A} = \{0, 1\}$, where 0 denotes keeping the current TSP, and 1 means switching to the next TSP according to the defined TSP cycle. The primary drawback of this design is that the extra phase time between the current TSP and the target TSP reduces the TSC's efficiency if the next TSP is not the optimal option for the current state.

The third action design is an action space representing the duration of TSP. One of the

functions of this action design is to define the ratio of TSP duration in a TSP cycle [77]. The action space is defined as $\mathcal{A} = \{r_{p_1}, r_{p_1}, \cdots, r_{p_n}\}$, where r_{p_i} denotes the ratio of TSP p_i duration in a TSP cycle. Jin and Ma [78] and Liang et al. [62] proposed another approach to adjust TSP duration by using an adjustment vector $S^{adj} = \{-t_1, -t_2, \cdots, 0, t_1, t_2, \cdots\}$. The action is the adjustment in seconds to TSP duration. The lower and upper bounds of the TSP duration are set to constrain the duration after the adjustments.

2.1.3 Reward Design

Designing a reward is a critical part of RL-based TSC methods. A reward is a signal to frame an agent's objectives and assess the agent's actions in reaching the objectives [20]. Several objectives of TSC are listed below:

1. *Minimizing waiting time (MWT)*: MWT is an objective to minimize the waiting time of vehicles during their trips.

2. *Minimizing queue length (MQL)*: MQL is an objective to minimize the number of vehicles queueing at intersections.

3. *Minimizing delay (MD)*: MD is an objective to minimize the extra time spent by vehicles on the trips. The extra travel time is the absolute difference between the expected travel time and the actual travel time. The expected travel time is the duration of time for a vehicle to travel from the start place to destination in free flows.

4. *Maximizing throughput (MT)*: MT is an objective to maximize the number of vehicles passing through a traffic network in a period.

5. *Minimizing travel time (MTT)*: MTT is an objective to minimize the time of vehicles spending on their trips. Travel time of a trip includes both the time waiting and moving in a traffic road network.

6. *Minimizing trip stops (MTS)*: MTS is an objective to minimize the number of vehicle stops during the trips. The stop times of a vehicle denotes how often the speed is reduced to 0.1 m/s or less than 0.1 m/s. Fewer stop times means vehicles move more smoothly in the traffic road network.

7. *Minimizing fuel consumption (MFC)*: MFC is an objective to minimize the fuel consumption of vehicles on trips. Higher stop-go movements increase vehicle fuel consumption.

Reward for a Single Objective

A reward model for a single objective evaluates the executed action by a single traffic measurement. Similar to the reward model proposed in [79, 80] for the objective of MWT (see Eq. 2.1), the reward calculates the difference in the waiting times of all vehicles between two executing actions, where w_t denotes the total waiting time or average waiting time of approaching vehicles at time step t, a and b are scaling factors. An agent gets the positive reward for the action a_t if w_{t+1} is reduced.

$$r_t = aw_{t+1} - bw_t \tag{2.1}$$

A reward design shown in Eq. 2.2 is another reward model for the objective of MQL [23, 66, 76, 81]. The q_t in Eq. 2.2 denotes total QL or average QL of the intersection at time step t. The action a_t results in a positive reward if q_{t+1} is less than q_t. Another reward design for MQL is the absolute difference of QL between different directions (see Eq. 2.3) [82, 83]. The q^{e-w}, q^{w-e}, q^{s-n}, q^{n-s} denotes the QL on the movements from east to west, west to east, south to north and north to south, respectively. This reward design tries to balance the QL from north-south and west-east directions.

$$r_t = aq_{t+1} - bq_t \tag{2.2}$$

$$r_t = |max\{q_t^{e-w}, q_t^{w-e}\} - max\{q_t^{s-n}, q_t^{n-s}\}| \tag{2.3}$$

The objective of MD is generally framed by the reward design shown in Eq. 2.4 [63, 67, 84]. The q_t in Eq. 2.4 denotes the delay of approaching vehicles at time step t. It returns a positive reward for the action if the overall delay of vehicles is reduced. Another design with expected delay of vehicles is proposed in [85] (see Eq. 2.5). The design calculates the difference between the current delay d_t and the expected delay D_0 instead of calculating the differential delay between two actions. The reward design returns a better reward if

the action a_t reduces the difference between the vehicles' delay and the expected delay.

$$r_t = ad_{t+1} - bd_t \tag{2.4}$$

$$r_t = K \times (D_0 - D_t) \tag{2.5}$$

MT is another objective to maximize the throughput of a traffic network. Lin et al. [86] designed a reward model as the difference between the number of vehicles entering and leaving the traffic network, meaning that more vehicles remain in the traffic road network if the number of exiting vehicles is lower than the number of entering vehicles. This objective can also be framed by a reward that calculates the number of vehicles going through the intersections after executing the action [69].

Reward for Multiple Objectives

A reward design for multiple objectives has been proposed in recent research in RL-based TSC methods to let agents perform well on different demands. The reward design's generic way for multiple objectives is to combine multiple weighted factors into a single scaled reward value. Eq. 2.6 shows a reward design for multiple objectives with five factors [60]. The c in Eq. 2.6 indicates whether or not the agent changes the TSP to avoid the flickering issue, j denotes penalties for teleports, $\sum_{i=1}^{N} e_i$ denotes the total number of emergency stopping of vehicles, $\sum_{i=1}^{N} d_i$ denotes the total delay of vehicles, and $\sum_{i=1}^{N} W_i$ is the total waiting time of all vehicles. This reward design lets the agent control traffic signals to keep the five different traffic measurements as low as possible, including flickering TSP, teleports, stops, delay, and waiting time at intersections. Wei et al. [87] designed a reward model with six factors (see Eq. 2.7), which considers the objectives of MQL, MD, MWT, MT, and MTT, and avoid the flickering issue as well. The $\sum_{i \in L} q_i$ in Eq. 2.7 denotes the total QL over all lanes for the objective of MQL, $\sum_{i \in L} d_i$ is the total delay of all lanes for the objective of MD, $\sum_{i \in L} W_i$ is the total waiting time of vehicles running on all lanes for the objective of MWT, C denotes changing or keeping TSP to avoid flickering issue, N is the total number of vehicles going through the intersection for the objective of

MT and T denotes the travel time of vehicles for the objective of MTT.

$$r_t = -w_1 c - w_2 \sum_{i=1}^{N} j_i - w_3 \sum_{i=1}^{N} e_i - w_4 \sum_{i=1}^{N} d_i - w_5 \sum_{i=1}^{N} W_i \tag{2.6}$$

$$r_t = w_1 \sum_{i \in L} q_i + w_2 \sum_{i \in L} d_i + w_3 \sum_{i \in L} W_i + w_4 C + w_5 N + w_6 T \tag{2.7}$$

Another reward design for multiple objectives depends on traffic conditions. Houli et al. [59] designed a reward model for three different traffic conditions: free-flow condition with the objective of MTS, medium vehicle demand condition with the objective of MWT, and oversaturated condition with the objective of MQL. In different traffic conditions, the reward value $r_t = 1$ if the traffic measurements of the objectives increase. Otherwise $r_t = 0$.

Several other reward designs with fewer factors were proposed for multiple objectives. For example, Ha-li and Ke [75] designed a reward model with QL and throughput factors for MQL and MT objectives, Casas [77] designed a reward model with VN and VS factors for MT and MD objectives, Xu et al. [88], [89] and Chu et al. [90] designed a reward model with VN and WT factors for MT and MWT objectives, Tan et al. [91] designed a reward model with the factors of QL and VN for the objectives of MQL and MT.

2.2 Reinforcement Learning-based Traffic Signal Control Methods

This section presents the traffic signal control methods based on reinforcement learning techniques. Table 2.1 summarizes the methods and their strengths to solve the TSC problem. Model-based and model-free methods are first presented to show the methods of learning expected value functions. Cooperative TSC methods are summarized in Section 2.2.3 to present methods for cooperatively learning and controlling among multiple intersections. Section 2.2.4 presents the state-of-the-art DRL-based TSC methods that utilize deep learning techniques to improve the agent learning capability.

Table 2.1: Summary of RL-based Methods and Their Strengths for Traffic Signal Control Problem.

Methods	Section	Description	Strength
Model-based Methods	2.2.1	Agents get the transition probability of states and the expected rewards by proposed models .	Improve learning speed.
Model-free Methods	2.2.2	Agents calculates the value of states or state-action pairs by delayed rewards without using proposed models.	Solve TSC problems in dynamic traffic conditions.
Cooperative TSC Methods	2.2.3	Cooperatively control traffic signals among multiple intersections.	Upstream and downstream traffic conditions are considered to achieve global TSC optimization.
Deep RL-based Methods	2.2.4	Utilize deep learning techniques to generalize traffic states and learn TSC policy.	Address the issue of high traffic state dimensions.

2.2.1 Model-based RL Methods for Traffic Signal Control

Model-based RL methods model traffic environments to estimate the transition probability $P(s, a, s')$ of the current state s transiting to the next state s' by the executed action a. The model predicts possible next traffic states $\mathcal{S}'$ along with the state transition probabilities based on the current traffic state s and the action a. The rest of this section summarizes two proposed models for predicting state transition probabilities.

Vehicle-based Model for States Transition Probabilities

The vehicle-based model estimates the state transition probabilities by counting the number of states that occur based on vehicle positions. As an example in [58], the state of a vehicle is represented as $s[TSP, VP, V_{des}]$, where TSP denotes the current traffic signal settings including two possible entries: red or green for the vehicle, VP denotes the vehicle position in the queue, and V_{des} denotes the destination of the vehicle. Hence, the probability of the state transition probability from the current state s to the next state s' is represented as $P(s[TSP, VP, V_{des}], TSP, s'[TSP', VP', V_{des}])$.

Transition probabilities are calculated by following counters that are stored in a lookup table:

- $C(VP, V_{des})$: the number of times that a vehicle arrives at the position VP with the destination V_{des}.
- $C(TSP, VP, V_{des})$: the number of times that a vehicle arrives at the position VP with the destination V_{des} and traffic signal settings TSP.
- $C(TSP, VP, V_{des}, VP')$: the number of times that a vehicle arrives at the position VP and will move to the position VP' with the destination V_{des} and traffic signal settings TSP.

Then, the state transition probability $P(s[TSP, VP, V_{des}], TSP, s[TSP', VP', V_{des}])$ is calculated in the equation below:

$$P\Big(s[TSP, VP, V_{des}], TSP, s'[TSP', VP', V_{des}]\Big) = \frac{C\Big(TSP, VP, V_{des}, VP'\Big)}{C\Big(TSP, VP, V_{des}\Big)}. \tag{2.8}$$

The state transition probabilities are then used to updated Q-values of the state-action pairs by Eq. 2.9. The reward r in Eq. 2.9 is designed by a binary value for the objective of MWT. It returns 1 when the vehicle stays at the same place and 0 when the vehicle moves forward.

$$\begin{aligned} Q\Big(s[TSP, VP, V_{des}], TSP\Big) = \sum_{s'} P\Big(s[TSP, VP, V_{des}], TSP, s'[TSP', VP', V_{des}]\Big)\Big(r+ \\ \gamma \sum_{TSP} P(TSP|s[TSP, VP, V_{des}]) Q(s[TSP, VP, V_{des}], TSP)\Big) \end{aligned} \tag{2.9}$$

with

$$P\Big(TSP|s[TSP, VP, V_{des}]\Big) = \frac{C\Big(TSP, VP, V_{des}\Big)}{C\Big(VP, V_{des}\Big)}, \tag{2.10}$$

where $P(TSP|s[TSP, VP, V_{des}])$ denotes the probability of a vehicle at the position VP when the traffic signal phase is TSP.

The vehicle-based model proposed by Wiering [58] has been shown to reduce vehicle waiting time and reduce the state dimensions. The model has been applied in [59], who extended the MWT objective to multiple objectives, including MTS, MWT, and MQL, by designing three different reward designs corresponding to three levels of traffic demands: free-flow traffic demand, medium, and congested traffic demand. Khamis et al. [92] also applied the vehicle-based model and improved the way of calculating the transition probability by utilizing the Bayes rule.

Agent-based Model for States Transition Probabilities

The agent-based model estimates the state transition probabilities by counting the experience that the agent has encountered. As an example in [56], the model of transition probability is collaboratively updated by the local signalized intersection (agent i) and its neighbor intersections NB_i. The experience of the agent i is represented by joint state-action pairs $([s_i, s_{NB_i}], [a_i, a_{NB_i}])$ including the local states s_i, local actions a_i, neighbor states s_{NB_i} and neighbor actions a_{NB_i}.

Two kinds of lookup tables were applied by El-Tantawy and Abdulhai [56] to estimate the state transition probabilities. The first lookup table $M_{i,NB_i[j]}$ estimates the states transition probabilities of the neighbor $NB_i[j]$, where $NB_i[j] \in NB_i$. It counts the number of times that the agent i and neighbor $NB_i[j]$ have visited. The row of the table $M_{i,NB_i[j]}$ is represented by the joint state $[s_i, s_{NB_i[j]}]$ and the columns are the actions $a_{NB_i}[j]$ of the neighbor $NB_i[j]$. The estimated states transition probabilities $M_{i,NB_i[j]}([s_i, s_{NB_i[j]}], a_{NB_i}[j])$ of neighbor $NB_i[j]$ is calculated by the following equation:

$$M_{i,NB_i[j]}\Big([s_i, s_{NB_i[j]}], a_{NB_i}[j]\Big) = \frac{C\Big([s_i, s_{NB_i[j]}], a_{NB_i}[j]\Big)}{C\Big([s_i, s_{NB_i[j]}]\Big)}, \tag{2.11}$$

where $C(*)$ denotes the counts of the experience $*$.

Another lookup table $Q_{i,NB_i[j]}$ estimates the Q-values of the experience based on the first lookup table $M_{i,NB_i[j]}$. The rows of $Q_{i,NB_i[j]}$ are represented by joint states $[s_i, s_{NB_i}[j]]$ and the columns are joint actions $[a_i, a_{NB_i}[j]]$. Based on the estimated state transition probabilities $M_{i,NB_i[j]}([s_i, s_{NB_i[j]}], a_{NB_i}[j])$, the expected Q value $\tilde{Q}_{i,NB_i[j]}$ of the next state

s_i' is calculated by Eq. 2.12. It is then used to update the Q-value of the experience $([s_i, s_{NB_i}[j]], [a_i, a_{NB_i}[j]])$ by using the Q-learning algorithm shown in Eq. 2.13.

$$\tilde{Q}_{i,NB_i[j]} = \max_{a_i \in \mathcal{A}_i} \Bigg[\sum_{a_{NB_i}[j] \in \mathcal{A}_{NB_i[j]}} Q_{i,NB_i[j]}\Big([s_i, s_{NB_i}[j]], [a_i, a_{NB_i}[j]]\Big) \times M_{i,NB_i[j]}\Big([s_i, s_{NB_i[j]}], a_{NB_i}[j]\Big) \Bigg], \tag{2.12}$$

where $\mathcal{A}_i$ denotes the action space of the agent i and $\mathcal{A}_{NB_i[j]}$ denotes the action space of the neighbor $NB_i[j]$.

$$Q'_{i,NB_i[j]}\Big([s_i, s_{NB_i}[j]], [a_i, a_{NB_i}[j]]\Big) = (1-\alpha) Q_{i,NB_i[j]}\Big([s_i, s_{NB_i}[j]], [a_i, a_{NB_i}[j]]\Big) + \alpha\Big[r_i + \gamma \tilde{Q}_{i,NB_i[j]}\Big], \tag{2.13}$$

where r_i denotes the reward that is returned to the agent i for the state-action pair $\Big([s_i, s_{NB_i}[j]], [a_i, a_{NB_i}[j]]\Big)$.

The agent-based model proposed by El-Tantawy and Abdulhai [56] has shown the improvements on multiple objectives, such as MWT, MQL, and MD. It controls traffic signals based on the estimated models of the adjacent intersections. However, the joint state-action pairs of local and adjacent intersections increase the lookup table's dimensions, leading to high memory and computing capability requirements.

2.2.2 Model-free RL Methods for Traffic Signal Control

Model-free RL methods learn value functions by way of trial-and-error learning progress [20]. They do not estimate how the states will change in response to the executed actions, which is opposite to the model-based methods that focus on learning the environment's models. In the traffic signal control problems, most RL-based TSC methods rely on model-free RL methods because it is hard to predict traffic states due to the large scale of the traffic environment and dynamic traffic patterns. The rest of this section summarizes the literature model-free RL-based TSC methods.

One-step Tabular-based RL Methods for Traffic Signal Control

One-step tabular-based RL methods store the Q-values of state-action pairs in a lookup table and update the Q-value of the current state-action pair immediately when the state transits to the next state [20]. The lookup table's rows are encountered traffic states, and columns are actions that the agent can execute. Every cell in the table is the Q-value of the corresponding state-action pair. Q-learning [54, 93, 94] and SARSA [95–97] are two algorithms commonly applied to update the Q-values on every time step. We will further introduce the Q-learning and SARSA algorithms in the next chapter.

One-step tabular-based RL methods have been shown to improve the efficiency of TSC in small traffic road networks. However, these methods are not capable of extensive traffic road network since the state-action space is too large to store in tables.

Multi-step Tabular-based RL Methods for Traffic Signal Control

Compared to one-step methods, multi-step tabular-based RL methods memorize multi-step backups of experience to update the Q-values instead of updating the Q values immediately when the state transits to the next state. These methods are also called Temporal-Difference(λ) algorithms that utilize eligibility traces to achieve a smooth updating of Q values, where λ denotes the number of backup steps. As an example in [98], SARSA(λ) algorithm was applied to update Q values. Assuming that an episode of a learning progress has T steps, then the Q value $Q(s_t, a_t)$ of the state-action pair (s_t, a_t) at time step t is updated by the following equation,

$$\begin{aligned} Q(s_t, a_t) = Q(s_t, a_t) + \alpha\delta_t(s_t, a_t)e_t(s_t, a_t) + \alpha\delta_{t+1}(s_{t+1}, a_{t+1})e_{t+1}(s_{t+1}, a_{t+1}) \\ + \cdots + \alpha\delta_T(s_T, a_T)e_T(s_T, a_T) \end{aligned} \tag{2.14}$$

with

$$\delta_t(s_t, a_t) = r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t), \tag{2.15}$$

where e_t denotes the degree of the trace eligibility with two options defined in Eq. 2.16 and Eq. 2.17. There are two ways to calculate the cumulative discounted trace eligibility. The

first option adds more credits to the trace if the agent visits the same state-action pairs. The second option replaces the credits of trace eligibility by 1 if the agent visits the same state-action pairs, otherwise 0 if the actions are different. The eligibility of the new state that the agent has not visited is calculated as a discounted trace eligibility by a discounted factor μ.

$$e_t(s,a) = \begin{cases} \gamma\mu e_{t-1}(s,a) + 1, & \text{if } s = s_t \text{ and } a = a_t \\ \gamma\mu e_{t-1}(s,a), & \text{otherwise} \end{cases} \tag{2.16}$$

$$e_t(s,a) = \begin{cases} 1, & \text{if } s = s_t \text{ and } a = a_t \\ 0, & \text{if } s = s_t \text{ and } a \neq a_t \\ \gamma\mu e_{t-1}(s,a), & \text{if } s \neq s_t \end{cases} \tag{2.17}$$

The multi-step tabular-based RL method in [98] has been shown to improve traffic mobility efficiency in dynamic traffic environments. The multi-step eligibility smoothly traces the updates of Q-values and enhances the representation of reward by multi-step backups because a one-step reward shows little difference between two continuous actions in a short period.

RL Methods with Function Approximation for Traffic Signal Control

Tabular-based RL methods can only address the TCS problems in the traffic environment with small state and action space. However, in the real-world traffic scenarios, the dimensions of traffic state and action space are too large to be represented as arrays or tables. Therefore, the RL methods with function approximation are more applicable for addressing the curse-dimensional state and action space issue. These methods use an approximated function to generalize the traffic states to Q-values of actions so that the agent does not need to store all encountered state-action pairs. One example is the approximated function shown in Eq. 2.18 proposed by LA and Bhatnagar [55]. The function calculates the Q values of the input state-action pairs (s, a) based on a set of parameters θ and a vector of features σ. In this Q-value representation, the controller agent only needs to adjust the parameters θ instead of storing Q values of all possible state-action pairs in a lookup table.

$$Q(s,a) = \theta^T \sigma_{s,a} \tag{2.18}$$

In [55], the features σ_t at time step t of the traffic environment is represented as a vector, as shown in Eq. 2.19, which consists of the queue lengths of all lanes, the elapsed time of TSPs and the actions of all intersections. The parameters θ are updated at every time step when the state transits to the next state by the Eq. 2.20.

$$\sigma_{s_t,a_t} = \Big(\sigma_{q_{1,t}}, \sigma_{q_{2,t}}, \cdots, \sigma_{q_{L,t}}, \sigma_{et_{1,t}}, \sigma_{et_{2,t}}, \cdots, \sigma_{et_{L,t}}, \sigma_{a_1,t}, \sigma_{a_2,t}, \cdots, \sigma_{a_m,t}\Big), \tag{2.19}$$

where $\sigma_{q_{i,t}}, i \in L$ denotes the queue length of the lane i, L denotes the lanes of all intersections, $\sigma_{et_{i,t}}, i \in L$ denotes the elapsed time of the TSP on the lane i, $\sigma_{a_j,t}, j \in m$ denotes the action at the intersection j, m denotes the number of intersections in the traffic environment.

$$\theta_{t+1} = \theta_t + \alpha\sigma_{s_t,a_t} \cdot \Big(r_t + \gamma \min_{a\in\mathcal{A}} \theta_t^T \sigma_{s_{t+1},a} - \theta_t^T \sigma_{s_t,a_t}\Big), \tag{2.20}$$

where α denotes the learning step size, γ denotes the discount factor, r_t denotes the reward value at time step t, which is calculated by Eq. 2.21 to represent the cost value of queue lengths and the elapsed time of TSPs.

$$r_t = \eta_1 \cdot \Big(\sum_{i\in I_p} \eta_2 \cdot q_{i,t} + \sum_{i\notin I_p} (1-\eta_2) \cdot q_{i,t}\Big) + (1-\eta_1) \cdot \Big(\sum_{i\in I_p} \eta_2 \cdot et_{i,t} + \sum_{i\notin I_p} (1-\eta_2) \cdot et_{i,t}\Big) \tag{2.21}$$

where I_p denotes a set of lanes on primary roads, $\eta_1, \eta_2 \in [0, 1]$ are weights of the lanes on primary roads and minor roads, respectively.

RL method with function approximation in [55] has been shown to improve the speed of learning progress in a traffic road network with multiple intersections, as well as reduce the delay of vehicles.

2.2.3 Cooperative Traffic Signal Control Methods

Cooperative TSC methods enable all controller agents to control traffic signals cooperatively to achieve global optimization. The TSC approaches for an isolated intersection only collect local traffic states and control traffic signals for local objectives. These methods do not consider the traffic conditions of system-wide/global intersections. However, the traffic flows of an isolated intersection depend on the traffic conditions of its adjacent intersec-

tions and itself [99–102]. Therefore, controlling traffic signals cooperatively for multiple intersections is critical to the TSC application in a massive traffic road network. The rest of this section presents several approaches for addressing traffic signal control problems cooperatively among multiple intersections.

Cooperative Traffic Signal Control based on Max-plus Algorithm

The max-plus algorithm enables an agent to select an optimal action based on payoff values among connected agents in a coordination graph [103–105]. A payoff value of an agent is repeatedly sent to its neighbors until the payoff is convergent at a point in finite iterations. The purpose of the Max-plus algorithm is to find the optimal joint actions so as to maximize global payoff.

As an example in [106], a traffic road network is represented as a coordination graph. Every intersection is an agent in the graph, and the link between two intersections is the edge of the graph. Let us assume two connected agents i and j represent two adjacent intersections in a traffic road network. The payoff value from the local agent i to the neighbor agent j is defined by the following equation:

$$\mu_{ij}(a_j) = \max_{a_i} \left\{ Q_{ij}(s, a_i, a_j) + \sum_{k \in \Gamma(i)/j} \mu_{ki}(a_i) \right\} + c_{ij}, \tag{2.22}$$

where $\Gamma(i)/j$ denotes the neighbor agents of i except the agent j, c_{ij} is a normalization vector, s_i denotes the state of agent i, a_i and a_j are the actions of the agent i and j, respectively. $Q_{ij}(s, a_i, a_j)$ is the local value function that maps the state s, local action a_i and neighbor action a_j to a real number (see Eq. 2.23), indicating the payoff value of the joint actions. The optimal action a^* is selected by Eq. 2.24 when the sum of the payoff value received by the local agent i is convergent or the iteration is terminated.

$$Q_{ij}(s, a_i, a_j) = \sum_{p_{l_i}} O_{p_{l_i}} \cdot Q_{p_{l_i}}(s_{p_{l_i}}, a_i, a_j) + \sum_{p_{l_j}} O_{p_{l_j}} \cdot Q_{p_{l_j}}(s_{p_{l_j}}, a_j, a_i), \tag{2.23}$$

where l_i denotes the connecting lanes of the agent i, p_{l_i} denotes the positions on lanes l_i,

$O_{p_{l_i}}$ is a binary value indicating the present vehicle at the position p_{l_i}.

$$a_i^* = \arg\max_{a_i} \sum_{j \in \Gamma(i)} \mu_{ji}(a_i), \tag{2.24}$$

The max-plus algorithm has also achieved success in cooperative traffic signal control in several other studies [60, 107]. Medina and Benekohal [107] used the payoffs of the max-plus algorithm to represent the expected rewards of all possible actions. Van der Pol and Oliehoek [60] applied the max-plus algorithm to optimize global joint actions based on the transfer planning technique [108].

Cooperative Traffic Signal Control based on Joint States

Joint states enable an agent to parallel learn both the local traffic environment and the environment of the adjacent or even global intersections. It considers the traffic conditions of the upstream and downstream intersections to avoid the congestion problem in a system-wide traffic road network.

As an example in [23], the technique of graph attention networks [109] was applied to learn the influence of neighbor intersections. It enables the local agent i to consider the state of neighbor intersections by evaluating their influences. The intersection with more significant influence increases its attention to the local agent i to impact the action selection of the agent i. Three steps were proposed to calculate the influence of neighbor intersections:

1) Observation Interaction: It calculates the importance of the neighbor intersection j by joining the state s_j of the neighbor j with the local state s_i by the following equation:

$$e_{ij} = (h_i W_i) \cdot (h_j W_j)^T \tag{2.25}$$

with

$$h = \sigma(s \cdot W_e + b_e), \tag{2.26}$$

where h denotes the embedded state of the state s, W_e and b_e are weight matrix and bias vector, respectively, W_i and W_j are the scaled parameters of the embedded states h_i and h_j, respectively, σ is ReLU function to transform the embedded state to a real value.

2) Attention Distribution: This step is to generalize the influence of the neighbor intersections by the softmax algorithm shown below:

$$\alpha_{ij} = \text{softmax}(e_{ij}) = \frac{\exp(e_{ij}/\tau)}{\sum_{j \in NB_i} \exp(e_{ij}/\tau)}, \tag{2.27}$$

where NB_i denotes the neighbor intersections of the agent i.

3) Neighborhood Cooperation: The last step is to calculate the overall influence hs_i of several neighbor intersections on the local agent i by combining their weighted influences. The overall influence is calculated by the following definition:

$$hs_i = \sigma\left(W_q \cdot \sum_{j \in NB_i} \alpha_{ij}(h_j \cdot W_c) + b_q\right) \tag{2.28}$$

where W_c denotes the weight parameters for the neighbor intersections, W_q and b_q are scaled factors.

Joint states for cooperative TSC in [23] have shown to improve the efficiency of TSC in large-scale traffic road networks. The algorithm of calculating the influence of neighbor intersections significantly reduces the dimension of joint states compared to the earlier studies in [21, 110], who applied the joint states by concatenating local and neighbor states directly.

Cooperative Traffic Signal Control based on Joint Q-Values

Joint Q-values enable the agents to share Q-values among adjacent intersections and cooperatively learn the global optimal value functions. The optimal action that is selected by the agent not only depends on the local Q-values, but also considers the Q-values of adjacent intersections.

For example, in [22], the Q-values of agents were exchanged among adjacent agents to learn both local and adjacent Q-values. In this case, the local agent i selected the optimal action based on both the reward of the local state-action pair and the rewards of the adjacent state-action pairs. This assumed that at time step t, the Q-value of a state-action pair $\langle s_{i,t}, a_{i,t} \rangle$ of the agent i is defined as $Q_{i,t}(s_{i,t}, a_{i,t})$. The collection of the neighbor agents of the local agent i is defined as NB_i. Hence, the joint Q-value of a state-action pair

$\langle s_{i,t}, a_{i,t} \rangle$ at time step t is updated by following equation:

$$Q_{i,t}(s_{i,t}, a_{i,t}) = Q_{i,t}(s_{i,t}, a_{i,t}) + \alpha \Big[r_{i,t} + \gamma \max_{a} Q_{i,t}(s_{t+1}, a) - Q_{i,t}(s_{i,t}, a_{i,t}) \Big] + \sum_{j \in NB_i} \omega_{i,j} Q_{j,t+1}(s_{j,t+1}, a_{j,t+1}) \Big), \tag{2.29}$$

where $\omega_{i,j}$ denotes the weights of the Q-values from the neighbor agent j, which is represented as $\omega_{i,j} = \frac{1}{|NB_i|}$.

Another similar example of joint Q-values learning was proposed in [66]. The value function they proposed is represented by two deep Q-networks with parameters θ and θ', respectively. The exchanged Q-values of the neighbor agents are not the Q values of the next time step $t+1$. Instead, they used the Q-values of the previous time step $t-1$. The updating function of the joint Q-values is defined as follows:

$$Q_{i,t}(s_{i,t}, a_{i,t}) = (1-\alpha) Q_{i,t}(s_{i,t}, a_{i,t}; \theta_i) + \alpha \Big[r_{i,t} + \gamma \max_{a} Q_{i,t}(s_{t+1}, a; \theta_i') \Big] + \sum_{j \in NB_i} \omega_{i,j} Q_{j,t-1}(s_{j,t-1}, a_{j,t-1}; \theta_j) \Big), \tag{2.30}$$

where θ' denotes the parameters of the target value function with a lower updating frequency in order to avoid overestimated Q values.

Centralized Cooperative Traffic Signal Control

Centralized cooperative TSC methods learn the global optimal TSC policy for multiple intersections. These methods have a single agent to learn a global value function of the entire intersections and cooperatively output joint actions for every intersection. As an example of the centralized cooperative TSC method proposed by [55], the state of the entire intersections is represented as a vector, including the queue lengths of all lanes and the elapsed time of the TSP at every intersection (see Section 2.2.2). However, the size of the state-action space is enormous when the traffic road network scale is massive.

Tan et al. [91] proposed an approach called decentralized-to-centralized coder to solve the issue of curse dimensionality. The approach consists of two layers of learning models to find the global optimal Q function Q^g. The first layer of the approach is a decentralized learning model. It segments the large-scale traffic road network into several small regions,

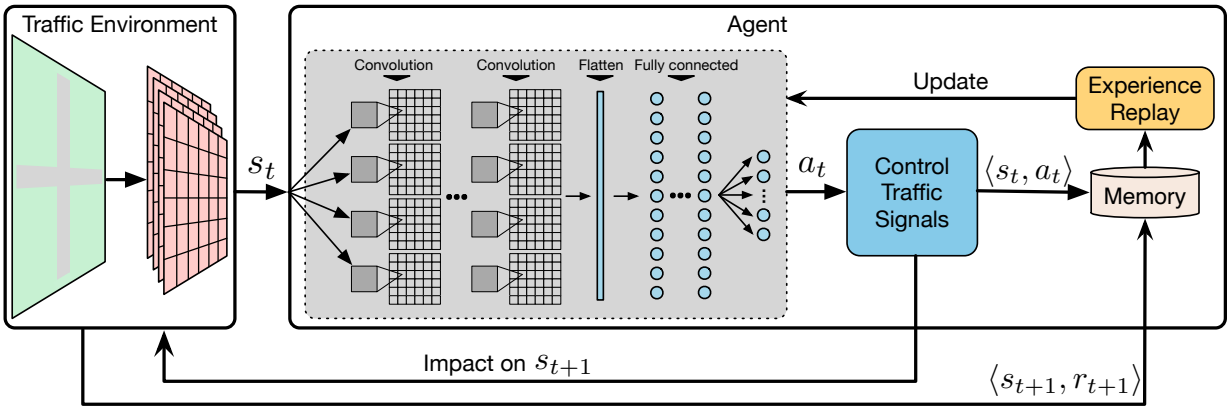


Figure 2.2: The architecture of deep reinforcement learning in traffic signal control problem.

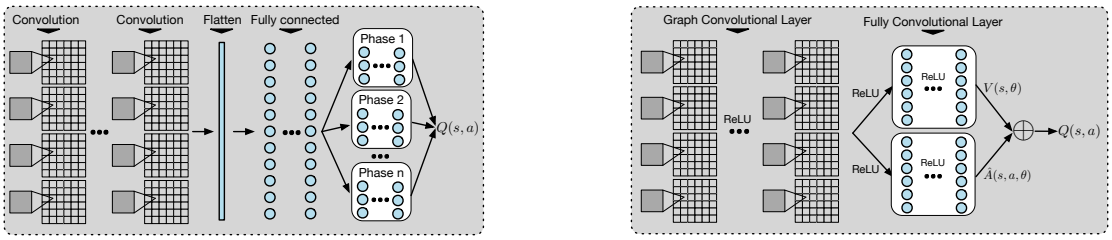
containing a small number of intersections. The agent of a region learns the optimal policy based on the regional traffic states and reward values. The second layer aggregates regions' optimal actions to learn a global Q function to generate the optimal global actions for regions.

Centralized cooperative TSC methods in [55, 91] have been shown to improve the performance of cooperative TSC in the traffic environment with multiple intersections. However, these methods increase the learning cost compared to the distributed methods for isolated intersections.

2.2.4 Deep Reinforcement Learning-based Traffic Signal Control Methods

The Deep Reinforcement Learning (DRL) technique is an approximation function-based RL method. It applies deep learning techniques as an approximation function to address high-dimensional problems. In traffic signal control problems, a large scale of a traffic road network and detected features result in the curse dimensionality issue in constructing traffic states, meaning that tabular-based representation, such as arrays or tables, is not enough to store such high-dimensional states. Therefore, DRL techniques have been attracted to the interests of the research in the TSC problems to address high dimensionality.

Figure 2.2 shows the architecture of DRL based on DQN for addressing traffic signal control problems. Convolutional neural networks (CNN) is applied as the DQN in the architecture, allowing the agent to learn raw and high-dimensional traffic states. Let us



(a) The architecture of Phase-Gate DQN [87].

(b) The architecture of Dueling DQN with graph convolutional neural networks [111].

Figure 2.3: The architecture of deep Q-network in value-based DRL methods.

assume that the traffic environment features are detected by detectors and transformed into image-based state representation as to the input of the DQN at time step t. The agent controls traffic signals based on the output of the DQN and stores the experience $\langle s_t, a_t, s_{t+1}, r_{t+1} \rangle$ in memory for experience replay. The rest of this section presents the DRL-based methods for the TSC problem based on the architecture.

Value-based Deep Reinforcement Learning for Traffic Signal Control

Value-based DRL methods use natural deep Q-networks (DQN) as an approximation function to generalize traffic states to the Q values of actions [60]. In the experience replay, the parameters of the DQN are adjusted by the temporal difference (TD) of action Q-values based on rewards to minimize the TD. The agent performs experience replay in learning progress until the action Q-values are convergent.

As an example in [87], a state representation of an intersection consists of a matrix denoting vehicle positions, QLs of connecting lanes, number of on-road vehicles, WT of the vehicles, current TSP and the next TSP. These features are concatenated and flattened into a vector as the input of the DQN. A "Phase-Gate" DQN was proposed to learn action Q-values separately (see Figure 2.3a) based on different TSPs in order to solve the bias of action Q-values because the TSP feature in the state representation is represented as a single value, resulting in lower importance compared with the features represented by matrices or vectors. The experience in memory is also separately stored and sampled by TSPs for the experience replay progress.

Another example in [111] applied dueling DQN [112] instead of the natural DQN to improve the learning efficiency and stability (see Figure 2.3b). Dueling DQN is a neural

network containing two separate DQN streams. One stream estimates state values $V(s;\theta)$ and the other one estimates state-dependent action advantages $\hat{A}(s,a;\theta)$. The output of the dueling DQN is the Q-values of state-action pairs $Q(s,a;\theta)$ combining the state values $V(s;\theta)$ and the value of state-dependent action advantages $\hat{A}(s,a;\theta)$ (see Eq. 2.31). Double dueling DQN (3DQN) improves the dueling DQN by addressing overestimated Q-values [62, 79, 89]. It consists of two dueling DQNs based on the idea of the natural DQN. One of the dueling DQNs is a target network that is trained with a lower frequency than the other one.

$$Q(s,a;\theta) = V(s;\theta) + \Big(\hat{A}(s,a;\theta) - \frac{1}{|A|}\sum_{a\in\mathcal{A}} \hat{A}(s,a;\theta)\Big) \tag{2.31}$$

Policy-based Deep Reinforcement Learning for Traffic Signal Control

Policy-based DRL methods learn policy parameters of the DQN instead of learning the Q-values of actions. The actor-critic DRL method is a policy-based DRL method that has been widely applied in traffic signal control problems. It consists of two models: "actor" and "critic". "Actor" is a policy modelled by a DQN, which generates the action probability distribution for the controller agent instead of action Q-values. "Critic" is a value-based model that can be represented as either a DQN or a linear function for learning action Q-values. At every learning step, "critic" calculates the temporal difference of the action Q-values based on the received reward, and then uses the difference to adjust the parameters of the policy "actor". The agent chooses an action based on the output of the policy "actor" instead of comparing action Q-values.

As an example in [91], Wolpertinger architecture [113] was applied with deep deterministic policy gradient (DDPG) [114] to learn a policy for traffic signal control in a traffic road network. The goal of the Wolpertinger algorithm is to map the continuous action space, which is generated by the policy-based RL method (DDPG), to a discrete action space by the k-nearest neighbor (KNN) algorithm. The architecture of the Wolpertinger algorithm applied by Tan et al. [91] is shown in Figure 2.4. The policy "actor" is represented as a neural network that takes traffic states s as input and outputs a continuous proto-action space $\hat{\mathcal{A}}$. Based on KNN algorithm, the continuous proto-action space $\hat{\mathcal{A}}$ is then mapped into discrete actions $\mathcal{A}$ by $g_k(\hat{\mathcal{A}}) = \arg\min^k_{a\in\mathcal{A}} \parallel a - \hat{\mathcal{A}} \parallel_2$, which returns k-nearest L_2 norm

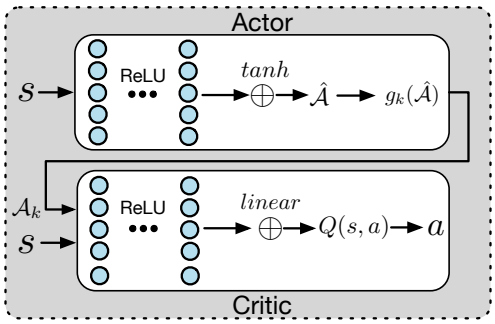


Figure 2.4: The architecture of wolpertinger algorithm [91].

values of discrete actions $\mathcal{A}_k \in \mathcal{A}$. The Q-value function "critic" is represented as a DQN that takes the traffic states s and $\mathcal{A}_k$ as input and outputs the Q-values of state-action pairs. The agent selects the action a with the highest state-action Q-value as the optimal action from $\mathcal{A}_k$. In the experience replay progress, the parameters of "critic" DQN are updated by the optimal action a and the parameters of "actor" DQN are updated by the continuous proto-action space $\hat{\mathcal{A}}$.

DRL-based methods have achieved significant success in addressing the TSC problems in large-scale traffic road networks. It solves the issue of curse-dimensional state space in a dynamic traffic environment and is able to map the traffic state to both discrete and continuous action space for different settings of traffic signal timing plans.

2.3 Summary

This chapter outlines the literature studies on the domain of traffic signal control by using reinforcement learning algorithms. Traffic states, actions and reward designs are summarized at the beginning of this chapter to present the overview of formulating traffic signal control problems by reinforcement learning algorithms. Based on the summary shown in Table 2.1, RL-based traffic signal control methods are classified into model-based and model-free methods. The methods of the model-based classification try to model the traffic environment. The goal of the studies is to find the state transition probability of the traffic environment and the expected rewards for actions. While the model-free methods adjust the state-action Q values step by step based on delayed rewards instead of the models. The approaches to cooperative traffic signal control in a traffic road network are

also summarized in this chapter. Compared with the traffic signal control on an isolated intersection, cooperative traffic signal control methods improve the efficiency of the traffic flow regulation at an intersection by considering the traffic conditions of its adjacent intersections. Deep reinforcement learning-based traffic signal control methods are summarized at the end of this chapter. Deep reinforcement learning algorithms take the advances of deep learning to generalize traffic states so that the signal controllers have capabilities to learn high-dimensional traffic states. Since the dimensions of traffic states in the real traffic environment are too large to store in the tabular-based data structure, deep reinforcement learning algorithms have become the focus of the studies in the domain of traffic signal control.

Chapter 3

Preliminaries

This chapter provides basic concepts of traffic signals in the first section before diving into traffic signal control problems, including the primary attributes of traffic signals and categories of traffic signal control strategies. The second section of this chapter presents the fundamentals of reinforcement learning (RL) techniques for better understanding the RL-based traffic signal control algorithms.

3.1 Traffic Signal Control System

A traffic signal control system (TSCS) adjusts traffic signal timing plans of intersections to regulate traffic movements based on real-time traffic states, and intersection capacity, vehicle demands, vehicular movements, intersection conditions [115]. It broadly cooperates with different areas, such as traffic detection [116–120], data collection [121–125], and V2X communication technology [126–129], to improve the efficiency of traffic signal control. TSCS utilizes algorithms that can determine the optimal traffic signal timing plans in real-time, such as signals' split, phase duration, and sequences to alleviate congestion at intersections, achieving the improvement of TSCS efficiency. This section presents the primary attributes and concepts of TSCS to better understand the algorithms introduced in the subsequent sections.

3.1.1 Traffic Signal Concepts

Traffic signals have three traditional colours: red ("stop"), yellow ("warning") and green ("go"). They serve users at intersections by regulating users' movements to avoid conflict movements.

The users at an intersection include pedestrians, bicycles, and different categories of vehicles that want to go through the intersection [13]. According to the definitions of Federal Highway Administration (FHWA)[1], vehicles are generally categorized into eight classes based on gross vehicle weight rating (GVWR) and the maximum weight of the vehicle specified by manufacturers. Classes 1-2 are defined as light-duty vehicles whose GVWR is lower than 10,000 lbs, classes 3-6 are defined as medium-duty vehicles with GVWR between 10,001 and 26,000 lbs, and classes 7-8 are defined as heavy-duty vehicles with GVWR more than 26,001 lbs.

User movement describes the actions of users regulated by traffic signals at intersections [13]. The actions usually denote the directions of user movements. At an intersection with four two-way approaches, vehicles have three movements, including left-turn movement, go-through movement, and right-turn movement. Pedestrian and bicycle movements are both associated with the go-through movement of the vehicles, since the traffic signals do not regulate the left-turn and right-turn movements of the pedestrians and cyclists.

3.1.2 Traffic Signal Timing Plan

Traffic signal timing plan is defined by various parameters for regulating user movements at intersections [13]. It designs traffic signal phase (TSP), duration, and sequences to manage user movements to avoid conflict movements at intersections. The traffic signal timing plan for vehicular movements is mainly introduced in this thesis work since pedestrian and bicycle movements are both associated with vehicles' go-through movement.

Vehicular movements include 12 one-way movements at an intersection with four approaches according to the definitions in Highway Capacity Manual (HCM) [130]. Figure 3.1 shows the vehicular movements signed by numbers based on HCM definitions. Note that the right-turn movements are signed with the number that is ten more than the number of

[1]https://afdc.energy.gov/data/10380

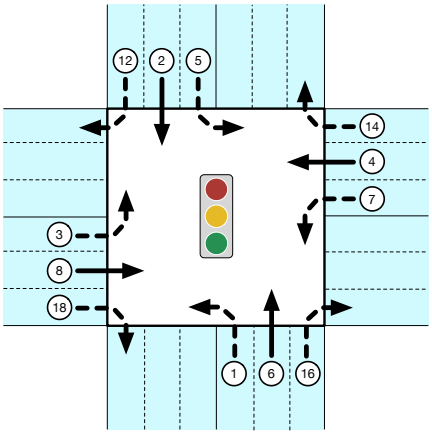


Figure 3.1: Vehicular movements at an intersection with four approaches.

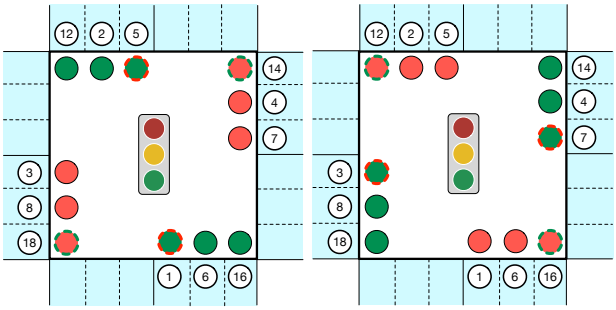


Figure 3.2: The design of left-turn permitted traffic signal phases.

adjacent go-through movements [13]. We can see that the movements with even numbers can be associated with go-through movements, and the movements with odd numbers are left-turn movements. The movements with dashed arrows denote that they must yield to go-through movements in some scenarios. Every movement can be assigned with one or more street lanes; for example, the right-turn movements are usually assigned on the right-most lanes, and the go-through movements can be assigned on both middle and right-turn lanes.

A traffic signal timing plan defines TSPs for non-conflict multiple movements at the same time [13]. A TSP determines which movements can be assigned with green signals simultaneously and the duration of the TSP for vehicles going through the intersections. It mainly includes two types of TSP designs: permitted left-turn design and protected left-turn design [13]. Permitted left-turn TSP design assigns a TSP with green signals for both go-through and left-turn movements, as shown in Figure 3.2. In this scenario, left-turn movements share the green signals with go-through movements, but they must yield

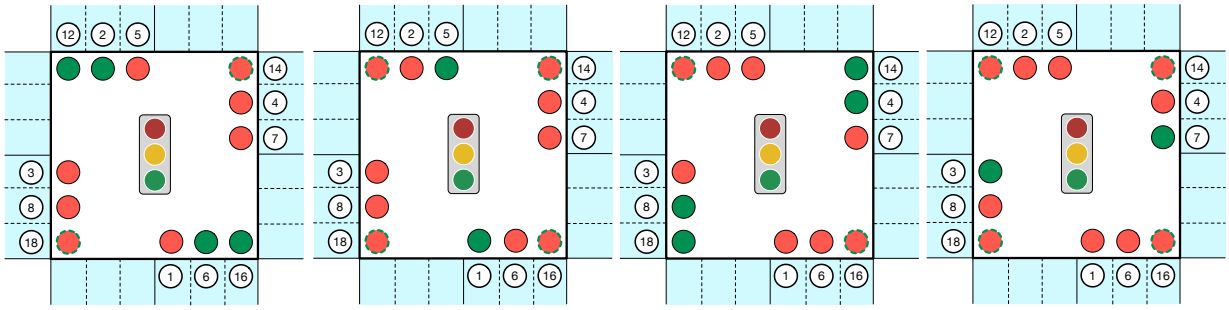


Figure 3.3: The design of left-turn protected traffic signal phases.

to the go-through movements. Meanwhile, protected left-turn TSP design, as shown in Figure 3.3, assigns an individual TSP for left-turn movements and red signals for the rest movements. In both permitted and protected left-turn designs, right-turn movements are signed with a dashed red circle since they share the same TSP with go-through movements and must yield to the go-through movements.

3.1.3 Traffic Signal Control

Traffic signal control (TSC) is a critical part of intelligent transportation management [131–135]. The goal of TSC is to control traffic signal timing plans at intersections to improve traffic management efficiency. Several objectives and strategies have been proposed to achieve the goal of TSC. The rest of this section presents the objectives and corresponding performance measures of TSC algorithms and the categories of TSC strategies.

Objectives and Performance Measures

The objectives of TSC determine the solutions of the TSC algorithms. Objectives affect algorithms to generate different solutions to achieve corresponding objectives. There are four main perspectives for the objectives: pedestrians, bicycles, transit, and vehicles [13]. Different perspectives have their own criteria on traffic signal timing plans.

The performance measures evaluate the efficiency and effectiveness of TSC algorithms. In this research scope, Table 3.1 summarizes the objectives and corresponding measures from the vehicles' perspective.

The vehicle mobility objective is to improve the efficiency of vehicle movements [13].

Table 3.1: Objectives and performance measures of traffic signal control [13, 136].

Objectives	**Measures**
Vehicle Mobility	• Percent arrival on green • Intersection throughput • Waiting queue length • Stops
Vehicle Costs	• Waiting time • Delay • Travel time
Environmentally Friendly	• Fuel consumption & CO_x emissions

Percent arrival on green (PAG) calculates the proportion of vehicles that arrive at intersections when the signals are green [136]. Higher PAG denotes that more vehicles receive green signals in their trips, resulting in better performance on the vehicle mobility objective. Intersection throughput (IT) calculates the volume of vehicles that can go through the intersections in a specific period or the time cost for a specific traffic volume moving through the intersections. Improving IT can significantly improve the efficiency of TSC and alleviate congestion. Waiting queue length (WQL) calculates the number of vehicles queueing at the intersections in a specific period. It evaluates the capability of balancing vehicle volumes from different approaches. Stops measures the number of stops that vehicles have per mile. Fewer stops smooth the traffic flows and improve fuel efficiency since the high frequency of go-stop movements has a significant impact on fuel consumption [38].

Vehicle costs objective is mainly to minimize the time and dollar costs of vehicles or drivers [136]. Waiting time (WT) measures the cumulative waiting time that vehicles spend at intersections. Delay calculates the extra time that vehicles spend on their trips. It includes WT and the extra time cost when vehicles' speeds are lower than the speed limit of lanes. Travel time calculates the total time that a vehicle spends from an original place to the destination of the vehicle.

Environment friendly measures the fuel consumption and CO_x emissions that a vehicle would consume on a trip. It can be calculated by vehicle fuel consumption models (e.g., Handbook Emission Factors for Road Transport (HBEFA) [137]) based on various factors, such as vehicle mass and driving resistances.

Traffic Signal Control Strategies

Traffic signal control algorithms provide solutions for the signal controllers to adjust traffic signal timing plans. Three stages of TSC algorithms are summarized below according to the historical development [138].

Stage 1 - Pre-timed Strategy: The signal controller of the pre-timed strategy uses the predefined traffic signal timing plans to control traffic signals [138]. It does not detect any real-time traffic data, such as vehicle volume, or speed, to dynamically adjust its signal plans [13]. Instead, it usually observes and learns historical traffic data to find an optimal signal plan for a specific objective, such as reducing vehicle delay [139]. The pre-timed strategy is the least flexible but the most economical way to control signals compared to other strategies because it does not depend on detection devices at intersections and complicated algorithms for dynamic traffic conditions.

Stage 2 - Actuated Strategy: With the development of sensing and V2X communication techniques [140–143], the actuated strategy was proposed to control traffic signals for dynamic traffic conditions. It is a real-time responsive signal control considering the traffic measurements detected by the detectors at intersections [144–147]. Semi-actuated and fully-actuated TSC are the two main types of actuated strategies [13]. Semi-actuated TSC controls traffic signals depending on detection of partial movements, especially the movements on minor streets [144]. It assigns green signals to the movements of minor streets when the traffic conditions of minor streets are fitted to the designed rules or patterns. The major drawback of semi-actuated TSC is that unpredictable minor street conditions may cause congestion issues on major streets [144]. Fully-actuated TSC controls traffic signals based on all detectors at intersections. It improves the performance of semi-actuated TSC but increases the cost and complexity of TSC.

Stage 3 - Adaptive Strategy: Adaptive traffic signal control (ATSC) is the state-of-the-art way to control signals compared to traditional ways. ATSC controls signals based on detectors at intersections and signal control algorithms [13]. Unlike actuated strategy, which uses designed rules and traffic patterns, ATSC is more flexible for adjusting signal timing plans based on real-time traffic conditions. Furthermore, it predicts future traffic conditions and control signals for long-term traffic flows. ATSC is the most costly way to build a traffic signal control system because it usually requires microscopic traffic data,

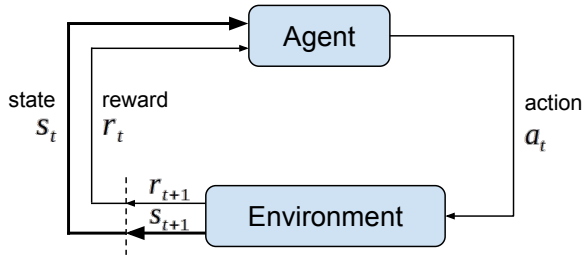


Figure 3.4: Agent-environment interaction in Reinforcement Learning. Every time step t, the agent observes the environment state s_t and takes an action a_t. The environment sends the delay reward r_{t+1} and the new state s_{t+1} back to the agent at time step $t+1$.

such as the instantaneous vehicle speed, position, and volume, to adjust traffic signal timing plans. However, it is the most efficient compared to traditional strategies because it controls signals for dynamic and unforeseeable traffic conditions.

3.2 Reinforcement Learning

Reinforcement learning (RL) is an algorithm that lets a learner agent learn what to do for the environment to maximize its reward [20]. There are four main elements in reinforcement learning: policy, reward, value function, and model of environment [20]. A policy defines mappings from the environment to actions. The agent senses environment states for the policy and receives an action that the agent can execute to affect the environment. Reward frames the objectives or goals of the agent. It is a numeric value that evaluates how good or bad the executed action is. Value function defines the cumulative reward of states. It outputs the expected reward value for the given state. The environment model predicts the future state and reward for the given inputs of states and actions. It is used in model-based RL methods for planning the sequence of state-action pairs, which is the opposite of model-free RL methods.

Figure 3.4 shows the fundamental interactions and iterations between an agent and the environment. At time step t, the agent receives the state s_t from the environment and executes an action a_t based on the state. In the next step $t+1$, a reward r_{t+1} for the action a_t along with the next state s_{t+1} are returned to the agent for the next iteration. Over a set of trial and error learning iterations, the agent tries to learn which action is optimal for a given state of the environment, resulting in the highest reward in the future.

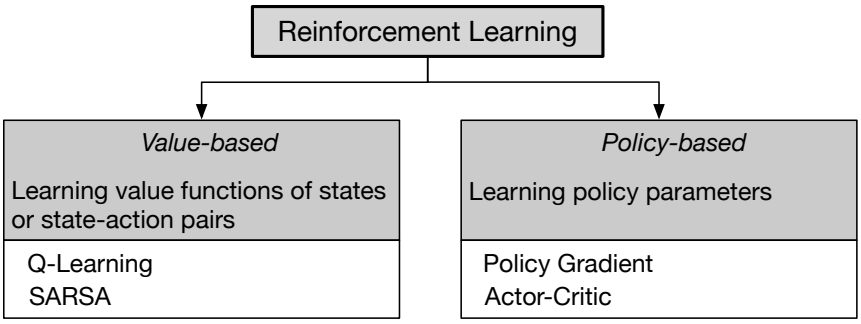


Figure 3.5: The outlines of reinforcement learning methods based on the categories of value-based, policy-based and hybrid methods.

3.2.1 Reinforcement Learning Methods

Reinforcement learning methods can be categorized into value-based and policy-based methods [20, 148]. This section briefly introduces RL methods based on the outlines shown in Figure 3.5.

Value-based Reinforcement Learning

Value-based RL methods try to learn value functions of given states or state-action pairs [20]. Q-learning and state-action-reward-state-action (SARSA) are two value-based RL methods corresponding to off-policy and on-policy temporal-different (TD) learning, respectively [20].

Q-learning [149], defined by Equation 3.1, is an off-policy TD method that estimates state-action values that are independent of the exploring policy [20]. At time step t of the Q-leaning control, state-action Q-value $Q(s_t, a_t)$ is updated by the delayed reward r_{t+1} and the maximum state-action Q-values $Q(s_{t+1}, a)$ of the next step $t+1$, which may be unrelated to the Q-value $Q(s_{t+1}, a_{t+1})$ followed by the exploring policy.

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \tag{3.1}$$

SARSA is an on-policy TD method that estimates state-action values following the exploring policy [20]. At time step t of the SARSA control, state-action Q-value $Q(s_t, a_t)$ is updated by the delayed reward r_{t+1} and the state-action Q-values $Q(s_{t+1}, a_{t+1})$ of the next step $t+1$, which is followed by the exploring policy. The definition of SARSA is

Algorithm 1: Value-based RL method

Parameters: step size $\alpha \in (0, 1]$, discount factor $\gamma \in (0, 1]$, $\epsilon > 0$
Initialize: s_1, a_1, $Q(s, a), \forall s \in \mathcal{S}$
for *steps* $t = 1$ *to* N **do**
  Take action a_t
  Observe the next state s_{t+1}, get delayed reward r_t
  Choose action a_{t+1} based on $Q(s_{t+1})$ using ϵ-greedy
  if *SARSA* **then**
    Update Q value: $Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$
  else
    Update Q value:
    $Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$
  end
  $a_t \leftarrow a_{t+1}$; $s_t \leftarrow s_{t+1}$
end

shown as follows:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \tag{3.2}$$

ϵ-greedy method is commonly used in RL methods. It helps the agent of RL continue to explore new conditions and improve the probability of finding the optimal global solution and avoiding the local optimum issue [20]. The basic algorithm of value-based RL methods, including Q-learning and SARSA, is shown in Algorithm 1.

Policy-based Reinforcement Learning

Policy-based RL methods learn a parameterized policy π_θ for the action selection instead of value functions [20]. It outputs the probability of an action that can be taken by the agent. The one of policy-based RL method is the policy gradient method. It learns policy parameters θ based on the gradient $\widehat{\nabla J(\theta)}$ of parameters. They usually use value functions $v(s, w)$ to calculate the stochastic estimate $\widehat{\nabla J(\theta)}$ for updating parameters θ.

REINFORCE [150] is an episode-based policy gradient method. It uses the cumulative reward G of an episode to estimate $\widehat{\nabla J(\theta)}$ by the following definition [20]:

$$\widehat{\nabla J(\theta)} = \mathbb{E}_\pi \left[G_t \frac{\nabla \pi(a_t|s_t, \theta_t)}{\pi(a_t|s_t, \theta_t} \right]. \tag{3.3}$$

Algorithm 2: One step of Actor-Critic RL method

Inputs: a policy $\pi(a|s,\theta)$ as "actor" with parameter θ, a value function $v(s,\omega)$ as "critic" with parameter ω, current state s
Parameters: $\alpha^{\theta}, \alpha^{\omega}, \alpha^{r} \in (0,1]$, $\gamma^{\theta}, \gamma^{\omega} \in (0,1]$, $\epsilon > 0$
Initialize: s_1, a_1, $Q(s,a), \forall s \in \mathcal{S}$
begin
  $a \sim \pi(\cdot|s,\theta)$
  Take action a
  Observe the next state s' and delayed reward r
  $\delta = r + \gamma^{r} v(s',\omega) - v(s,\omega)$
  $\omega = \omega + \alpha^{\omega}\delta\nabla v(s,\omega)$
  $\theta = \theta + \alpha^{\theta}\delta\frac{\nabla\pi(a|s,\theta)}{\pi(a|s,\theta)}$
  $s \leftarrow s'$
end

Value function $v(s,w)$ can be used in REINFORCE to reduce variance of gradient by the following definition [20]:

$$\widehat{\nabla J(\theta)} = \mathbb{E}_{\pi}\left[(G_t - v(s_t,w))\frac{\nabla\pi(a_t|s_t,\theta_t)}{\pi(a_t|s_t,\theta_t}\right]. \tag{3.4}$$

The actor-critic method is another policy-based RL method. It learns both value functions and policy parameters at the same time in learning progress [20]. The actor-critic method uses "actor" to learn a policy for action selection and "critic" to learn a value function for improving the performance of "actor". It is an online method for both episode-based and continuing problems. In every learning step, "critic" learns the value function and outputs the TD errors δ for "actor". The policy of "actor" calculates the estimated gradient $\widehat{\nabla J(\theta)}$ based on the generated TD errors δ. Algorithm 2 represents the one-step learning progress of the actor-critic method.

3.2.2 Deep Reinforcement Learning Methods

Deep reinforcement learning is a technique that takes the advantages of deep learning (DL) techniques to address the high-dimensional problems [148]. Due to a large number of discrete states in TSC problems, tabular-based RL methods are unable to store the states in arrays or matrices. However, DRL effectively addresses the issue by utilizing Q-network

Algorithm 3: Value-based DRL algorithm

Parameters: discount factor $\gamma \in (0, 1]$, small $\epsilon > 0$, updating delay C for target DQN
Initialize: DQN Q with parameters θ
Initialize: Target DQN $\hat{Q}$ with parameters $\theta^- = \theta$
Initialize: Experience memory pool D
begin
  Take action a based on state s using ϵ-greedy algorithm
  Observe the next state s', reward r
  Store experience $[s, a, r, s']$ in D
  Sample a batch of experience from D
  Set $y = r + \gamma max_{a'} \hat{Q}(s', a'; \theta_i^-)$
  Perform a gradient descent step to update parameters θ of $Q(s, a, \theta)$ using $(y - Q(s, a, \theta))^2$
  if *steps* % C = *0* **then**
    $\hat{Q} \leftarrow Q$
  end
  $a \leftarrow a'$; $s \leftarrow s'$
end

to generalize the continuous states to a hyperspace. Such generalization helps the agent extract features from the input and find the similar encountered states in the previous experience.

DRL uses deep Q-network (DQN) as a value function to estimate state or state-action values [151] (see Algorithm 3) or as a policy for continuing action space [114] (see Algorithm 4). DQN is a deep neural network that uses a convolutional neural network (CNN) to generalize raw visual states to hyperspace so that the agent can learn patterns from the high-dimensional state space. Two techniques are applied in DQN to address the problem of over-estimated Q-values in DRL methods: experience replay [152] and target network [151]. Experience replay stores the state-action transitions $\langle s_t, a_t, s_{t+1}, r_{t+1} \rangle$ in memory so that the agent can sample the past experience for training the DQN. Several improvements have been proposed, such as batch sampling and prioritized experience replay, to improve the efficiency of learning progress and reduce the variance of parameter updates [153]. The Target network is another DQN maintained by the agent with a lower updating frequency of parameters compared to the learning DQN. The architecture of the target DQN is the same as the learning DQN except for the learning frequency. The Lower learning frequency

Algorithm 4: Policy-based DRL algorithm

Parameters: discount factor $\gamma \in (0, 1]$
Initialize: DQN $Q_\pi(s, \theta)$ as the policy of "actor"
Initialize: Value function $Q_v(s, a, w)$ as "critic"
Initialize: Experience memory pool D
begin
    Take action a based on state s by policy $Q_\pi(s, \theta)$
    Observe the next state s', reward r
    Store experience $[s, a, r, s']$ in D
    Sample a batch of experience from D
    Set $y = r + \gamma max'_a Q(s', a', w)$
    Set loss $\mathcal{L} = (y - Q(s, a, w))^2$
    Update w of $Q(s, a, w)$ to minimize the loss $\mathcal{L}$
    Perform a gradient descent step to update θ of $Q_\pi(s, \theta)$ using loss $\mathcal{L}$
    $a \leftarrow a'$; $s \leftarrow s'$
end

of the target DQN helps the agent keep the earlier parameters to reduce the fluctuating estimates of the Q-values.

3.3 Summary

This chapter presents the preliminaries of traffic signal control and reinforcement learning techniques. Section 3.1 provides an overview of a traffic signal control system at intersections. The basic concepts of traffic signals are firstly presented to explain the signal definitions, users, and how the signals are designed to regulate user movements at an intersection. Also, Section 3.1 summarizes the objectives and performance measures of traffic signal control problems and three signal control strategies. Section 3.2 introduces the preliminaries of RL algorithms. The natural RL algorithms are presented based on the classifications of value-based and policy-based groups. Moreover, the state-of-the-art RL algorithms based on deep learning techniques are introduced to better understand the algorithms in the following chapters.

Chapter 4

Traffic Signal Control for Isolated Intersection

4.1 Introduction

In this chapter, we propose to improve the efficiency of traffic control for an isolated intersection by deep reinforcement learning-based traffic signal control and vehicular speed profile control. The approach utilizes deep reinforcement learning (DRL) techniques that take the advance of both deep learning and reinforcement learning techniques so that the signal controller agent can learn a high-dimensional traffic environment and adjust traffic signal timing plans efficiently. Moreover, based on the improved adaptive traffic signal control policy, the approach provides the optimal speed profiles for the approaching vehicles to smooth the traffic flows and improve the vehicle fuel economy. In this chapter, we refer to the proposed TSC method for an isolated intersection as Fuel-Eco TSC (FECO-TSC) method.

The first benefit of the FECO-TSC approach is that the agent can control traffic signals to regulate traffic flows on multiple objectives smoothly. With a one-objective TSC approach, the agent performs well on the objective but can not guarantee the performance on the other conflict objectives. For example, the objective of minimizing waiting queue length at an intersection can not guarantee that the agent can minimize the waiting time of approaching vehicles because the long waiting time of vehicles can happen on the lanes

with short waiting queue length.

Another benefit of the proposed TSC approach is that the optimal speed profiles for approaching vehicles are provided based on the adaptive traffic signal control policy instead of the pre-timed traffic signal control policy. A speed profile describes the motion information, including the instantaneous velocity and acceleration/deceleration of the vehicle. Controlling vehicle speed profiles prevents hard accelerating or decelerating movements and reduces the number of stop-go movements, leading to smooth traffic flows and improving the efficiency of vehicle fuel consumption. The approach allows the agent to provide the optimal instantaneous speed profiles for on-road vehicles according to the dynamic traffic signal control policy.

This chapter first describes the architecture of the Fuel-ECO TSC approach. Then the agent designs are explained to formulate the TSC problem to the DRL-based TSC method for multiple objectives. Following the section on the agent designs, the TSC strategy is described for controlling traffic signals on multiple objectives. An approach of speed control for vehicles based on the dynamic traffic signal control is then explained to further smooth traffic flows and improve the fuel efficiency of the approaching vehicles at signalized intersections. Finally, we describe the experimental environment and results to demonstrate the effectiveness of the proposed method.

4.2 DRL-based Traffic Signal Control Method

4.2.1 Framework

RL is an advanced algorithm based on Markov Decision Processes (MDPs) [20]. The agent in RL takes action to impact the environment and gets a reward for the executed action. Based on RL's architecture, the framework of FECO-TSC is mainly composed of a learner agent and two control components to impact the environment. The agent learns traffic states and trains a TSC model containing DQNs for multiple objectives to find the optimal actions. Two control components utilize the actions to impact the traffic states. Control traffic signals (CTS) component controls traffic signals by the action generated from the TSC model to smooth the traffic flows. Control speed profiles (CSP) component

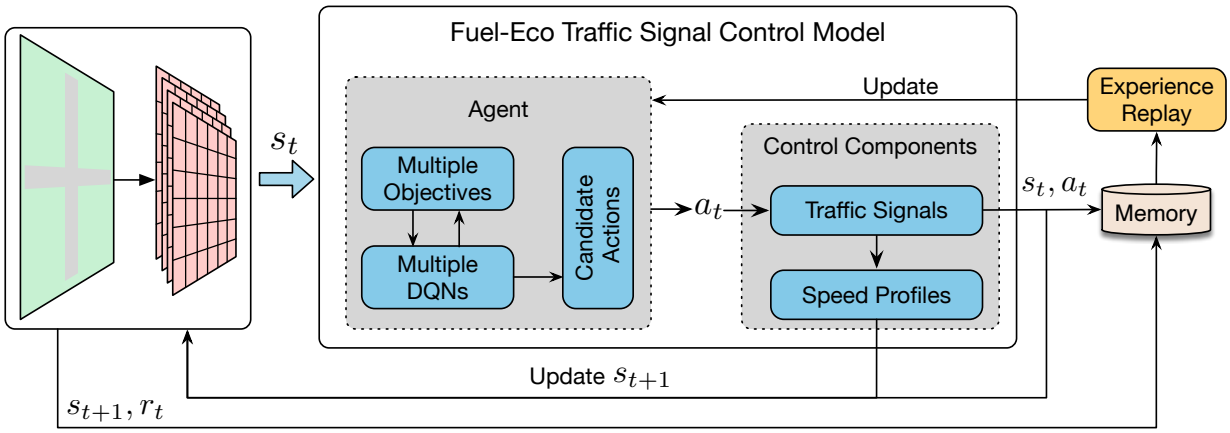


Figure 4.1: The framework of DRL-based Fuel-ECO TSC.

controls the instantaneous speed profiles (speeds and acceleration/deceleration) of moving vehicles based on traffic signals to improve driving styles.

Figure 4.1 shows the interactions between the traffic environment, agent, and control components. At time step t, the agent collects the traffic environment information via V2I techniques and transforms to state representation s_t. Given the state s_t, the agent chooses an action a_t based on the TSC model to find the optimal TSP for the current traffic state s_t. Based on the TSP information, the CTS component controls the TSP by keeping the current TSP or switching to the new TSP, and the CSP component controls the speed profiles of the moving vehicles in the intersection area according to the TSP information. Both controls of CTS and CSP components have impacts on the traffic state s_{t+1} at time step $t+1$. CTS impacts the moving vehicles by controlling traffic signals, which allow vehicular movements on specific lanes by green signals and stop vehicular movements on the rest of the lanes. CSP impacts traffic states by adjusting vehicle speed profiles. It controls the speeds and acceleration/deceleration of moving vehicles, leading to changes in vehicle movements and positions. The state s_{t+1} is then returned from the traffic environment with the reward r_t of the state-action pair $\langle s_t, a_t \rangle$ at time step t, and used as the input of the TSC model for the next time step. The agent's experience at time step t is defined as $\langle s_t, a_t, r_t, s_{t+1} \rangle$. It is stored in memory for the agent to learn and update the TSC model periodically by experience replay method [151].

The following sections introduce the designs of the agent and two control components in the framework.

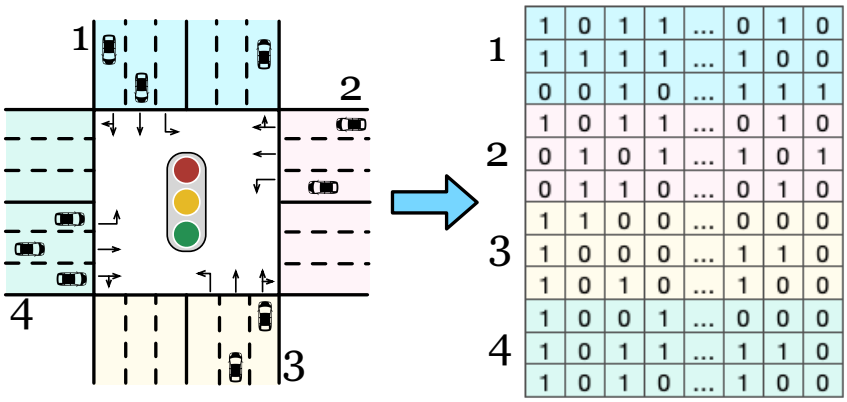


Figure 4.2: Intersection and state representation.

4.2.2 Agent Design

Traffic state representation

We proposed an image-based traffic state to represent the traffic environment since the DL technique has the ability to extract the features and generalize the input to a hyperspace. Such transformation significantly reduces the traffic state space's size and enhances the feature representation for the traffic environment.

Figure 4.2 shows our image-based state representation. In our design, the state is represented as a four-channel matrix $M_{L\times n\times 4}$, where L denotes the number of lanes, n denotes the capacity of a lane, and 4 denotes the depth of the matrix, including the presence of a vehicle, the vehicle's velocity, the acceleration/deceleration of the vehicle, and the index of the traffic signal phase. The first channel stores the positions of approaching vehicles at an intersection. To more accurately indicate vehicles' presence through a matrix, we map lanes into a grid structure in which each grid's length is equal to the average vehicle length plus the minimum safe interval between vehicles. Then, we adopt the position representation method introduced in [62], where the cell value is 1 if there is a vehicle in the grid cell. The second and third channels store the speed profiles of vehicles. With the same dimension of the first channel, we put vehicle speed data in the second channel and vehicle acceleration/deceleration information in the third channel. We put the TSP information in the fourth channel matrix. The cell value in the fourth channel matrix is a binary value where 1 denotes that the traffic signal is green for the vehicle in the cell areas, and 0 denotes a red or yellow traffic signal.

Action

Action space defines the actions that the agent can take to impact the environment [20]. Our action space is defined as $\mathbf{A} = \{a_1, a_2, a_3, a_4\}$ to represent four TSPs respectively. The selected action by the agent means that the agent will receive higher rewards in the following steps.

Reward

The reward is a critical part of the DRL technique. It frames the application's objectives and evaluates how good or bad the executed action is [20]. Here, we design three reward models to evaluate actions for three objectives as follows:

Minimize AWT: AWT denotes the average time that a vehicle waits at an intersection area before going through the intersection. Longer AWT contributes to extra fuel consumption and longer delay. Here, we design the reward of the objective as the difference of AWT between two executing actions. The reward returns a positive value if the average waiting time of approaching vehicles at an intersection decreases. Otherwise, it returns a negative value,

$$R_{AWT}^{(t)} = -(\bar{w}^{(t+1)} - \bar{w}^{(t)}), \tag{4.1}$$

where $\bar{w}^{(t)}$ denotes the average waiting time of approaching vehicles in the state matrix at step t.

Minimize AQL: AQL denotes the number of vehicles waiting at an intersection. The objective is to balance AQL among incoming lanes to avoid congestion in some directions. The incoming lanes with longer waiting queue lengths indicate higher priority for accessing green signals. In order to receive a higher reward if the agent reduces the higher waiting queue length, we design the reward for the objective as the difference of the cumulative sum of waiting queue length between two executing actions,

$$R_{AQL}^{(t)} = -\frac{1}{L}\left(\sum_{l\in L}\sum_{j=1}^{q_l^{(t+1)}} j - \sum_{l\in L}\sum_{k=1}^{q_l^{(t)}} k\right), \tag{4.2}$$

where $q_l^{(t)}$ denotes the number of waiting vehicles on the lane l in the state matrix at step t.

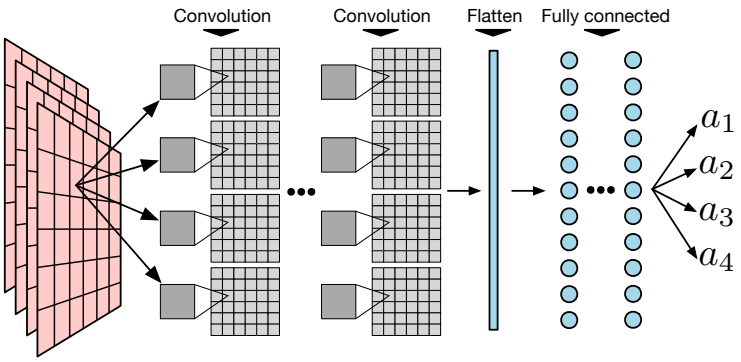


Figure 4.3: The architecture of Deep Q-Network (DQN).

Minimize AD: delay is defined as the extra time that a vehicle travels [13]. The lower delay means that the vehicle moves through the intersection at a higher speed. We design the reward as the difference of the average delay between two executing actions,

$$R_{AD}^{(t)} = -(\bar{d}^{(t+1)} - \bar{d}^{(t)}), \tag{4.3}$$

where $d^{(t)}$ denotes the average delay of moving vehicles in the traffic state matrix s_t. To evaluate the control of traffic signals and speed profiles, we adopt the delay definition based on speed profiles in [60] to evaluate the delay of vehicles. The delay is a function of average vehicle speed, meaning that moving vehicles' speed profiles have a lower delay with higher moving speed. The delay of vehicles is calculated by the following equations:

$$d = 1 - \frac{\sum_{i=1}^{V} v_i^{(t)}}{V \times v_{max}}, \tag{4.4}$$

where v_{max} is the legal speed limit for the corresponding road segment, $v_i^{(t)}$ denotes the measured speed of the vehicle i at time step t.

Traffic Signal Control Model

TSC model is composed of several DQNs for multiple objectives. Each of the DQNs learns traffic states and outputs the optimal action based on its corresponding objective's instantaneous traffic states. Figure 4.3 shows the architecture of the individual DQN. The given state matrix is mapped to action Q values by passing through convolutional layers and fully connected layers. Since each row of the matrix represents the traffic flow of a lane, we apply the filter with size 1×3 and stride 2 in the convolutional layers to extract

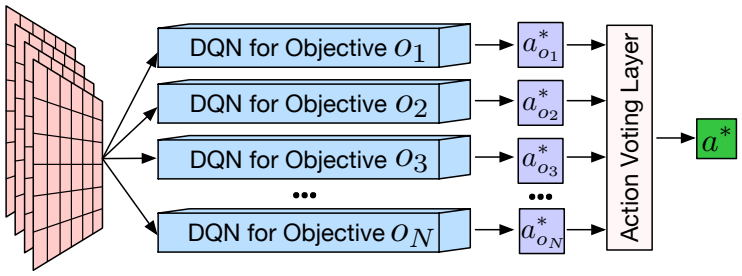


Figure 4.4: The architecture of DRL-MRs framework using Deep Q-Network (DQN).

features of each lane individually. The convolutional layer's output is then flattened into a vector as the input of the first fully connected layers. The optimal action has the highest Q value since the Q value quantifies the agent's rewards in the future steps.

To train the DQN, we store the agent's experiences $e = \langle s, a^*, r, s' \rangle$ at each time step. Experience replay technique [151, 152] on the samples from the memory, and Q-learning algorithm [20] are performed to update DQN's weights θ.

Based on the DQN architecture above, we apply a DRL-based framework with multiple resources of rewards (DRL-MRs) for multiple objectives [154] to train the agent so that the agent can perform effectively on multiple objectives. DRL-MRs combines several DQNs to learn policies for multiple objectives and controls traffic signals more stably by action voting policy (AVP). The architecture of DRL-MRs is shown in Figure 4.4, where $\mathcal{O}$ denotes the collection of objectives, and $\mathcal{Q}$ represents DQNs. According to its own predetermined objective $o \in \mathcal{O}$, each DQN $Q_o \in \mathcal{Q}$ calculates the corresponding optimal solution a_o based on the input state matrix at each time step. Finally, the action voting layer computes AVP's optimal action from all candidate actions $\mathcal{C}$ (see Eq. 4.5). We also apply ϵ-greedy algorithm to select actions, in which case if votes of every action are equal, the optimal action is randomly selected from all candidate actions,

$$a^* = \begin{cases} \text{argmax}_{a \in \mathbf{A}} \, \mathcal{V}(a), & 1 - \epsilon \\ \text{random}_{a \in \mathcal{C}} \, a, & \epsilon \text{ or } \mathcal{V}(a_i) = \mathcal{V}(a_j), \forall i \neq j \end{cases}, \tag{4.5}$$

where $\mathbf{A}$ denotes the action space, $\mathcal{C}$ denotes candidate actions, a_i denotes the action with index i in $\mathbf{A}$, $\mathcal{V}(a)$ denotes the number of the appearance of action a in $\mathcal{C}$.

4.2.3 Control Traffic Signals

TSP is a traffic signal pattern used by the agent to regulate traffic flows to avoid intersections' conflict movements. Based on the output of the TSC model, the optimal TSP is set to the intersection in order to improve the efficiency of TSC, such as reducing the waiting time of vehicles in the intersection areas. However, due to the unpredictable and changing traffic states, the TSC model may produce different actions in a short period, leading to the issue of flickering TSP. Such an issue causes a higher frequency of vehicular stop-go movements along with higher VFC. Here, we adopt the way in previous studies [66, 84, 91] to solve the issue by setting a minimum duration Δt seconds for each TSP.

Keeping the current TSP or switching to the new TSP are two ways of controlling traffic signals. These are achieved by adjusting the duration of the current TSP. We assume that the optimal action at time step t for the state s_t is a_t and the TSP is p_t. If the corresponding TSP p_{a_t} of a_t is the same as p_t, meaning that keeping the current TSP is optimal for the current traffic state, then the agent extends the duration of p_t by Δt seconds. Otherwise, we compare the elapsed time e_{p_t} with Δt to ensure that the duration of p_t is long enough for the vehicular movements to pass through the intersection. In this case, the agent switches the current TSP p_t to the new TSP p_{a_t} after a 5-second yellow signal phase when the elapsed time of the current TSP p_t is greater than Δt. Based on the control strategy above, the duration T_{p_t} of the current TSP p_t can be defined as follows:

$$T_{p_t} = \begin{cases} e_{p_t} + t_Y, p_t \neq p_{a_t}, e_{p_t} \geq \Delta t \\ \Delta t + t_Y, p_t \neq p_{a_t}, e_{p_t} \leq \Delta t \\ e_{p_t} + \Delta t + t_Y, p_t = p_{a_t} \end{cases}, \tag{4.6}$$

where t_Y denotes the duration of 5-second yellow signal.

4.2.4 Control Speed Profiles

This section describes a speed profile control method that can smooth vehicular movements by controlling speed profiles of on-road vehicles presenting in the state matrix based on the TSP information. The control method is based on the idea of trying to reduce the number

of vehicle stops and keep the vehicles moving at high speed for a lower delay. Further, instead of using fixed traffic signal timings, the method calculates the speed profiles of moving vehicles based on the one-step TSP provided from the DRL-based TSC.

Here, we formulate the expected time for a leading vehicle to pass through the designated intersection as follows,

$$T_e^{(t)} = \begin{cases} t_{acc} + \dfrac{D_{0,l}^{(t)} - D_{acc}}{v_{max}} & \text{if } D_{0,l}^{(t)} \geq D_{acc}; \\ \dfrac{\sqrt{{v_{0,l}^{(t)}}^2 + 2acc_{max} D_{0,l}^{(t)}} - v_{0,l}^{(t)}}{acc_{max}}, & \text{otherwise;} \end{cases} \tag{4.7}$$

where $D_{acc} = v_i^{(t)} t_{acc} + \frac{1}{2} acc_{max} t_{acc}^2$ and $t_{acc} = \frac{v_{max} - v_{0,l}^{(t)}}{acc_{max}}$. $v_{0,l}^{(t)}$ and $D_{0,l}^{(t)}$ denote the measured speed of the leading vehicle and its distance to the intersection at time step t, respectively. acc_{max} is the maximum acceleration, t_{acc} is the time that the vehicle accelerates to v_{max}, and D_{acc} is the corresponding moving distance when the speed reaches v_{max}.

Based on the expected-through time $T_e^{(t)}$ of the leading vehicle, we let the leading vehicle travel through the intersection as fast as possible if $T_{p_t} \geq T_e^{(t)}$ and the signal is green for the vehicle, meaning that the duration of the current TSP is long enough for the leading vehicle safely through the intersection. Otherwise, the leading vehicle has to decelerate until the speed is 0 km/h, or the signal turns to green. We gently decelerate the leading vehicle as early as possible, using Eq. 4.8, to avoid hard deceleration. Furthermore, gentle deceleration delays the leading vehicle stop at the stop line, leading to less waiting time compared to the earlier vehicle stops.

$$dec = max \left\{ -\frac{{v_{0,l}^{(t)}}^2}{2D_{0,l}^{(t)}}, dec_{max} \right\}, \tag{4.8}$$

where dec_{max} denotes the maximum deceleration.

Moreover, we also need to control the instantaneous speed profiles of the vehicles following the leading vehicle. Accordingly, we define the speed of the vehicle i that is immediately following the leading vehicle l at time t as $v_{i,l}^{(t)}$ $(i \geq 1)$. Then the instantaneous target speed $\hat{v}_{i,l}^{(t)}$ is set as the speed of the vehicle $i-1$ represented as $v_{i-1,l}^{(t)}$. Based on the idea of gently controlling vehicle speed profiles, we control the acceleration/deceleration of the following

Algorithm 5: FECO-TSC algorithm

Initialize: Agent g, time step t, minimum duration Δt
begin
  g collects traffic state s_t
  g feeds s_t to the defined DQNs representing multiple objectives $\mathcal{O}$
  g collects the candidate actions $\mathcal{C} = \{a_o^*\}, o \in \mathcal{O}$ based on Q-values of DQNs
  g chooses the optimal action a_t based on Eq. 4.5
  g maps a_t to the TSP p_{a_t}
  begin control traffic signals
    if $e_{p_t} < \Delta t$ **then**
      Keep current TSP p_t
    else
      if $p_t == p_{a_t}$ **then**
        Extend p_t by Δt
      else
        Switch to p_{a_t} after yellow signal
      end
    end
  end
  begin control speed profiles
    Calculate the current TSP duration T_{p_t} by Eq. 4.6
    for *lane* $l \in L$ **do**
      for *vehicle* $i \in$ *vehicles* **do**
        if $i == 0$ **then**
          // this is leading vehicle
          Calculate the $T_e^{(t)}$ by Eq. 4.6
          if $T_{p_t} \geq T_e^{(t)}$ **then**
            Through the intersection freely
          else
            Set the speed profiles by Eq. 4.8
          end
        else
          Set the speed profiles by Eq. 4.9
        end
      end
    end
  end
  Update s_t to s_{t+1} by the new speed profiles and vehicle positions
end

vehicle i based on the target speed $v_{i-1,l}^{(t)}$ by the equation as follows:

$$acc/dec = \frac{(\hat{v}_{i,l}^{(t)})^2 - (v_{i,l}^{(t)})^2}{2(D_{i,l}^{(t)} - D_{i-1,l}^{(t)})}, \tag{4.9}$$

where $\hat{v}_{i,l}^{(t)} = v_{i-1,l}^{(t)}$ denotes the target speed of the vehicle i. acc/dec is constrained by the predetermined maximum acceleration acc_{max} and maximum deceleration dec_{max}, respectively.

The full algorithm of FECO-TSC is represented in Algorithm 5. At controlling time step t, lines 3-7 find the optimal TSP p_{a_t} for the traffic state s_t. Lines 8-18 are the algorithm of the CTS component. The algorithm of CSP is represented in lines 19-36. The new speed profiles and vehicle positions are then updated the traffic state s_{t+1} at time step $t+1$. We limit the frequency of the CTS component by the minimum TSP duration Δt to reduce computation complexity, meaning that the agent controls the traffic signals per Δt s. In comparison, the frequency of the CSP component is higher than the frequency of the CTS component. The agent controls speed profiles per 1s to avoid a two-vehicle crash.

4.3 Experiments and Results

4.3.1 Simulation Environment

Here, we adopt the SUMO [70] to conduct the simulation experience by considering two scenarios:

Synthetic scenario

Here, we first set up a hypothetical experimental scenario, that is, an arbitrary four-leg intersection, in which every leg (incoming road segment) has two lanes (see Figure 4.5). Lane 1 and Lane 2 are used for go-through (or right-turn) vehicular movements and left-turn vehicular movements, respectively. The length of each incoming and outgoing street is 150 meters. Left-turn protected TSP design was adopted in this scenario (see Figure 3.3). TSP 1 and TSP 3 set the green signal to the lanes for go-through and right-turn movements in East-West and North-South direction, respectively. Accordingly, TSP 2 and TSP 4 set

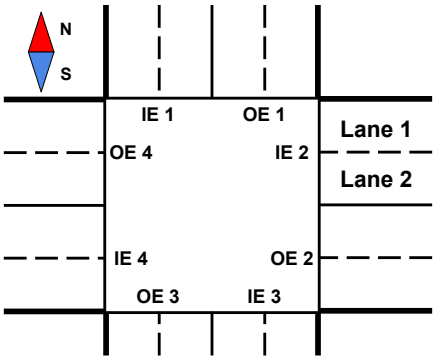


Figure 4.5: Intersection environment. IE: Incoming Edge, OE: Outgoing Edge.

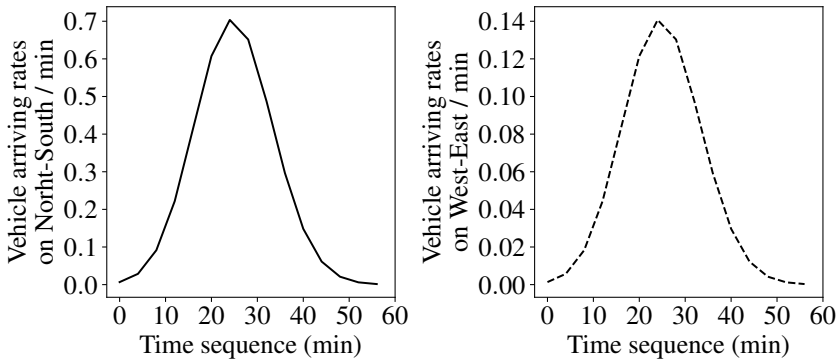


Figure 4.6: Vehicle arrival rates in the synthetic scenario

the green signal to the lanes for left-turn movements in the corresponding lane-directions, respectively. We initially set 60 seconds for TSP 1 and TSP3, and 15 seconds for TSP 2 and TSP 3. To approximate the real-world scenarios, we set a five-second yellow signal between two adjacent phases.

Further, an unbalanced traffic flow generated by binomial distributed vehicle arrival rates is adopted to conduct the experiments, see Figure 4.6. Here, the arrival rates denote the probability of emitting a vehicle each second. We can see that the vehicle demand starts increasing at around 10 minutes and decreasing after around 30 minutes. We set the unbalanced vehicle demands in different directions by setting vehicle arriving rates in North-South direction roughly 5 times more than West-East directions. By adopting the setting introduced in [154], we set the following turning rates of all approaching vehicles, i.e., 40% for go-through movements, 35% for right-turn movements and 25% for left-turn movements.

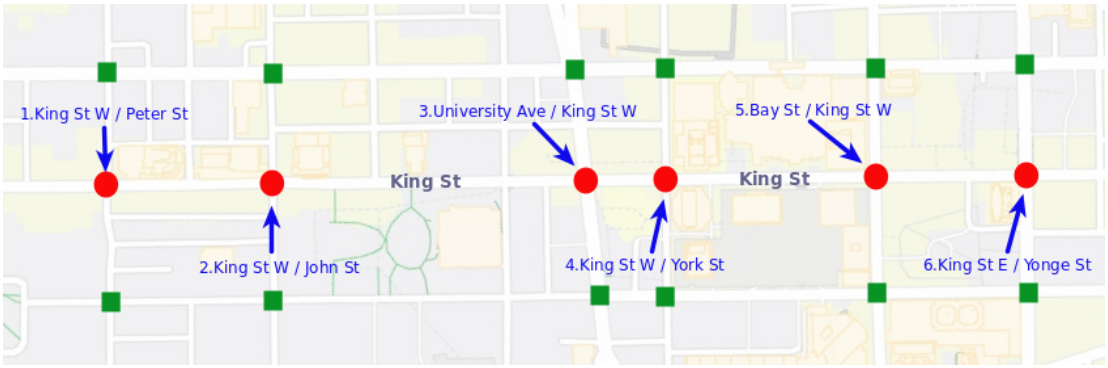


Figure 4.7: King St. intersection map in Toronto generated by OpenStreetMap[1].

Real-world scenario

Besides the synthetic scenario, we also adopt the traffic flow data recorded by the City of Toronto [155], i.e., six signalized intersections on King St. in Toronto. The names of intersections are 1) King St. W/Peter St. (I-KP); 2) King St. W/John St. (I-KJ); 3) University Ave./King St. W (I-UK); 4) King St. W/York St. (I-KWY); 5) Bay St./King St. W (I-BK); 6) King St. E/Yonge St. (I-KEY), respectively (see Figure 4.7). More specifically, the traffic data of six intersections that are collected by using a video-based counting system and aggregated by 15-minute time bins. It counts the volumes of five types of users, including articulated trucks, bicycles, lights, pedestrians and single-unit trucks on four approach legs (N, S, E, W) and directions of travel (NB, SB, EB, WB), respectively. In our test, we evaluated our method on the traffic flows of light vehicles recorded on February 12th, 2018 in the time period of 6:00 to 20:00.

To conduct the simulation, we calculate the turning ratios of every leg based on the original incoming volume bins per 15 minutes. We assume that the right-turn ratio, the through ratios and the left-turn ratios are represented as R_r, R_t and R_l, respectively. Hence, the turning ratios of legs (N,E,S,W) ares N_{R_r}, N_{R_l}, N_{R_t}, E_{R_r}, E_{R_l}, E_{R_t}, S_{R_r}, S_{R_l}, S_{R_t}, W_{R_r}, W_{R_l}, W_{R_t}, respectively. And the incoming volumes per 15 min of approach legs are V^N_{SB}, V^S_{NB}, V^E_{WB}, V^W_{EB}, and the outgoing volumes are V^N_{NB}, V^S_{SB}, V^E_{EB}, V^W_{WB}, respectively. Then we can get a solution of the turning ratios of all legs by a system of equations as

[1]https://www.openstreetmap.org

follows,

$$V_{NB}^{N} = V_{WB}^{E} \cdot E_{R_r} + V_{NB}^{S} \cdot S_{R_t} + V_{EB}^{W} \cdot W_{R_l}, \tag{4.10}$$

$$V_{SB}^{S} = V_{SB}^{N} \cdot N_{R_t} + V_{WB}^{E} \cdot E_{R_l} + V_{EB}^{W} \cdot W_{R_r}, \tag{4.11}$$

$$V_{EB}^{E} = V_{SB}^{N} \cdot N_{R_l} + V_{NB}^{S} \cdot S_{R_r} + V_{EB}^{W} \cdot W_{R_t}, \tag{4.12}$$

$$V_{WB}^{W} = V_{SB}^{N} \cdot N_{R_r} + V_{NB}^{S} \cdot S_{R_l} + V_{WB}^{E} \cdot E_{R_t}, \tag{4.13}$$

subjecting to $N_{R_r} + N_{R_l} + N_{R_t} = 1$, $S_{R_r} + S_{R_l} + S_{R_t} = 1$, $W_{R_r} + W_{R_l} + W_{R_t} = 1$ and $E_{R_r} + E_{R_l} + E_{R_t} = 1$.

Evaluation method of fuel economy

To evaluate the fuel economy, in this paper, the method introduced in Handbook Emission Factors for Road Transport (HBEFA) [137] is adopted to evaluate the instantaneous VFC. In detail, HBEFA estimates VFC based on seven emission standards from EURO 0 to EURO 6 according to various factors, such as vehicle mass, driving resistances, loss in the transmission system. Here, we adopt EURO 4, which is the default setting in SUMO, as our emission standard for gasoline passenger cars to guarantee all vehicles have the same traffic situation, driving resistance and transmission loss.

4.3.2 Compared Methods

We conducted two rounds of performance comparisons with TSC and eco-driving counterparts. First, we compared with fixed time TSC policy and pure FTM-based eco-driving methods to show that FECO-TSC achieves a success in VFE collaborating with eco-driving technology.

1) Fixed Time Model (FTM): We set 60 seconds for phase p_1 and p_3, 15 seconds for phase p_2 and p_4, and 5 seconds for yellow traffic signals between two phases.

2) FTM-based Speed Guidance Strategy (SGS) [40]: SGS is an eco-driving strategy based on a car-following model for the leading vehicles in successive signalized intersections. It calculates the target velocities and trajectories for the leading vehicles through the intersections with lower fuel consumption and stop times. The approach

improves the performance of the leading vehicles in the traffic environment with FTM settings.

3) FTM-based Speed & Acceleration/Deceleration Control (SADC) [41]: SADC is an instantaneous eco-driving strategy for the approaching vehicles in the traffic environment with successive signalized intersections. It estimates the vehicles' feasible vehicle speed and acceleration based on the predicted queue lengths and a car-following model. The approach demonstrates the effectiveness of the traffic environment with FTM settings.

Second, we compared our method with FTM and pure DRL-based TSC methods to evaluate the effectiveness of our method.

1) TSC using DRL for Single Objective with single reward (TSC-SO) [63]: TSC-SO applies DRL techniques to develop an adaptive TSC agent for the objective of minimizing average travel delays. The reward model is defined as the difference in vehicle delays between two successive actions.

2) TSC using DRL for Multiple Objectives with scaled reward (TSC-MO) [60]: TSC-MO applies DRL techniques to develop an adaptive TSC agent considering multiple traffic measurements. The reward model is defined as a scaled value by combining multiple weighted traffic measurements, including vehicle delays, waiting time, crashes, stops, and the switching status of traffic signal phases, in order to improve the performance of the agent on different objectives.

4.3.3 Parameter Settings

Here, we provide detailed information on the simulation parameters' setting. We assume that the length of each vehicle is 4 meters. The minimum gap between the two cars is 1 meter. Accordingly, for representing the traffic state, the length of the grid will be 5-meter. To simulate the motion of vehicles, We adopt the Krauss car-following model with 50 km/h as maximum vehicle speed, 2.6 m/s^2 as maximum acceleration, -4.5 m/s^2 as maximum deceleration.

Table 4.1 presents the parameters of training DQN progress referring to the parameters in [62, 87]. The ϵ decays from 0.5 to 0.01 by a factor 0.99999 at every time step. We adopt

Table 4.1: Parameters of DQN with DRL-MRs framework

Parameters	**Value**
Experience memory pool D size	1000 [87]
Starting ϵ	0.5
Ending ϵ	0.01
Discount factor γ	0.8 [87]
Learning rate α	0.001 [62]
Action time interval Δt	5 s [87]
Leaky ReLU β	0.01 [62]
Target update delay steps	300 [87]

the action time interval $\Delta t = 5$s used in [87], which is the minimum duration of a TSP.

Based on the DQN architecture shown in Figure 4.3, the first hidden layer convolves 8 filters with size 1×3 and stride 2 on the input state matrices. The second hidden layer is a fully-connected layer consisting of 128 Leaky ReLU units. This is followed by a third fully-connected layer consisting of 64 Leaky ReLU units. The output layer is a fully-connected layer with 4 outputs for 4 actions, each of which represents the corresponding TSP.

4.3.4 Evaluation Metrics

We evaluated the performance of our method by the following metrics.

1) Vehicle Fuel Consumption (VFC): It is the total fuel consumption of vehicles going through intersections. Lower VFC denotes higher vehicle fuel efficiency.

2) Average Waiting Time (AWT): It is the average cumulative waiting time whenever a vehicle stops. Lower AWT denotes that vehicles spend less waiting time at intersections, leading to less extra fuel consumption.

3) Average Delay (AD): It is the delay of vehicles measured by definition in Section 4.2.2. The lower AD denotes vehicles move faster in the trips, contributing to lower fuel consumption since a vehicle consumes less fuel when the moving speed is close to around 70 km/h [137].

4) Average Waiting Queue Length (AQL): It is the average number of waiting vehicles at an intersection. Lower AQL denotes fewer vehicles queueing at the intersection, meaning a lower probability of congestion occurring.

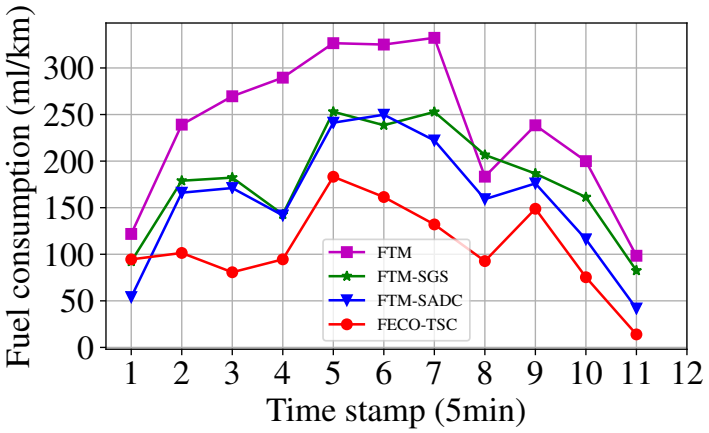


Figure 4.8: Performance on synthetic data with respective to vehicle fuel consumption.

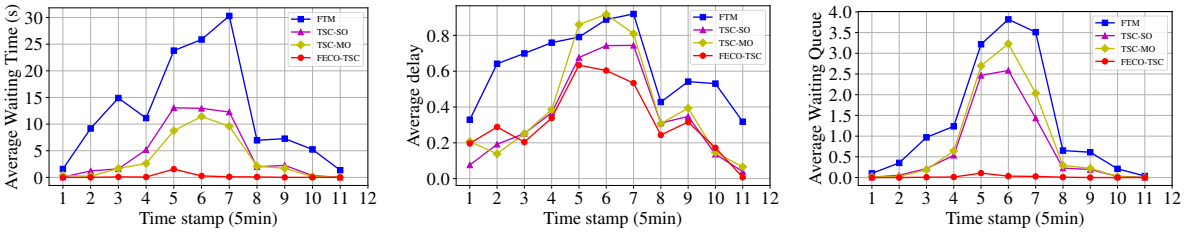
5) Stability of TSC: It compares the stability of TSC methods when the traffic volume changes dynamically, especially when the traffic volume increases in a short period. Higher stability denotes that the VFC, AWT, AD, and AQL have lower standard deviations during the period of changing traffic flows.

4.3.5 Simulation Results on Synthetic Scenario

Overall Analysis

We first compare our method with other baseline methods on one intersection scenario. Figure 4.8 shows the average VFC per 5 minutes, corresponding to the changing vehicle arrival rates. We can see that our FECO-TSC method had the best performance. The highest VFC per 5 minutes of the FECO-TSC method was less than 200 ml/km during the peak time. However, the average VFC of FTM increased dramatically when the vehicle arrival rates rose and surpassed 300 ml/km around peak time. Moreover, the peak-time VFCs of SGS, SADC performed better than FTM but fluctuated around 250 ml/km. The results suggest both eco-driving strategies and the FECO-TSC method can significantly reduce VFC due to speed profile control and delay reduction, respectively. Moreover, our FECO-TSC achieved a higher amount of VFC savings, indicating TSC collaborating with eco-driving strategy provides a better solution to VFE by reducing vehicle stops, delays, and speed loss of moving vehicles.

Figure 4.9a demonstrates that our FECO-TSC outperformed FTM, TSC-SO, and TSC-



(a) Average waiting time. (b) Average delay. (c) Average waiting queue.

Figure 4.9: Performance on synthetic data w.r.t. AWT, AD and AQL.

MO on the AWT metric. Our FECO-TSC achieved consistent lower AWT in an episode with the peak AWT as 2.91s, while TSC-MO, TSC-SO, and FTM did not regulate AWT consistency and had significantly higher peak AWTs with 11.42s, 13.05s, and 30.30s, respectively. The result shows that FECO-TSC not only optimizes the signal policy but also provides better speed profiles to move vehicles smoothly.

Figure 4.9b shows the performance of the AD metric. The peak AD of FECO-TSC was 0.63, which was lower than TSC-SO with 0.74 and TSC-MO with 0.92. It means the average speed of vehicles in peak hours controlled by FECO-TSC was 5.1 m/s compared with 3.6 m/s of TSC-SO and 1.1 m/s of TSC-MO. The results suggest that the vehicles controlled by FECO-TSC moved faster compared to TSC-SO and TSC-MO's. This is because FECO-TSC not only provides a longer green time for the lanes with higher vehicle volumes but also adjusts vehicle speed more moderately than pure TSC methods, in which case more vehicles could pass the intersection with a higher speed.

Figure 4.9c shows the performances of the AQL metric. FECO-TSC achieved the best performance with a near-zero queue length during peak time compared to TSC-SO with 2.6, TSC-MO with 3.2, and FTM with 3.8. We can see that FECO-TSC reduced the number of vehicles waiting at the intersection by moderately adjusting vehicles' deceleration. While under the control of TSC-SO and TSC-MO, vehicles stopped at the intersection with the highest deceleration if the signal was red, suggesting an inefficient adjustment of the queue length.

Table 4.2: The stability comparison of TSC on synthetic scenario.

Methods	**AWT|Std.**	**AD|Std.**	**AQL|Std.**
FTM	12.70|13.77	0.63|0.42	1.36|2.04
TSC-SO	4.81|8.56	0.37|0.42	0.73|1.45
TSC-MO	3.64|7.04	0.42|0.43	0.88|2.43
FECO-TSC	**0.21|1.15**	**0.33|0.38**	**0.02|0.07**

Stability Comparison of TSC

Here, we compared the stability of FECO-TSC and pure DRL-based TSC methods on the synthetic data through the standard deviations (Std.) of AWT, AD, and AQL metrics. Table 4.2 shows the average performance on AWT, AD, and AQL metrics of FTM and TSC methods. Based on the arrival rates shown in Figure 4.6, the vehicle demands increase dramatically in the first 30 minutes and decrease in the second 30 minutes. We can see that FECO-TSC outperformed FTM and other pure DRL-based TSC methods when the arrival rates of vehicles fluctuated in a short period. With the control of FECO-TSC, AWT, AD, and AQL were lower than the other methods, meaning that every vehicle spent less waiting time at the intersection and passed through the intersection with a higher speed, and the waiting queue length of the intersection was shorter. The reason is that a more efficient TSC reduced the red signal time for approaching vehicles, and a better speed profile control increased vehicle speed, and prevented vehicles from stopping at the intersection.

Furthermore, the standard deviations of AWT, AD, and AQL were lower than FTM and pure DRL-based TSC methods, meaning that FECO-TSC improves the ability to stably control the traffic flows to prevent the dramatic increments of AWT, AD, and AQL in a short period. Such improvement reduces the delay of the traffic flows and alleviates congestion at intersections. The reason for the stability improvement is that DRL-MRs provides a framework to trade-off the candidate actions for the traffic states, leading to the optimal action under multiple objectives. Therefore, FECO-TSC has higher stability for smoothing the traffic flows by controlling traffic signals and speed profiles.

4.3.6 Simulation Results on Real-world Scenario

In this section, we compared FECO-TSC with other baseline methods on real-world datasets.

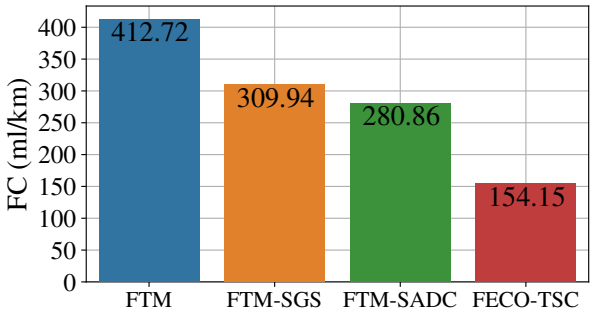
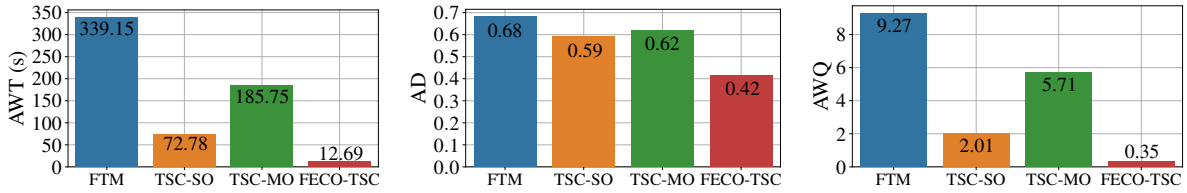


Figure 4.10: Performance on the real-world scenario w.r.t. vehicle fuel consumption (FC) (ml/km).

Overall Analysis

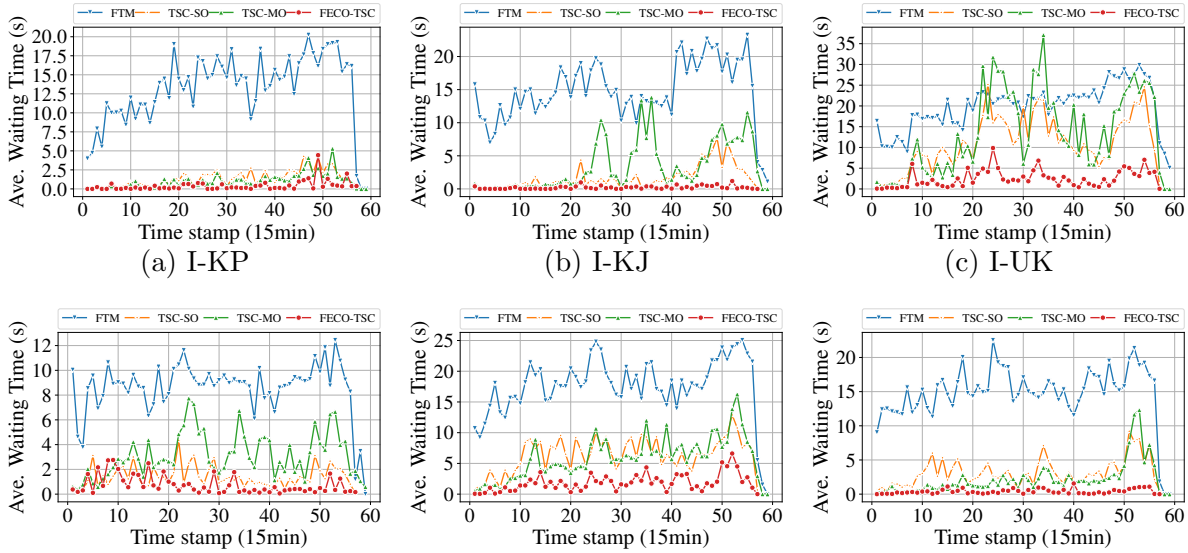
Figure 4.10 shows the performance of FECO-TSC with respect to the fuel consumption on the six successive intersections. The performance in Figure 4.10 is calculated as the average fuel consumption that a vehicle needs to pass through the six intersections. Note that the performance includes the vehicles that left the traffic road network in the middle of six intersections. Compared with FTM and pure eco-driving techniques, FECO-TSC achieved significant vehicle fuel savings and outperformed FTM-SGS and FTM-SADC. The average VFC of FECO-TSC is 154.15 ml/km, which reduced VFC by 62.65% compared to FTM, 50.26% compared to FTM-SGS, and 45.12% compared to FTM-SADC, respectively. The significant improvements benefitted from both the advances of TSC and eco-driving techniques. Based on TSC techniques, FECO-TSC provides an optimal traffic signal timings to assign longer green signal timing to the lanes with higher demands to minimize the vehicle waiting time, waiting queue length, and delays. Vehicles can pass through the intersections with fewer stops than the vehicles controlled by FTM-based methods.

Moreover, FECO-TSC provides the optimal speed profiles from approaching vehicles to prevent speed loss. We know that hard stop-go movements at signalized intersections consume more fuel than smooth movement. Therefore FECO-TSC controls the acceleration/deceleration of vehicles so that vehicles can smoothly slow down or accelerate to



(a) Average Waiting Time (b) Average Delay (c) Average Queue Length

Figure 4.11: Performance on the real-world scenario w.r.t. AWT, AD and AQL.



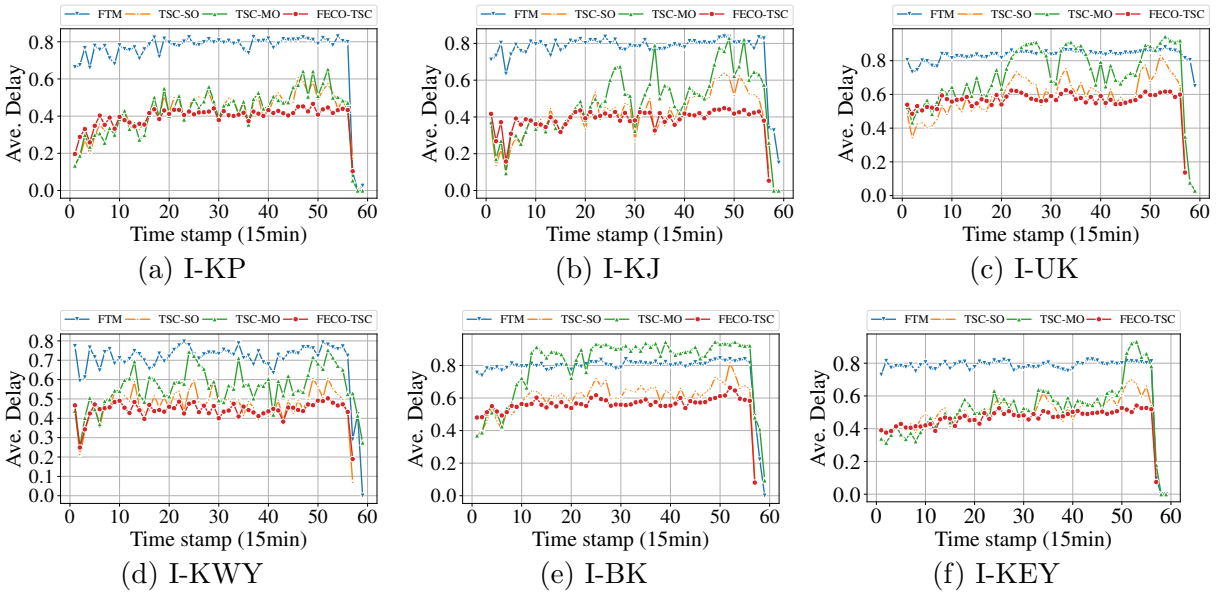
(a) I-KP (b) I-KJ (c) I-UK

(d) I-KWY (e) I-BK (f) I-KEY

Figure 4.12: Performance on real-world scenario w.r.t. average waiting time per 15 min from 6:00 to 20:00 at the six intersections.

move through the intersections instead of multiple instances of hard stop-go movements. Although pure eco-driving methods improved fuel economy to some extent by controlling vehicle speed profiles, these methods can hardly achieve satisfactory performance because they control vehicles' speed based on fixed traffic signal timings of incoming intersections. However, the traffic signal timings controlled by adaptive TSC methods are changing due to dynamic vehicle demands, so pure eco-driving methods can hardly perform as well as the FECO-TSC method.

Figure 4.11 presents the overall performance of FECO-TSC on real-world data compared to FTM and pure DRL-based TSC methods. We can see that FECO-TSC outperformed FTM and pure DRL-based TSC methods on AWT, AD, and AQL metrics. With the advances of the eco-driving technique, FECO-TSC controls the acceleration/deceleration of



(a) I-KP (b) I-KJ (c) I-UK

(d) I-KWY (e) I-BK (f) I-KEY

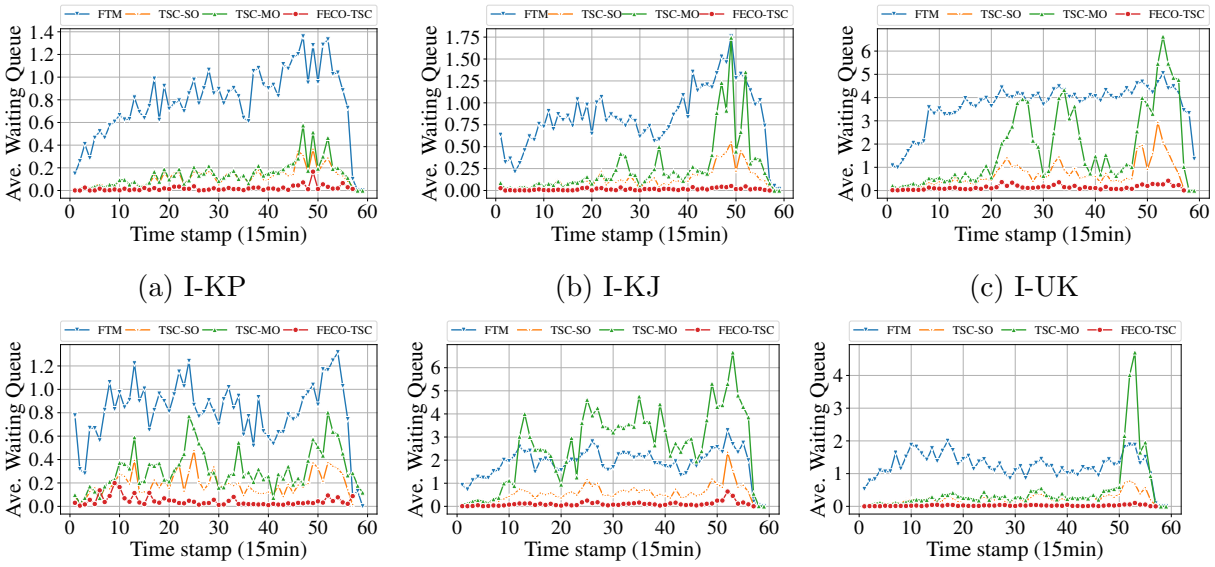
Figure 4.13: Performance on real-world scenario w.r.t. average delay per 15 min from 6:00 to 20:00 at the six intersections.

vehicles to smooth the movements and prevent the unnecessary stops at the intersections. Since more stop times result in not only higher fuel consumption due to acceleration from 0 m/s to the expected speed but also longer waiting time, delays and waiting queue length of intersections, FECO-TSC smooths the speed reduction instead of hard stop so as to keep vehicles moving and through the intersections. On the other hand, pure DRL-based TSC methods can hardly achieve satisfactory performance as FECO-TSC because these methods only control traffic signal timings, not approaching vehicle movements.

Stability Comparison of TSC

Here, we analyze the stability of FECO-TSC and pure DRL-based TSC methods on a real-world dataset.

Figure 4.12 compared the stability performances of FECO-TSC and pure DRL-based TSC methods on the AWT metric. We can see that FECO-TSC improves stability over FTM and pure DRL-based TSC methods with dynamic vehicle demands. The average waiting time of TSC-SO and TSC-MO was fluctuated more during the rush hours compared to FECO-TSC, especially at the intersections with higher volume demand, such as the performance shown in Figure 4.12c. The improvement was benefited from the framework



(a) I-KP (b) I-KJ (c) I-UK

(d) I-KWY (e) I-BK (f) I-KEY

Figure 4.14: Performance on real-world scenario w.r.t. average waiting queue per 15 min from 6:00 to 20:00 at the six intersections.

DRL-MRs, which collectively optimized the TSC policy for multiple objectives and traded off the actions by AVP method. Moreover, the efficient speed control approach smoothed vehicular movements and prevented vehicles from stopping at the intersections.

Figure 4.13 compared the stability performances of FECO-TSC and pure DRL-based TSC methods on the AD metric. Based on the definition of delay in Eq. 4.4, lower AD denotes that vehicles can move through the intersections at higher speeds. From the performances in Figure 4.13, we can see that the average speeds of vehicles were higher than FTM and pure DRL-base TSC methods on six intersections, and also were stably controlled in higher values when the vehicle volumes were increased during the rush hours. The reason is that vehicles controlled by TSC-SO and TSC-MO had longer waiting time at intersections, leading to longer instances of 0 km/h speed. Meanwhile, FECO-TSC efficiently controlled vehicle speed profiles to keep vehicle speeds as high as possible. It prevented speed loss and improved the average speed of moving vehicles.

Figure 4.14 compared the stability performances of FECO-TSC and pure DRL-based TSC methods on the AQL metric. Lower AQL denotes fewer vehicles waiting at the intersections, meaning that the intersections have a lower probability of being congested. According to Figure 4.14, we can see that FECO-TSC stably controlled the waiting queue

lengths of intersections in lower values compared to other methods. While the AQL of TSC-SO and TSC-MO fluctuated more during the test period and had a more sensitive response to the dynamic vehicle volumes; for example, the AQL at the intersection I-UK of the TSC-MO method was increased remarkably due to the high vehicle volumes from 10:00 to 14:00 and 17:00 to 20:00. The figure demonstrates that FECO-TSC has a higher ability to keep vehicles moving and prevent vehicles from stopping at the intersections, reducing the probability of stops and waiting at intersections.

Based on the performances on the synthetic and real-word multi-intersection traffic scenarios, we can see that the proposed method can be easily adopted in both an isolated intersection and multiple intersections.

4.4 Summary

In this chapter, an efficient DRL-based traffic signal control scheme is described to improve the efficiency of traffic management for an isolated intersection. The proposed method describes the design of the agent in the formulation of traffic signal control problem by DRL algorithms, as well as two strategies for controlling traffic signals and on-road vehicle speed profiles. The agent design explains the designs of traffic state representation, actions and reward models for formulating the traffic signal control problem by DRL algorithms. A multi-objective deep Q-networks model is presented in this chapter to help the agent stably regulate traffic flows under multiple criteria by controlling traffic signals. Moreover, a speed profile control strategy is proposed based on the traffic signal control strategy to further smooth the traffic flows.

Chapter 5

Cooperative Traffic Signal Control for Multi-intersections

5.1 Introduction

In Chapter 4, we have demonstrated that the proposed method not only significantly improves the efficiency of TSC for an isolated intersection but also smooths traffic flows and improves vehicle fuel efficiency. In the more realistic traffic environment, however, the traffic flows of an isolated intersection depend on the traffic conditions of its adjacent intersections and itself [99, 156–159]. Therefore, cooperative endeavours are requested from those associated intersections. Cooperative traffic signal control of multiple intersections helps better regulate traffic flows in connected intersections, improve the efficiency of traffic management, and even prevent traffic congestion problems. The achievement of cooperative traffic signal control requires the ultra-low-latency traffic information on on-road vehicles and adjacent intersections [160–164].

Many existing RL-based cooperative TSC methods try to improve the efficiency of signal control based on real-time joint traffic information [21] or joint Q-functions [22] in an ideal traffic environment. However, they expose several limitations. First, joint traffic information and Q-functions introduce high dimensions of traffic states and Q-values, increasing the learning complexity of RL methods on a large-scale traffic road network. Second, the ideal traffic environment does not consider data transmission delays

in vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication [165–169]. However, vehicular networks in real traffic environments are incapable of providing real-time traffic information without data transmission delays [170–172]. Such transmission delay affects the real-time traffic information collection and the cooperative traffic signal control over a vehicular network[172–176].

In this chapter, we propose an RL-based cooperative TSC scheme with the following contributions:

1) A new traffic state representation method was designed based on queue length-based max-pressure measurements [177–179] that considers the traffic states of both local and adjacent intersections. The proposed representation lets the signal controllers know the upcoming traffic flows from adjacent intersections in the future steps.

2) We improved the reward model used in [45] by considering the capacity of connecting lanes and upcoming vehicles from adjacent intersections. The improved reward model avoids the traffic congestion in the case of letting vehicles move into fully occupied lanes and reduces the frequency of switching traffic signals by considering the upcoming vehicle demands instead of the local traffic condition.

3) A new method was introduced to address the data transmission delay from approaching vehicles to intersections to enhance the state measurements for the real-time traffic conditions. Due to the data transmission delay in vehicular networks [180–183], we conducted comprehensive experiments with different ranges of data transmission delays to demonstrate the effectiveness of the method in the traffic environment.

5.2 RL-based Cooperative Traffic Signal Control Method

5.2.1 Architecture

Based on RL's architecture introduced in [20], the architecture of the proposed MP-CTSC is mainly composed of a traffic environment and an agent. The traffic environment is a traffic road network containing multiple connected intersections. The agent learns the collected traffic information and trains a function approximation model to find the optimal signal control action for the given traffic states. Due to the data transmission delay issue in

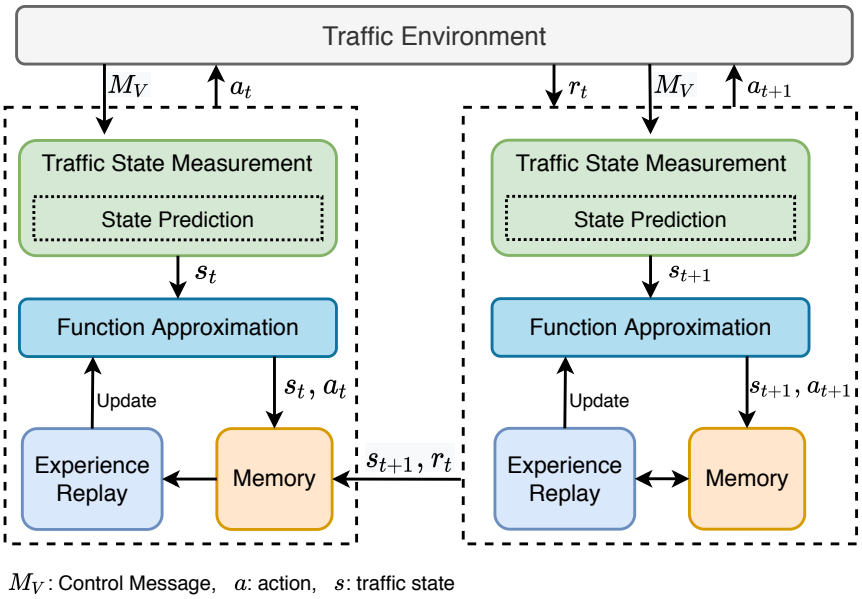


Figure 5.1: The architecture of the proposed MP-CTSC.

vehicular networks [184–188], the agent has a traffic state measurement module to predict the real-time traffic conditions based on the delayed traffic information.

Fig. 5.1 shows the interactions between the traffic environment and the agent at time step t. The agent listens continuously incoming control messages M_V from the traffic environment and transforms the messages to traffic states. Due to the delay for forwarding control message M_V, the traffic state measurement module predicts the traffic state, s_t, at time step t to reduce the discrepancy between the real-time and delayed traffic conditions. Given the predicted traffic state, the function approximation module generates the optimal action (a_t) for controlling the signals. The reward (r_t) for the executed action (a_t) and the next traffic state (s_{t+1}) are returned to the agent at time step $t+1$ and stored in memory with the s_t and a_t. The agent's experience is stored in the memory for improving the function approximation model through the experience replay process.

The following sections describe the agent design, function approximation module, and traffic state measurement module.

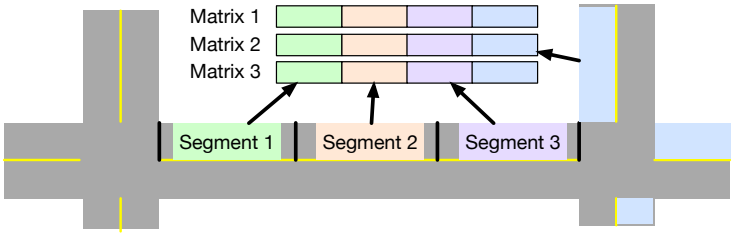


Figure 5.2: Traffic state representation.

5.2.2 Agent Design

Traffic state representation

In this thesis work, we use four traffic features to describe the traffic states of an arbitrary intersection. These four features are local current traffic signal phase, vehicle volumes, speed, and acceleration. The current traffic signal phase is the index of the current phase. Vehicle volumes, speed, and acceleration are represented by three matrices M_1, M_2, M_3, respectively. Each matrix has dimension $|L|\times 4$, where L denotes the connecting lanes of the intersection, and $|L|$ is the number of the lanes. In these matrices, the row vectors represent the feature states of the vehicles moving in the corresponding lanes. As shown in Fig. 5.2, each lane is evenly divided into three segments in order to reduce the dimensions of the state representation. Cell 1, cell 2, and cell 3 of the rows describe the features of corresponding segments on the connecting lanes, and cell 4 indicates the states of approaching vehicles from adjacent intersections. For example in Fig. 5.2, cell 1, cell 2, and cell 3 in M_1, M_2, and M_3 denote the the number of vehicles, the average vehicle speed, and the average acceleration of the vehicles in the lane segment 1, segment 2, and segment 3, respectively. Cell 4 in M_1, M_2, and M_3 describes the three features of approaching vehicles from the right adjacent intersection.

Action

Action space defines the actions that the agent can take to impact the environment [20]. Our action space is defined as $\mathbf{A} = \{a_1, a_2\}$ to represent two left-turn permitted TSPs, respectively. The action space $\mathbf{A}$ can also be defined as $\{a_1, a_2, a_3, a_4\}$ if the TSP design is left-turn protected TSPs. We define a minimum duration of a TSP (Δt) to avoid flickering TSP due to dynamic traffic states. The TSP can not be switched to another TSP until the

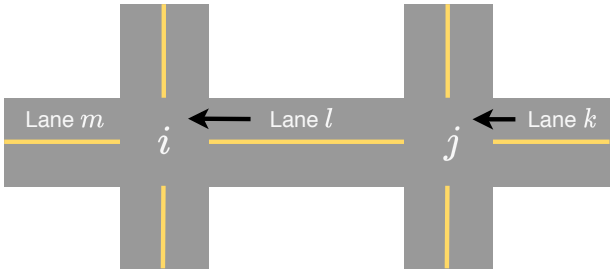


Figure 5.3: Pressure movements between two intersections.

elapsed time of the current TSP is higher than Δt. A five-second yellow TSP is applied between two different TSPs. When a newly selected TSP is different from the current one, the agent switches to the five-second yellow phase before updating the current phase to the target TSP.

Reward

Reward frames the application's objectives and assesses the executed action for improving the agent's performance in future steps [20]. Here, we design a reward model based on MP that measures the pressure of vehicular movements at an intersection based on incoming and outgoing queue length measurements.

We assume that a traffic environment contains an intersection i and an adjacent intersection $j \in NB_i$, where the NB_i denotes the collection of neighbor intersections of the intersection i (see Figure 5.3). Then, we define the pressure of a vehicular movement from incoming lane l to outgoing lane m at time step t as follows:

$$P_t(l, m) = q_t(l, m) - c_t(m) + \sum_{k \in L_j} s_{j,t}(k, l) \cdot q_t(k, l), \tag{5.1}$$

where $q_t(l, m)$ denotes the number of vehicles moving from lane l to lane m, L_j denotes the incoming lanes of adjacent intersection j. $s_{j,t}(k, l) = \{1, 0\}$ is a binary variable denoting the traffic signals of the vehicular movements from lane k to lane l. The value 1 means that the signal is green for the vehicular movement (k, l) at time step t; otherwise, it returns 0. $c_t(m)$ denotes the vacancy of exiting lane m that is available for incoming vehicles at time step t. The values of $c_t(m)$ describe the capability of reducing the intersection's pressure. When $c_t(m) = 0$, the lane m is full of vehicles and no space for feeding vehicles

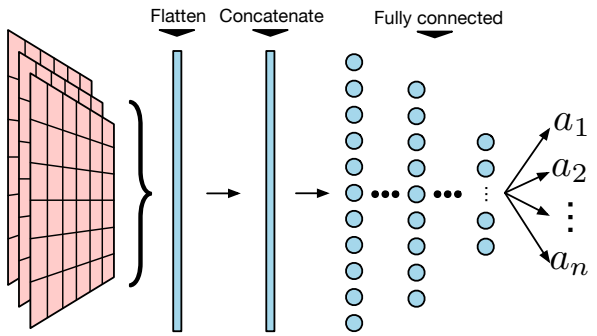


Figure 5.4: The architecture of Q-Network.

from upstream lanes. $c_t(m) > 0$ denotes the number of vehicles that can go through the intersection and enter into lane m.

Based on the pressure measurements described above, the pressure of intersection i at time step t is defined as

$$P_t = \sum_{l,m \in L_i} P_t(l, m). \tag{5.2}$$

The objective of the traffic signal controller is to minimize the total pressure of the intersection. Therefore, we define the reward r_t at time step t as follows:

$$r_t = -P_t. \tag{5.3}$$

5.2.3 Function Approximation

In this work, we adopt Q-network as the function approximation to estimate Q-values of actions for the given traffic states. As shown in Figure 5.4, the four features (one scalar, three matrics) of a given traffic state are flattened and concatenated into an array before passing through a neural network. The neural network consists of several fully connected layers and an output layer. In practical applications, the number of hidden layers of the network can be adjusted according to the actual conditions of different intersections. The output values of neurons in the output layer represent the Q-values of actions. The action with a higher Q-value is selected by the agent to control traffic signals. ϵ-greedy algorithm is used in action selection. The agent randomly selects an action with a small probability of ϵ, and generally selects the optimal action with the highest Q-value since the Q-value

quantifies the agent's rewards in the future steps.

To train the TSC model, we adopt experience replay techniques [151, 152] to update model's parameters θ. An experience cache is maintained in memory to store the encountered traffic states and executed actions and the corresponding rewards from the traffic environment. The cache is represented as a sequence queue structure with a maximum size and automatically removes the least recently inserted experience. To stabilize the learning progress, a target Q-network [151] is applied in our traffic signal model. Although similar to the structure of the learning Q-network, the target Q-network updates parameters θ^- based on θ with a lower frequency.

5.2.4 Traffic State Measurement

Date transmission delay is inevitable in wireless communications [189–194], especially in vehicular networks due to dynamic traffic environments [49, 195–198]. Various factors can cause the delay of messages sent by a vehicle to an intersection, such as the vehicle speed, the distances between the vehicle to the intersection or the front vehicles. Therefore, the traffic states collected by the controller agent at every time step can not represent the real-time traffic conditions. Here, we design a traffic state measurement to address the delay in vehicular networks.

Contents of the Control Message

In our traffic state measurement, the content of a control message must capture the necessary information needed in our later state prediction. Accordingly, in this work, we designed a new control message consisted of five fields representing the instantaneous vehicle data transmitting from a vehicle to an intersection agent (see Table 5.1). The first field is the timestamp t_{sent} of the message sent from the vehicle that creates the message. The timestamp field is used by the agent to estimate the delay of the message transmission. The second field is the globally unique id of the approaching vehicle. The third field is the position (p_{sent}) of the vehicle at timestamp t_{sent}. We assume that the position data can be fetched by the vehicle through GPS devices. To simplify the method's description, we define the vehicle position by the distance between the vehicle and the approaching inter-

Table 5.1: The field definitions of a control message content.

Field	Description
t_{sent}	The timestamp of a message sent by a vehicle
id	Global unique id of a vehicle
p_{sent}	Vehicle position at timestamp t_{sent}
s_{sent}	Vehicle speed at timestamp t_{sent}
a_{sent}	Vehicle acceleration at timestamp t_{sent}

section. The forth and fifth fields are the speed profile information of the vehicle including the vehicle speed (s_{sent}) and acceleration (a_{sent}) at timestamp t_{sent}.

State Prediction

To reduce the adverse effect of data transmission delay on the control efficiency of the TSC system, we propose a way to predict the real-time traffic state based on the received messages of upcoming vehicles from adjacent intersections.

Assuming that at time step t, an agent has received a sequence of messages M_V of upcoming vehicles V from adjacent intersection j in the previous control cycle. The delay of the message M_{V_i} from vehicle V_i to the intersection can be calculated as follows:

$$d(V_i) = t - M_{V_i}(t_{\text{sent}}). \tag{5.4}$$

Since the speed of vehicle V_i is changing dynamically, the control agent does not know the instantaneous speed of the vehicle V_i during the delay period $d(V_i)$. Hence, we assume that the received vehicle speed s_{sent} is the average speed of the vehicle V_i in the delay period. Based on the speed assumption, the expected moving distance of V_i can be estimated as follows:

$$E(dis(V_i)) = d(V_i) \times M_{V_i}(s_{\text{sent}}) \tag{5.5}$$

Here, we analyze the expected position of the vehicle V_i in two cases:

Case 1: We assume that V_0 is the leading vehicle on its lane. The expected position $E(p_{V_0})$ of the vehicle V_0 is calculated based on the expected moving distance and the signal

status of the adjacent intersection j as follows:

$$E(p_{V_0}) = \begin{cases} M_{V_0,l}(p_{\text{sent}}) - E(dis(V_0)), & \text{if signal} = \text{G} \\ \max\{0, M_{V_0,l}(p_{\text{sent}}) - E(dis(V_0))\}, & \text{otherwise} \end{cases} \tag{5.6}$$

When $E(p_{V_0}) < 0$ and the signal is green, the vehicle V_0 has already left the adjacent intersection j and moved into the incoming lane of the intersection i. In this case, we must re-estimate $E(p_{V_0})$ as V_0 is currently on the lane of the intersection i. Since we present the position of a vehicle by the distance between the vehicle and the intersection, we can recalculate the expected position of V_0 using $E(p_{V_0}) = len - |E(p_{V_0})|$, where len is the length of the lane at the intersection i.

Case 2: V_i is not the leading vehicle ($i \neq 0$). The expected position of the vehicle V_i is calculated by the expected moving distance and constrained by the expected position of the front vehicle V_{i-1}. By assuming the vehicle is not allowed to change the lane and overtake front vehicles, we can calculate the expected position of the vehicle V_i as follows:

$$E(p_{V_i}) = \max\{M_{V_i}(p_{\text{sent}}) - E(dis(V_i)), E(p_{V_{i-1}}) + Gap\}, \tag{5.7}$$

where Gap denotes the length of a vehicle plus the minimum gap between two vehicles.

5.3 Experiments and Results

Based on the experimental settings in Chapter 4, we further conducted the experiments to evaluate the performance of our max-pressure-based cooperative traffic signal control (MP-CTSC) in the traffic environment with multi-intersections. The measurements we used to evaluate the performance of our method fall into two categories: intersection-based and vehicle-based. The intersection-based measurements include queuing time and queue length at intersections. The vehicle-based measurements include vehicle waiting time, average vehicle speed and the number of stop times, and vehicle fuel consumption.

5.3.1 Traffic Environment

To evaluate the performance of our MP-CTSC method more accurately, we tested our work by considering two distinctive traffic scenarios.

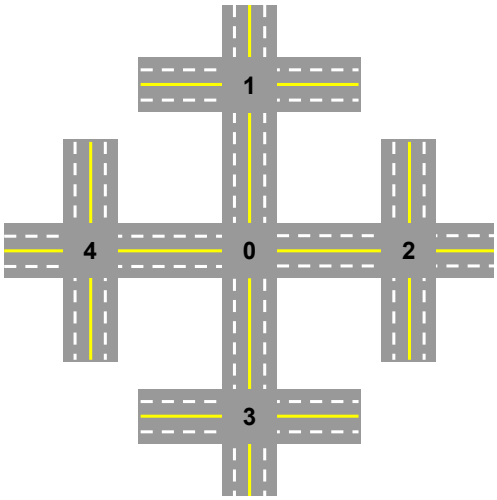


Figure 5.5: Traffic road network structure in Traffic Scenario 1.

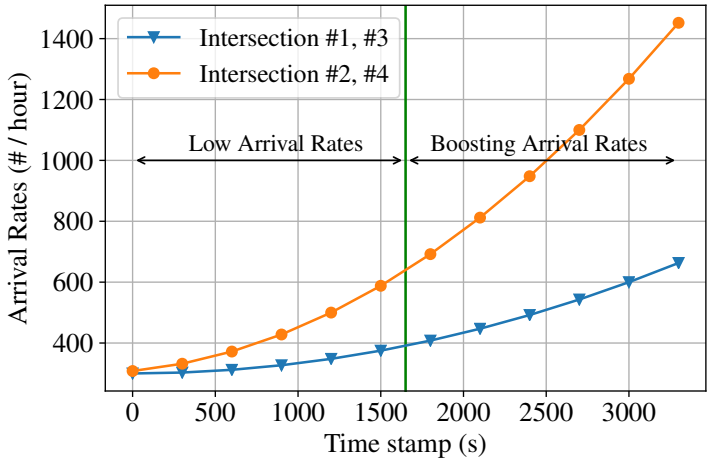


Figure 5.6: The vehicle arrival rates in Traffic Scenario 1.

Traffic Scenario 1

Based on the synthetic scenario in Chapter 4, we used SUMO to extend the traffic environment to an intersection connecting with four adjacent intersections (see Figure 5.5). The traffic signals of intersection No. 0 was controlled by MP-CTSC, and the rest intersections were controlled by pre-timed traffic signal phases. The pre-timed TSPs follows the left-turn permitted design in Figure 3.2. We set 60 seconds for each phase and 5 seconds for the yellow signal phase.

Figure 5.6 shows the arrival rates of vehicles entering the traffic environment in an hour. Unbalanced and dynamic vehicle demands were used to imitate real-world traffic

conditions. We simulated the vehicle demands from intersection No. 1 and 3 lower than the demands from intersection No. 2 and 4. Furthermore, the demands from intersection No. 2 and 4 were rising faster than the demands of other intersections. We increased the demands gently in the first half-hour and then boosted the demands of intersection No. 2 and 4 in the second half-hour to evaluate the performance of methods on the boosting vehicle demands. The vehicle routes were generated by the JTRROUTER tool provided by SUMO with the turning ratios of 35% for right-turn movements, 40% go-through movements, and 25% for left-turn movements.

In this scenario, in order to show the effectiveness of addressing data transmission delay issues by the proposed MP-CTSC, we estimated the average message transmission delay from vehicle V_i to an intersection as $\Delta D_i = k \times \delta$, where k denotes the expected number of hops that a message can be successfully transmitted to the intersection, δ denotes the average delay of one-hop [199]. Since our study focuses on the urban traffic road network environment, we ignored the data transmission delay in sparse traffic conditions. We set the transmission range of V2V and V2I communication as 50 meters in our test environment and the delay factor $\delta = \{0\text{s}, 0.5\text{s}, 1\text{s}, 2\text{s}\}$ to evaluate the performance on different ranges of delay.

Traffic Scenario 2

Traffic scenario 2 is a real-world traffic environment defined in Section 4.3.1. As shown in Figure 4.7, the traffic signals of the six intersections signed by circles were controlled by the proposed MP-CTSC method. Pre-fixed traffic signal timing plans controlled the traffic flows of the rest of intersections signed by squares with 60 seconds for each phase and 5 seconds for the yellow signal phase.

5.3.2 Compared Methods

We conducted performance comparisons with the following methods.

1) Pre-timed Traffic Signal Control: Pre-timed TSC is a natural way to control traffic signals with manually designed traffic signal phases. Here, we set 60 seconds for each green TSP and 5 seconds for the yellow warning phase.

2) Max-pressure Traffic Signal Control: Max-pressure TSC [177] is an actuated control method that greedily selects a TSP based on the real-time pressure of vehicular movements. The pressures of vehicular movements are measured by incoming queue lengths and exiting queue lengths without considering the capacity of lanes.

3) Travel Time-based Max-pressure Traffic Signal Control: TTMax-pressure TSC [179] is an actuated control method that greedily selects a TSP based on the travel time of the incoming and outgoing lanes.

4) Presslight Traffic Signal Control: Presslight [45] is a RL-based TSC method that uses scaled max-pressure algorithm [200] to design their reward model in learning progress. The pressures of vehicular movements are measured by the difference between the occupancy of incoming and exiting lanes.

5.3.3 Parameter Settings

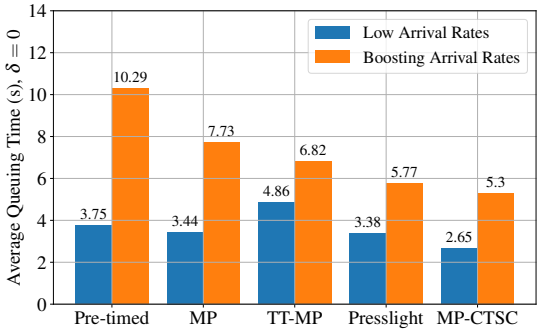
We reused the parameters presented in Table 4.1 to train the q-network. We trained our q-network based on the sampled experience per control cycle and updated the parameters of our target q-network every 300 control cycles.

Based on the q-network architecture shown in Figure 4.3, the input traffic states are firstly fed in flatten and concatenated layers and then two fully connected layers. The first fully-connected layer consists of 128 Leaky ReLU units. The second fully-connected layer consists of 32 Leaky ReLU units. The output layer is a fully-connected layer with two outputs for two actions, each representing the corresponding traffic signal phase.

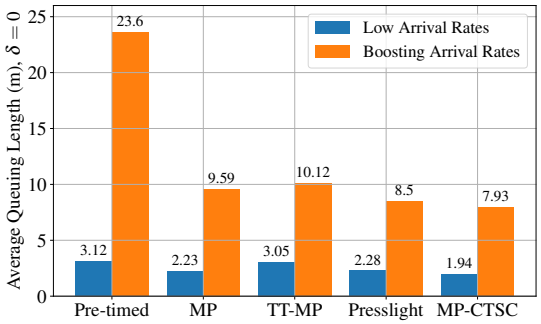
5.3.4 Simulation Results on Traffic Scenario 1

Performance on Intersection-based Measurements

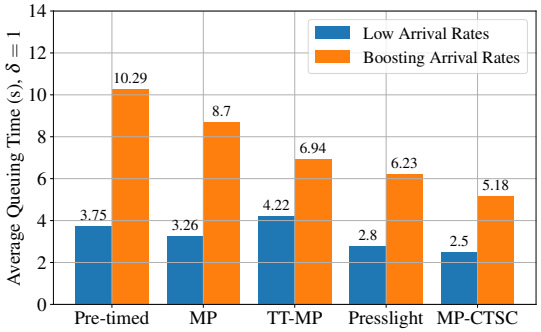
Figure 5.7 shows the performances on the synthetic traffic scenario using the intersection-based measurements of average queuing time (AQT) in seconds and average queuing length (AQL) in meters at intersection No. 0. According to the vehicle arrival rates shown in Figure 5.6, we compared the performance on the low changing arrival rates and boosting arrival rates.



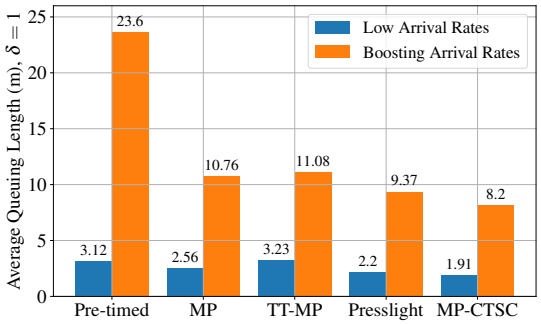
(a) Performances on average queuing time (AQT) metric without data transmission delay.



(b) Performances on average queuing length (AQL) metric without data transmission delay.



(c) Performances on average queuing time (AQT) metric with data transmission delay.



(d) Performances on average queuing length (AQL) metric with data transmission delay.

Figure 5.7: Performances of the intersection-based measurements at the intersection No. 0 in Traffic Scenario 1.

Figure 5.7a and Figure 5.7b are results of AQT and AQL metrics, respectively, without data transmission delay, meaning that the agent can receive instantaneous vehicle data immediately. We can see that all compared methods had similar performances on AQT and AQL metrics in the period with low arrival rates. Actuated and RL-based TSC methods only contributed to slightly better results compared to pre-timed TSC.

When the arrival rates were boosted, the performances of the methods demonstrated significant differences. With the increasing vehicle demands in the second half-hour, MP had the worst performance than others on the measurement of AQT. It was because that the objective of MP is to maximize the throughput of the intersection, which assigns the green signals to the lanes with higher pressures, leading to the higher frequency of switching TSPs in the case when the number of queuing vehicles is changing from different lanes. TT-MP performed better than MP on the AQT metric due to the travel time-based

pressure measurements. Presslight improved the performance of MP on AQT metric by using RL techniques to achieve adaptive TSC for long-term traffic conditions. It reduced the frequency of switching TSPs compared to the actuated TSC methods. MP-CTSC had the best performance on both AQT and AQL metrics compared to other methods. Based on the same advances of RL techniques applied in Presslight method, MP-CTSC considered the traffic pressures of adjacent intersections. Such consideration assigned longer green time to the lanes with higher pressures, including the local pressures and the incoming pressures from adjacent intersections, allowing the agent to keep green signals to the lanes with lower local pressures but higher incoming pressures. While in the Presslight without considering the adjacent pressures, the agent switches green signals to the lanes with higher local pressures, which may be different to the current TSP, and then switches back until the incoming pressures from adjacent intersections move to the local pressures. More switching TSP times cause longer yellow signal duration, resulting in lower efficiency of TSC.

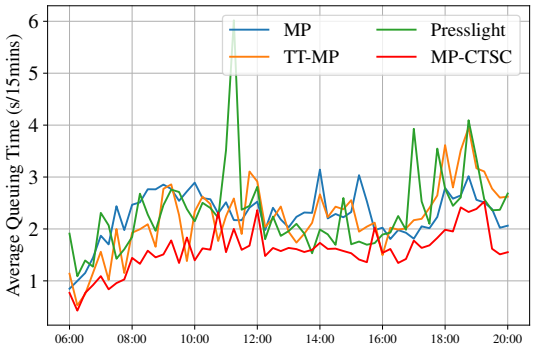
Figure 5.7c and Figure 5.7d are results of AQT and AQL metrics respectively with data transmission delay. We set $\delta = 1$ to ensure that the agent can only receive the instantaneous vehicle data within one hop, meaning that the agent can not receive the vehicle data with a delay of more than one second. Pre-timed traffic signal control had the same performance showed in Figure 5.7a and Figure 5.7b because it does not detect the traffic conditions. MP, TT-MP, and Presslight performed worse in the traffic environment with data transmission delay compared to $\delta = 0$. It is because that the agent was unable to detect the vehicles that had long distances to the intersection due to data transmission delay, resulting in lower pressures of the lanes that contained the undetected vehicles. Such delayed pressure measurements misguided the agent to control the traffic signals, leading to worse performances compared to the scenarios without data transmission delay [167, 201–203]. We can see that MP-CTSC kept the same performances on both traffic conditions owing to the traffic state measurements with prediction. The agent reused the delayed vehicle data to predict the traffic states at the current timestamp, reducing the gap between the traffic states with and without data transmission delay.

Table 5.2: Performances on vehicle-based measurements with respect to waiting time (WT), average stops (Avg. Stops), average speed (Avg. Speed), and fuel consumption (FC) at the intersection No. 0 in Traffic Scenario 1.

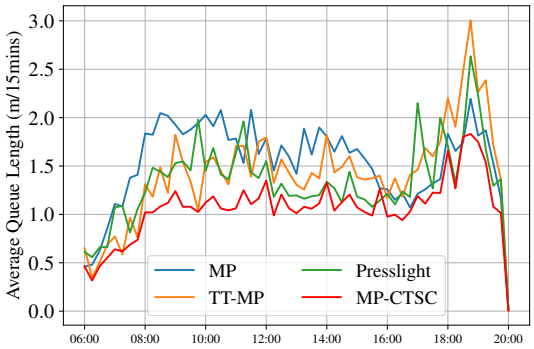
Metric	**Pre-timed**	**MP**		**TT-MP**		**Presslight**		**MP-CTSC**			
Delay δ	-	$\delta = 0$	$\delta = 1$	$\delta = 0$	$\delta = 1$	$\delta = 0$	$\delta = 1$	$\delta = 0$	$\delta = 0.5$	$\delta = 1$	$\delta = 2$
Waiting Time (s)	27.7	14.6	15.4	15.7	16.9	13.1	13.7	12.1	11.9	11.6	12.9
Avg. Speed (m/s)	5.6	6.2	5.6	5.8	5.7	6.2	6.3	6.5	6.6	6.5	6.4
Avg. Stops	1.44	1.01	1.20	1.08	1.05	0.96	0.99	0.94	0.84	0.89	0.92
Fuel Consumption (ml/km)	45.5	29.1	29.9	30.2	31.6	26.9	27.6	26.6	25.8	25.7	27.0

Table 5.3: Performances of vehicle-based measurements in the Traffic scenario 2.

Metric	**Pre-timed**	**MP**	**TT-MP**	**Presslight**	**MP-CTSC**			
Delay δ	-	$\delta = 1$	$\delta = 1$	$\delta = 1$	$\delta = 0$	$\delta = 0.5$	$\delta = 1$	$\delta = 2$
Waiting Time (s)	14.5	10.2	12.5	9.2	7.4	7.5	7.4	7.9
Avg. Speed (m/s)	6.6	6.9	6.8	7.6	7.9	7.6	7.8	7.1
Avg. Stops	0.69	0.67	0.67	0.52	0.47	0.51	0.46	0.51
Fuel Consumption (ml/km)	34.8	29.2	31.6	28.4	26.6	26.9	26.5	28.0



(a) Performances on average queuing time (AQT) metric.



(b) Performances on average queuing length (AQL) metric.

Figure 5.8: Performances of intersection-based measurements at six intersections in Traffic Scenario 2 with data transmission delay $\delta = 1$.

Performance on Vehicle-based Measurements

Table 5.2 summarizes the performances of compared methods on vehicle-based measurements for waiting time (WT) in seconds, stop times, average speed (m/s), and fuel consumption (ml/km) in Traffic Scenario 1. We can see that the pre-timed method resulted in the highest fuel consumption since the high frequency of go-stop movements can cause higher fuel consumption [38]. MP, TT-MP, and Presslight performed slightly worse in the traffic environment with data transmission delay than in the environment without delays. Such results are because of the loss of traffic state measurements due to delayed vehicle data. MP-CTSC outperformed other compared methods on those metrics in both traffic conditions $\delta = 0$ and $\delta \neq 0$. We evaluated MP-CTSC with different delay ranges to demonstrate the effectiveness of our state prediction method. The performance of MP-CTSC was close to each other in all delay conditions, meaning that MP-CTSC with traffic state prediction can effectively address the issue of delayed traffic states.

5.3.5 Simulation Results on Traffic Scenario 2

We further compared the methods in a real-world traffic scenario with a delay factor of $\delta = 1$. Figure 5.8 shows the performance of intersection-based measurements in Traffic Scenario 2, excluding the performance of the pre-timed method because it performed worse than other methods. The figures present the average AQL and AQT per 15 mins of all six

Table 5.4: The average green signal duration (GSD) and yellow signal duration (YSD) at six intersections in Traffic Scenario 2.

Method	**GSD (s)**	**YSD (s)**	**Percentage of YSD**
MP	36195	14509	0.29
TT-MP	36324	14377	0.28
Presslight	37504.6	13321.4	0.26
MP-CTSC	39110	11598	0.23

intersections. We can see that their performances were close to each other, but MP-CTSC performed more stably in the dynamic traffic flows. MP, TT-MP, and Presslight performed more fluctuated than MP-CTSC due to delayed vehicle data. Table 5.3 presents the results of vehicle-based measurements in Traffic Scenario 2. The vehicles had the lowest waiting time, stop times, fuel consumption, and the highest speed in the real-world traffic scenario, meaning that the traffic signal control of MP-CTSC is the most efficient compared with other methods. The reason behind the results is benefited from traffic state measurements with prediction. With the delay factor $\delta = 1$, the agents can only receive the vehicle data within the previous second and miss the data that delayed more than one second. Such missing vehicle data presents a delayed traffic condition at every time step for the agents, leading to a poor decision in the case when the agents assign red signals to the lanes with high pressure. MP-CTSC addresses the data transmission delay issue by traffic state measurements with prediction. It collected traffic vehicle data per second and predicted the current traffic state based on the previous 5-second traffic states. The state prediction reduces the gap between the real-time traffic states and the delayed traffic states.

On the other hand, when considering the pressure of adjacent intersections, our method reduced the frequency of switching TSPs and the duration of yellow signal phases, indicating that MP-CTSC achieved the improvement of the efficiency of the traffic control. Table 5.4 shows that MP-CTSC had the lowest yellow signal duration compared to other methods, meaning that MP-CTSC had fewer switching times during the test. Since the yellow signal phase happened only when the agent attempted to switch the current traffic signal phase to another phase, a higher frequency of switching traffic signal phases can increase the duration of yellow time and reduce the efficiency of the traffic signal control

at intersections because all vehicles must stop during the yellow signal phase.

We also evaluated the performance of MP-CTSC with the different ranges of the delay factor $\delta = \{0, 0.5, 1, 2\}$ to show the effectiveness of the proposed traffic state prediction method in the real-world traffic scenario. From Table 5.3, we can see that the increasing data transmission delay did not seriously worse the performances of MP-CTSC in the real-world traffic scenario. The performance in the traffic condition with delay factor $\delta \leq 1$ had the closest performance with the delay factor $\delta = 0$, and we found that the performance of $\delta = 2$ was worse compared to the performances of lower delay factors. It shows that the traffic state prediction method can thoroughly address the data transmission delay within one second per hop in our traffic scenarios. A longer delay may reduce the accuracy of the prediction and lower the efficiency of the traffic signal control [204–207].

5.4 Summary

In this chapter, we proposed a reinforcement learning-based cooperative traffic signal control scheme to promote the efficiency of cooperative TSC in multiple intersections. Our method considers the traffic conditions of both the local intersection and adjacent intersections. We proposed a max-pressure-based traffic state representation and a reward model to develop an agent that can evaluate a local pressure and the pressure from its adjacent intersections to avoid the unnecessary controls of switching traffic signal phases. Moreover, we proposed a traffic state prediction method to address data transmission delay in vehicular network environments. We evaluated our traffic state prediction method by setting different ranges of delays. The results show that our method achieved good performances on both vehicle-based and intersection-based metrics under boosting vehicle demands compared with previous max-pressure control methods. Also, our state prediction method can significantly reduce the gap between real-time traffic states and delayed traffic states. Our method demonstrated the ability to address the data transmission delay issue in vehicular networks.

Chapter 6

Conclusion and Future Work

Traffic congestion is a growing public concern worldwide, especially in urban areas. Recently, a considerable literature has grown up around the theme of traffic signal control (TSC) and shown the its power to improve the efficiency of traffic flow regulation. An efficient TSC policy significantly improves intersections' capacity and on-road vehicles' fuel efficiency by smoothing traffic flows and reducing vehicles' waiting time or delay. The thesis work, including literature reviews, methodologies and experimental results, has been presented in previous chapters to show the significance of TSC and the effectiveness of the proposed TSC methods. This chapter concludes the main contributions of the thesis work and discusses future research arising from the current work.

6.1 Contribution

In this thesis, we investigated traffic signal control problems for both an isolated intersection and a traffic road network containing multiple intersections. We proposed to employ reinforcement learning techniques to formulate TSC problems in different traffic scenarios and improved the efficiency of traditional RL-based TSC schemes in a more realistic traffic environment. Here, the main contributions of this thesis are summarized as follows:

- Reward designing is a critical part of RL-based TSC methods. Many studies have put effort into designing a scaled reward to achieve multi-objective by combining multiple weighted traffic measurements. However, the single reward value is less sensitive to

each individual's behaviours due to the scaled traffic measurements. In this thesis, we proposed a DRL-MRs framework to enable the agent to learn the optimal TSC solution for various objectives collectively. Meanwhile, an action voting policy was proposed to improve the performance of the action selection.

- Smoothing traffic flows is an essential objective in TSC. Many eco-driving approaches have been proposed to smooth vehicular movements. However, these methods are based on foreseeable traffic signal timing plans. They are inapplicable for the adaptive traffic signal control solutions. In this thesis, we proposed a vehicle speed profile control method to optimize the vehicle's instantaneous speed profiles based on the dynamic traffic timing plans generated by our TSC method.

- The traffic state dimension and the complexity of TSC problems are exponentially increased in the traffic environment with multiple intersections. In this thesis work, we adopted the max-pressure (MP) method [177] in this thesis to design a new efficient traffic state representation and reward model.

- An effective TSC method requires low-latency traffic conditions. However, the data transmission delay issue is unavoidable in a realistic traffic environment. In this thesis, we proposed a state prediction method to reduce the gap between the real-time traffic conditions and the delayed traffic conditions. Our experiments demonstrate that our state prediction method can keep the performance of the agent in different ranges of data transmission delays.

- Synthetic and real-world traffic scenarios were designed and set up in this thesis experimental parts. We evaluated our DRL-MRs framework with the vehicle speed profile control method on both traffic scenarios. The results show that the DRL-MRs TSC scheme can stably regulate traffic flows and improve the efficiency of the TSC under different criteria. Moreover, we evaluated our RL-based cooperative TSC method in a more realistic traffic scenario by adding data transmission delays. The results demonstrate that the proposed method can improve the efficiency of the TSC in a system-wide/global traffic road network and have better performance in the traffic environment with delayed traffic conditions compared to previous TSC methods.

6.2 Future Work

In this thesis, we proposed TSC schemes for both an isolated intersection and multi-intersections. Further research needs to be studied in the next work to apply the proposed method into the real-world traffic environment. This section presents the research directions of the RL-based TSC domain in future work.

Improving State Design in RL-based Traffic Signal Control. More traffic measurements can be added to state designs to enhance the representation of the traffic environment. For example, the feature of the vehicle category was ignored in the previous researches. However, different vehicle categories may impact the TSC solution. Buses, police cars, and ambulances have a higher priority than private cars to obtain green signals. Meanwhile, the traffic state dimension can be exponentially increased due to adding more features into traffic state representation, leading to higher learning cost and computational requirement. Therefore, improving the representation of the traffic state design with less information loss remains a challenge in future research.

Evaluating TSC Methods in a More Realistic Traffic Environment. Many previous TSC studies were evaluated to show the effectiveness in an ideal traffic environment. However, many unforeseeable issues would be emerged in the real-world traffic environment, such as the data transmission delay issue [199, 208]. Therefore, more realistic traffic factors need to be considered in the designs of RL-based TSC schemes. In future work, vehicular networks with road-side units [209, 210] will be investigated to evaluate the data transmission delays between signal controllers and on-road vehicles. Moreover, missing data will be considered because it may occur if some on-road vehicles are unable to communicate with the signal controllers via vehicular networks.

Evaluating TSC Methods in a Large Scale Traffic Road Network. Applying the TSC methods to the real-world traffic environment requires a comprehensive evaluation in a large scale traffic road network. Due to a large number of vehicle volumes and intersections in a traffic road network, the proposed TSC method must have the capability to evaluate the large-scale traffic conditions efficiently and accurately. In our future work, higher traffic demands and more intersections will be designed to evaluate the effectiveness of the proposed TSC methods.

References

[1] Graham Cookson and Bob Pishue. Inrix global traffic scorecard–appendices. *INRIX research*, 2017.

[2] Graham Cookson and Bob Pishue. Inrix global traffic scorecard. *Intelligence That Moves the World. United States: INRIX RESEARCH*, 2018.

[3] Susan Shaheen and Rachel Finson. Intelligent transportation systems. 2013.

[4] Maram Bani Younes and Azzedine Boukerche. A performance evaluation of an efficient traffic congestion detection protocol (ECODE) for intelligent transportation systems. *Ad Hoc Networks*, 24:317–336, 2015.

[5] Michael AP Taylor. Intelligent transport systems. *Handbook of transport systems and traffic control*, 3:461, 2001.

[6] Maram Bani Younes and Azzedine Boukerche. An efficient dynamic traffic light scheduling algorithm considering emergency vehicles for intelligent transportation systems. *Wireless Networks*, 24(7):2451–2463, 2018.

[7] Ahmed Mostefaoui, Mahmoud Melkemi, and Azzedine Boukerche. Localized routing approach to bypass holes in wireless sensor networks. *IEEE Trans. Comput.*, 63(12): 3053–3065, 2013.

[8] Azzedine Boukerche, Jan M Correa, Alba Cristina Magalhaes Melo, and Ricardo P Jacobi. A hardware accelerator for the fast retrieval of dialign biological sequence alignments in linear space. *IEEE Trans. Comput.*, 59(6):808–821, 2010.

[9] Azzedine Boukerche, Sajal K Das, Alessandro Fabbri, and Oktay Yildiz. Exploiting model independence for parallel PCS network simulation. In *Proc. IEEE PADS Workshops*, pages 166–173, 1999.

[10] PB Hunt, DI Robertson, RD Bretherton, and RI Winton. Scoot-a traffic responsive method of coordinating signals. Technical report, 1981.

[11] A. G. Sims and K. W. Dobinson. The sydney coordinated adaptive traffic (scat) system philosophy and benefits. *IEEE Trans. Veh. Technol.*, 29(2):130–137, 1980.

[12] Pitu Mirchandani and Fei-Yue Wang. Rhodes to intelligent transportation systems. *IEEE Intell. Syst.*, 20(1):10–15, 2005.

[13] Thomas Urbanik, Alison Tanaka, Bailey Lozner, Eric Lindstrom, Kevin Lee, Shaun Quayle, Scott Beaird, Shing Tsoi, Paul Ryus, Doug Gettman, et al. *Signal timing manual*, volume 1. Transportation Research Board Washington, DC, 2015.

[14] Azzedine Boukerche and Sotiris Nikoletseas. Protocols for data propagation in wireless sensor networks. In *JWCN*, pages 23–51. Springer, 2004.

[15] Rodolfo WL Coutinho, Azzedine Boukerche, Luiz FM Vieira, and Antonio AF Loureiro. Underwater wireless sensor networks: A new challenge for topology control–based systems. *ACM CSUR*, 51(1):1–36, 2018.

[16] Maram Bani Younes and Azzedine Boukerche. Intelligent traffic light controlling algorithms using vehicular networks. *IEEE Trans. Veh. Technol.*, 65(8):5887–5899, 2015.

[17] Azzedine Boukerche and Yonglin Ren. A trust-based security system for ubiquitous and pervasive computing environments. *Computer Communications*, 31(18):4343–4351, 2008.

[18] Horacio Antonio Braga Fernandes De Oliveira, Azzedine Boukerche, Eduardo Freire Nakamura, and Antonio Alfredo Ferreira Loureiro. An efficient directed localization recursion protocol for wireless sensor networks. *IEEE Trans. Comput.*, 58(5):677–691, 2008.

[19] Azzedine Boukerche, Horacio ABF Oliveira, Eduardo F Nakamura, and Antonio AF Loureiro. Secure localization algorithms for wireless sensor networks. *IEEE Communications Magazine*, 46(4):96–101, 2008.

[20] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction, 2nd ed.* MIT press, 2018.

[21] Samah El-Tantawy, Baher Abdulhai, and Hossam Abdelgawad. Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (MARLIN-ATSC): methodology and large-scale application on downtown toronto. *IEEE Trans. Intell. Transp. Syst.*, 14(3):1140–1150, 2013.

[22] Weirong Liu, Gaorong Qin, Yun He, and Fei Jiang. Distributed cooperative reinforcement learning-based traffic signal control that integrates v2x networks' dynamic clustering. *IEEE Trans. Veh. Technol.*, 66(10):8667–8681, 2017.

[23] Hua Wei, Nan Xu, Huichu Zhang, Guanjie Zheng, Xinshi Zang, Chacha Chen, Weinan Zhang, Yanmin Zhu, Kai Xu, and Zhenhui Li. Colight: Learning network-level cooperation for traffic signal control. In *Proc. ACM CIKM*, pages 1913–1922, 2019. ISBN 978-1-4503-6976-3.

[24] A Boukerche, H.A.B Oliveira, E.F Nakamura, and A.A.F Loureiro. Localization systems for wireless sensor networks. *IEEE Wireless Communications*, 14(6):6–12, 2007. ISSN 1536-1284.

[25] Azzedine Boukerche, Horacio A.B.F Oliveira, Eduardo F Nakamura, and Antonio A.F Loureiro. Vehicular ad hoc networks: A new challenge for localization-based systems. *Computer Communications*, 31(12):2838–2849, 2008.

[26] Azzedine Boukerche, Khalil El-Khatib, Li Xu, and Larry Korba. Sdar: a secure distributed anonymous routing protocol for wireless and mobile ad hoc networks. In *Proc. IEEE LCN*, pages 618–624, 2004.

[27] Azzedine Boukerche, Sungbum Hong, and Tom Jacob. An efficient synchronization scheme of multimedia streams in wireless and mobile systems. *IEEE Trans. Parallel Distrib. Syst.*, 13(9):911–923, 2002.

[28] Stefano Buzzi, I Chih-Lin, Thierry E Klein, H Vincent Poor, Chenyang Yang, and Alessio Zappone. A survey of energy-efficient techniques for 5G networks and challenges ahead. *IEEE J-SAC*, 34(4):697–709, 2016.

[29] Noura Aljeri and Azzedine Boukerche. Mobility management in 5G-enabled vehicular networks: Models, protocols, and classification. *CSUR*, 53(5):1–35, 2020.

[30] Noura AlJeri and Azzedine Boukerche. An efficient movement-based handover prediction scheme for hierarchical mobile IPv6 in VANETs. In *Proc. ACM PE-WASUN*, pages 47–54, 2018.

[31] Azzedine Boukerche, Khalil El-Khatib, Li Xu, and Larry Korba. An efficient secure distributed anonymous routing protocol for mobile and wireless ad hoc networks. *Computer Communications*, 28(10):1193–1203, 2005.

[32] Mourad Elhadef, Azzedine Boukerche, and Hisham Elkadiki. A distributed fault identification protocol for wireless and mobile ad hoc networks. *JPDC*, 68(3):321–335, 2008.

[33] Azzedine Boukerche, Sajal K Das, and Alessandro Fabbri. SWiMNet: a scalable parallel simulation testbed for wireless and mobile networks. *Wireless Networks*, 7 (5):467–486, 2001.

[34] Yonglin Ren and Azzedine Boukerche. Modeling and managing the trust for wireless and mobile ad hoc networks. In *Proc. IEEE ICC*, pages 2129–2133, 2008.

[35] Baraq Ghaleb, Ahmed Y Al-Dubai, Elias Ekonomou, Ayoub Alsarhan, Youssef Nasser, Lewis M Mackenzie, and Azzedine Boukerche. A survey of limitations and enhancements of the IPv6 routing protocol for low-power and lossy networks: A focus on core operations. *IEEE Commun. Surv. Tutor*, 21(2):1607–1635, 2018.

[36] Azzedine Boukerche, Yan Du, Jing Feng, and Richard Pazzi. A reliable synchronous transport protocol for wireless image sensor networks. In *Proc. IEEE ISCC*, pages 1083–1089, 2008.

[37] Noura Aljeri, Kaouther Abrougui, Mohammed Almulla, and Azzedine Boukerche. A performance evaluation of load balancing and QoS-aware gateway discovery protocol for VANETs. In *Proc. IEEE AINA Workshops*, pages 90–94, 2013.

[38] Matthew Barth and Kanok Boriboonsomsin. Real-world carbon dioxide impacts of traffic congestion. *Transp. Res. Rec.*, 2058(1):163–171, 2008.

[39] R. Jurgen. *V2V/V2I Communications for Improved Road Safety and Efficiency*, pages i–viii. 2012.

[40] Tie-Qiao Tang, Zhi-Yan Yi, Jian Zhang, Tao Wang, and Jun-Qiang Leng. A speed guidance strategy for multiple signalized intersections based on car-following model. *Physica A Stat. Mech. Appl.*, 496:399–409, 2018.

[41] Yingrong Lu, Xiaotong Xu, Chuan Ding, and Guangquan Lu. A speed control method at successive signalized intersections under connected vehicles environment. *IEEE Intell. Transp. Syst. Mag.*, 11(3):117–128, 2019.

[42] Noura Aljeri and Azzedine Boukerche. An adaptive traffic-flow based controller deployment scheme for software-defined vehicular networks. In *Proc. ACM MSWiM*, pages 191–198, 2020.

[43] Peng Sun, Noura Aljeri, and Azzedine Boukerche. Machine learning-based models for real-time traffic flow prediction in vehicular networks. *IEEE Network*, 34(3):178–185, 2020.

[44] Patrick Mannion, Jim Duggan, and Enda Howley. An experimental review of reinforcement learning algorithms for adaptive traffic signal control. In *ARTS*, pages 47–66. Springer, 2016.

[45] Hua Wei, Chacha Chen, Guanjie Zheng, Kan Wu, Vikash Gayah, Kai Xu, and Zhenhui Li. Presslight: Learning max pressure control to coordinate traffic signals in arterial network. In *Proc. ACM SIGKDD*, pages 1290–1298, 2019.

[46] Noura Aljeri and Azzedine Boukerche. Performance evaluation of movement prediction techniques for vehicular networks. In *Proc. IEEE ICC*, pages 1–6, 2017.

[47] Noura Aljeri and Azzedine Boukerche. Movement prediction models for vehicular networks: an empirical analysis. *Wireless Networks*, 25(4):1505–1518, 2019.

[48] Azzedine Boukerche, Alexander Magnano, and Noura Aljeri. Mobile IP handover for vehicular networks: Methods, models, and classifications. *ACM CSUR*, 49(4):1–34, 2017.

[49] Noura Aljeri, Mohammed Almulla, and Azzedine Boukerche. An efficient fault detection and diagnosis protocol for vehicular networks. In *Proc. ACM DIVANet*, pages 23–30, 2013.

[50] Yonglin Ren, Richard Werner, Nelem Pazzi, and Azzedine Boukerche. Monitoring patients via a secure and mobile healthcare system. *IEEE Wireless Communications*, 17(1):59–65, 2010.

[51] Osama Abumansoor and Azzedine Boukerche. A secure cooperative approach for nonline-of-sight location verification in VANET. *IEEE Trans. Veh. Technol.*, 61(1): 275–285, 2011.

[52] Peng Sun, Noura AlJeri, and Azzedine Boukerche. A novel proactive handover scheme for achieving energy-efficient vehicular networks. In *Proc. ACM Q2SWinet'18*, page 23–28, 2018.

[53] Wang Yaping and Zhang Zheng. A method of reinforcement learning based automatic traffic signal control. In *Proc. IEEE ICMTMA*, volume 1, pages 119–122, 2011.

[54] Sahar Araghi, Abbas Khosravi, Michael Johnstone, and Doug Creighton. Q-learning method for controlling traffic signal phase time in a single intersection. In *Proc. IEEE ITSC*, pages 1261–1265, 2013.

[55] P. LA and S. Bhatnagar. Reinforcement learning with function approximation for traffic signal control. *IEEE Trans. Intell. Transp. Syst.*, 12(2):412–421, 2011.

[56] Samah El-Tantawy and Baher Abdulhai. Multi-agent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc). In *Proc. IEEE ITSC*, pages 319–326, 2012.

[57] Patrick Mannion, Jim Duggan, and Enda Howley. Parallel reinforcement learning for traffic signal control. *Procedia Computer Science*, 52:956–961, 2015.

[58] MA Wiering. Multi-agent reinforcement learning for traffic light control. In *ICML'2000*, pages 1151–1158, 2000.

[59] Duan Houli, Li Zhiheng, and Zhang Yi. Multiobjective reinforcement learning for traffic signal control using vehicular ad hoc network. *EURASIP J Adv Signal Process*, 2010(1):724035, 2010.

[60] Elise Van der Pol and Frans A Oliehoek. Coordinated deep reinforcement learners for traffic light control. *NIPS'16 Workshop MALIC*, 2016.

[61] Shabestary, Soheil Mohamad Alizadeh, and Baher Abdulhai. Deep learning vs. discrete reinforcement learning for adaptive traffic signal control. In *Proc. IEEE ITSC*, pages 286–293, 2018.

[62] Xiaoyuan Liang, Xunsheng Du, Guiling Wang, and Zhu Han. A deep reinforcement learning network for traffic light cycle control. *IEEE Trans. Veh. Technol.*, 68(2): 1243–1253, 2019.

[63] Wade Genders and Saiedeh Razavi. Using a deep reinforcement learning agent for traffic signal control. *arXiv preprint arXiv:1611.01142*, 2016.

[64] Juntao Gao, Yulong Shen, Jia Liu, Minoru Ito, and Norio Shiratori. Adaptive traffic signal control: Deep reinforcement learning algorithm with experience replay and target network. *arXiv preprint arXiv:1705.02755*, 2017.

[65] Chung-Jae Choe, Seungho Baek, Bongyoung Woon, and Seung-Hyun Kong. Deep Q learning with lstm for traffic light control. In *IEEE APCC*, pages 331–336, 2018.

[66] Hongwei Ge, Yumei Song, Chunguo Wu, Jiankang Ren, and Guozhen Tan. Cooperative deep Q-learning with Q-value transfer for multi-intersection signal control. *IEEE Access*, 7:40797–40809, 2019.

[67] Seyed Sajad Mousavi, Michael Schukat, and Enda Howley. Traffic light control using deep policy-gradient and value-function-based reinforcement learning. *IET Intell. Transp. Syst.*, 11(7):417–423, 2017.

[68] Amir Darehshoorzadeh and Azzedine Boukerche. Underwater sensor networks: A new challenge for opportunistic routing protocols. *IEEE Communications Magazine*, 53(11):98–107, 2015.

[69] Deepeka Garg, Maria Chli, and George Vogiatzis. Deep reinforcement learning for autonomous traffic light control. In *Proc. IEEE ICITE*, pages 214–218, 2018.

[70] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. WieBner. Microscopic traffic simulation using SUMO. In *Proc. IEEE ITSC*, pages 2575–2582, Nov 2018.

[71] Horacio ABF Oliveira, Azzedine Boukerche, Eduardo F Nakamura, and Antonio AF Loureiro. Localization in time and space for wireless sensor networks: An efficient and lightweight algorithm. *Performance Evaluation*, 66(3-5):209–222, 2009.

[72] Robson E De Grande and Azzedine Boukerche. Dynamic balancing of communication and computation load for HLA-based simulations on large-scale distributed systems. *JPDC*, 71(1):40–52, 2011.

[73] Zhenxia Zhang, Richard W Pazzi, and Azzedine Boukerche. A mobility management scheme for wireless mesh networks based on a hybrid routing protocol. *Computer Networks*, 54(4):558–572, 2010.

[74] Kaidi Yang, Isabelle Tan, and Monica Menendez. A reinforcement learning based traffic signal control algorithm in a connected vehicle environment. In *Proc. STRC*. STRC, 2017.

[75] Pang Ha-li and Ding Ke. An intersection signal control method based on deep reinforcement learning. In *Proc. IEEE ICICTA*, pages 344–348, 2017.

[76] Jinghong Zeng, Jianming Hu, and Yi Zhang. Adaptive traffic signal control with deep recurrent q-learning. In *Proc. IEEE IV*, pages 1215–1220, 2018.

[77] Noe Casas. Deep deterministic policy gradient for urban traffic light control. *arXiv preprint arXiv:1703.09035*, 2017.

[78] Junchen Jin and Xiaoliang Ma. A multi-objective agent-based control approach with application in intelligent traffic signal system. *IEEE Trans. Intell. Transp. Syst.*, 2019.

[79] Yaobang Gong, Mohamed Abdel-Aty, Qing Cai, and Md Sharikur Rahman. Decentralized network level adaptive signal control by multi-agent deep reinforcement learning. *TRIP*, 1:100020, 06 2019.

[80] Andrea Vidali, Luca Crociani, Giuseppe Vizzari, and Stefania Bandini. A deep reinforcement learning approach to adaptive traffic lights management. In *WOA*, 2019.

[81] Ingook Jang, Donghun Kim, Donghun Lee, and Youngsung Son. An agent-based simulation modeling with deep reinforcement learning for smart traffic signal control. In *Proc. IEEE ICTC*, pages 1028–1030, 2018.

[82] Li Li, Yisheng Lv, and Fei-Yue Wang. Traffic signal timing via deep reinforcement learning. *IEEE/CAA Journal of Automatica Sinica*, 3(3):247–254, 2016.

[83] Li Congcong, Fei Yan, Yiduo Zhou, Jia Wu, and Xiaomin Wang. A regional traffic signal control strategy with deep reinforcement learning. In *IEEE CCC*, pages 7690–7695, 2018.

[84] Chia-Hao Wan and Ming-Chorng Hwang. Value-based deep reinforcement learning for adaptive isolated intersection signal control. *IET Intell. Trans. Syst.*, 12(9):1005–1010, 2018.

[85] Hali Pang and Weilong Gao. Deep deterministic policy gradient for traffic signal control of single intersection. In *Proc. IEEE CCDC*, pages 5861–5866, 2019.

[86] Yilun Lin, Xingyuan Dai, Li Li, and Fei-Yue Wang. An efficient deep reinforcement learning model for urban traffic control. *arXiv preprint arXiv:1808.01876*, 2018.

[87] Hua Wei, Guanjie Zheng, Huaxiu Yao, and Zhenhui Li. Intellilight: A reinforcement learning approach for intelligent traffic light control. In *Proc. ACM SIGKDD*, pages 2496–2505, 2018.

[88] Ming Xu, Jianping Wu, Ling Huang, Rui Zhou, Tian Wang, and Dongmei Hu. Network-wide traffic signal control based on the discovery of critical nodes and deep reinforcement learning. *J. Intell. Trans. Sys.*, pages 1–10, 2018.

[89] Song Wang, Xu Xie, Kedi Huang, Junjie Zeng, and Zimin Cai. Deep reinforcement learning-based traffic signal control using high-resolution event-based data. *Entropy*, 21(8):744, 2019.

[90] T. Chu, J. Wang, L. Codecà, and Z. Li. Multi-agent deep reinforcement learning for large-scale traffic signal control. *IEEE Trans. Intell. Transp. Syst.*, 21(3):1086–1095, 2020.

[91] Tian Tan, Feng Bao, Yue Deng, Alex Jin, Qionghai Dai, and Jie Wang. Cooperative deep reinforcement learning for large-scale traffic grid signal control. *IEEE Trans. Cybern.*, 2019.

[92] Mohamed A Khamis, Walid Gomaa, and Hisham El-Shishiny. Multi-objective traffic light control system based on bayesian probability interpretation. In *Proc. IEEE Intell. Transp. Syst.*, pages 995–1000, 2012.

[93] Monireh Abdoos, Nasser Mozayani, and Ana LC Bazzan. Traffic light control in non-stationary environments based on multi agent q-learning. In *Proc. IEEE ITSC*, pages 1580–1585, 2011.

[94] Yit Kwong Chin, Nurmin Bolong, Aroland Kiring, Soo Siang Yang, and Kenneth Tze Kin Teo. Q-learning based traffic optimization in management of signal timing plan. *IJSSST*, 12(3):29–35, 2011.

[95] Junchen Jin and Xiaoliang Ma. Adaptive group-based signal control by reinforcement learning. *Transp. Res. Proc.*, 10:207–216, 2015.

[96] Tim Brys, Tong T Pham, and Matthew E Taylor. Distributed learning and multi-objectivity in traffic light control. *Connection Science*, 26(1):65–83, 2014.

[97] Thomas L Thorpe and Charles W Anderson. Traffc light control using sarsa with three state representations. Technical report, Citeseer, 1996.

[98] Junchen Jin and Xiaoliang Ma. Adaptive group-based signal control using reinforcement learning with eligibility traces. In *Proc. IEEE ITSC*, pages 2412–2417, 2015.

[99] Nishant Kheterpal, Kanaad Parvate, Cathy Wu, Aboudy Kreidieh, Eugene Vinitsky, and Alexandre Bayen. Flow: Deep reinforcement learning for control in SUMO. *EPiC Series in Engineering*, 2:134–151, 2018.

[100] Azzedine Boukerche, Richard Werner Nelem Pazzi, and Regina B Araujo. Hpeq a hierarchical periodic, event-driven and query-based wireless sensor network protocol. In *Proc. IEEE LCN*, pages 560–567, 2005.

[101] Richard W Pazzi and Azzedine Boukerche. Propane: A progressive panorama streaming protocol to support interactive 3d virtual environment exploration on graphics-constrained devices. *ACM TOMM*, 11(1):1–22, 2014.

[102] Peng Sun, Noura AlJeri, and Azzedine Boukerche. A fast vehicular traffic flow prediction scheme based on fourier and wavelet analysis. In *Proc. IEEE GLOBECOM*, pages 1–6, 2018.

[103] Jelle R Kok and Nikos Vlassis. Using the max-plus algorithm for multiagent decision making in coordination graphs. In *Robot Soccer World Cup*, pages 1–12. Springer, 2005.

[104] Anahit Martirosyan, Azzedine Boukerche, and Richard Pazzi. A taxonomy of cluster-based routing protocols for wireless sensor networks. In *Proc. IEEE I-SPAN*, pages 247–253, 2008.

[105] Azzedine Boukerche and Damla Turgut. Secure time synchronization protocols for wireless sensor networks. *IEEE Wireless Communications*, 14(5):64–69, 2007.

[106] Lior Kuyer, Shimon Whiteson, Bram Bakker, and Nikos Vlassis. Multiagent reinforcement learning for urban traffic control using coordination graphs. In *ECML PKDD*, pages 656–671. Springer, 2008.

[107] Juan C Medina and Rahim F Benekohal. Traffic signal control using reinforcement learning and the max-plus algorithm as a coordinating strategy. In *Proc. IEEE ITSC*, pages 596–601, 2012.

[108] Frans A Oliehoek, Shimon Whiteson, Matthijs TJ Spaan, et al. Approximate solutions for factored dec-pomdps with many agents. In *AAMAS*, pages 563–570, 2013.

[109] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[110] Itamar Arel, Cong Liu, Tom Urbanik, and Airton G Kohls. Reinforcement learning-based multi-agent system for network traffic signal control. *IET Intell. Transp. Syst.*, 4(2):128–135, 2010.

[111] Tomoki Nishi, Keisuke Otaki, Keiichiro Hayakawa, and Takayoshi Yoshimura. Traffic signal control based on reinforcement learning with graph convolutional neural nets. In *Proc. IEEE ITSC*, pages 877–883, 2018.

[112] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*, 2015.

[113] Gabriel Dulac-Arnold, Richard Evans, Hado van Hasselt, Peter Sunehag, Timothy Lillicrap, Jonathan Hunt, Timothy Mann, Theophane Weber, Thomas Degris, and Ben Coppin. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*, 2015.

[114] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[115] Aleksandar Stevanovic. *Adaptive traffic control systems: domestic and foreign state of practice*. Number Project 20-5 (Topic 40-03). 2010.

[116] Azzedine Boukerche, Cristiano Rezende, and Richard W Pazzi. Improving neighbor localization in vehicular ad hoc networks to avoid overhead from periodic messages. In *Proc. IEEE GLOBECOM*, pages 1–6, 2009.

[117] Fabricio A Silva, Azzedine Boukerche, Thais RM Braga Silva, Linnyer B Ruiz, Eduardo Cerqueira, and Antonio AF Loureiro. Vehicular networks: A new challenge for content-delivery-based applications. *ACM CSUR*, 49(1):1–29, 2016.

[118] Clayson Celes, Fabrício A Silva, Azzedine Boukerche, Rossana Maria de Castro Andrade, and Antonio AF Loureiro. Improving VANET simulation with calibrated vehicular mobility traces. *IEEE Trans. Mobile Comput.*, 16(12):3376–3389, 2017.

[119] Renfei Wang, Cristiano Rezende, Heitor S Ramos, Richard W Pazzi, Azzedine Boukerche, and Antonio AF Loureiro. Liaithon: A location-aware multipath video streaming scheme for urban vehicular networks. In *Proc. IEEE ISCC*, pages 000436–000441, 2012.

[120] Abdelhamid Mammeri, Azzedine Boukerche, and Zongzhi Tang. A real-time lane marking localization, tracking and communication system. *Computer Communications*, 73:132–143, 2016.

[121] Peng Sun, Noura AlJeri, and Azzedine Boukerche. Dacon: A novel traffic prediction and data-highway-assisted content delivery protocol for intelligent vehicular networks. *IEEE Trans. Sustain. Comput.*, 5(4):501–513, 2020.

[122] Azzedine Boukerche and Samer Samarah. An efficient data extraction mechanism for mining association rules from wireless sensor networks. In *Proc. IEEE ICC*, pages 3936–3941, 2007.

[123] Azzedine Boukerche, Nathan J McGraw, Caron Dzermajko, and Kaiyuan Lu. Grid-filtered region-based data distribution management in large-scale distributed simulation systems. In *Proc. IEEE ANSS*, pages 259–266, 2005.

[124] Elie El Ajaltouni, Azzedine Boukerche, and Ming Zhang. An efficient dynamic load balancing scheme for distributed simulations on a grid infrastructure. In *Proc. IEEE/ACM DS-RT*, pages 61–68, 2008.

[125] Azzedine Boukerche and Amber Roy. Dynamic grid-based approach to data distribution management. *JPDC*, 62(3):366–392, 2002.

[126] Mourad Elhadef, Azzedine Boukerche, and Hisham Elkadiki. Performance analysis of a distributed comparison-based self-diagnosis protocol for wireless ad-hoc networks. In *Proc. ACM MSWiM*, pages 165–172, 2006.

[127] Noura Aljeri, Kaouther Abrougui, Mohammed Almulla, and Azzedine Boukerche. A reliable quality of service aware fault tolerant gateway discovery protocol for vehicular networks. *Wirel. Commun. Mob. Comput.*, 15(10):1485–1495, 2015.

[128] Peng Sun, Noura AlJeri, and Azzedine Boukerche. A novel passive road side unit detection scheme in vehicular networks. In *Proc. IEEE GLOBECOM*, pages 1–5, 2017.

[129] Noura Aljeri and Azzedine Boukerche. A probabilistic neural network-based road side unit prediction scheme for autonomous driving. In *Proc. IEEE ICC*, pages 1–6, 2019.

[130] Transportation Research Board of the National Academies. Highway capacity manual 2010. *Washington, DC*, 2010.

[131] Stephen Ezell. Explaining international it application leadership: Intelligent transportation systems. 2010.

[132] Azzedine Boukerche and Sajal K Das. Dynamic load balancing strategies for conservative parallel simulations. In *Proc. IEEE PADS Workshops*, pages 20–28, 1997.

[133] Azzedine Boukerche, Kathia Regina Lemos Jucá, Joao Bosco Sobral, and Mirela Sechi Moretti Annoni Notare. An artificial immune based intrusion detection model for computer and telecommunication systems. *Parallel Computing*, 30(5-6):629–646, 2004.

[134] Cristiano Rezende, Azzedine Boukerche, Heitor S Ramos, and Antonio AF Loureiro. A reactive and scalable unicast solution for video streaming over VANETs. *IEEE Trans. Comput.*, 64(3):614–626, 2014.

[135] Azzedine Boukerche, Noura Aljeri, Kaouther Abrougui, and Yan Wang. Towards a secure hybrid adaptive gateway discovery mechanism for intelligent transportation systems. *Security and Communication Networks*, 9(17):4027–4047, 2016.

[136] Douglas Gettman, Edward Folk, Eddie Curtis, Kent Kacir Dan Ormand, Matthew Mayer, and Erin Flanigan. Measures of effectiveness and validation guidance for adaptive signal control technologies. Technical report, United States. Federal Highway Administration, 2013.

[137] Mario Keller. Handbook of emission factors for road transport (HBEFA) 3.1. Technical report, INFRAS, 2010.

[138] Dongbin Zhao, Yujie Dai, and Zhen Zhang. Computational intelligence in urban traffic signal control: A survey. *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, 42 (4):485–494, 2011.

[139] Alan J Miller. Settings for fixed-cycle traffic signals. *J Oper. Res. Soc.*, 14(4):373–386, 1963.

[140] Azzedine Boukerche, Anis Zarrad, and Regina Araujo. A cross-layer approach-based gnutella for collaborative virtual environments over mobile ad hoc networks. *IEEE Trans. Parallel Distrib. Syst.*, 21(7):911–924, 2009.

[141] Azzedine Boukerche, Richard WN Pazzi, and Jing Feng. An end-to-end virtual environment streaming technique for thin mobile devices over heterogeneous networks. *Computer Communications*, 31(11):2716–2725, 2008.

[142] Azzedine Boukerche and Yonglin Ren. A security management scheme using a novel computational reputation model for wireless and mobile ad hoc networks. In *Proc. ACM PE-WASUN*, pages 88–95, 2008.

[143] Athanasios Bamis, Azzedine Boukerche, Ioannis Chatzigiannakis, and Sotiris Nikoletseas. A mobility aware protocol synthesis for efficient routing in ad hoc mobile networks. *Computer Networks*, 52(1):130–154, 2008.

[144] Peter Koonce and Lee Rodegerdts. Traffic signal timing manual. Technical report, United States. Federal Highway Administration, 2008.

[145] Horacio ABF Oliveira, Eduardo F Nakamura, Antonio AF Loureiro, and Azzedine Boukerche. Error analysis of localization systems for sensor networks. In *Proc. ACM GIS Workshops*, pages 71–78, 2005.

[146] Rodolfo WL Coutinho, Azzedine Boukerche, Luiz FM Vieira, and Antonio AF Loureiro. A novel void node recovery paradigm for long-term underwater sensor networks. *Ad Hoc Networks*, 34:144–156, 2015.

[147] Azzedine Boukerche, Sungbum Hong, and Tom Jacob. A distributed algorithm for dynamic channel allocation. *Mobile Networks and Applications*, 7(2):115–126, 2002.

[148] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

[149] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4): 279–292, 1992.

[150] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[151] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[152] Long-Ji Lin. Reinforcement learning for robots using neural networks. Technical report, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science, 1993.

[153] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[154] Dunhao Zhong and Azzedine Boukerche. Traffic signal control using deep reinforcement learning with multiple resources of rewards. In *Proc. ACM PE-WASUN*, pages 23–28, 2019.

[155] Toronto Transportation Services. King st. transit pilot - detailed traffic & pedestrian volumes, 2018. URL `https://ckan0.cf.opendata.inter.prod-toronto.ca/dataset/5a4e2ef7-8eab-45da-9080-9956a8605229/resource/a5efb524-f062-48e9-84a0-589b12f0754b/download/detailed-traffic-pedestrian-volumes-2018.gz`. Accessed on: May 21st, 2020.

[156] Azzedine Boukerche and Carl Tropper. A distributed graph algorithm for the detection of local cycles and knots. *IEEE Trans. Parallel Distrib. Syst.*, 9(8):748–757, 1998.

[157] Azzedine Boukerche, Amber Roy, and Neville Thomas. Dynamic grid-based multicast group assignment in data distribution management. In *Proc. IEEE DS-RT Workshops*, pages 47–54. IEEE, 2000.

[158] Rodolfo Bezerra Batista, Azzedine Boukerche, and Alba Cristina Magalhaes Alves de Melo. A parallel strategy for biological sequence alignment in restricted memory space. *JPDC*, 68(4):548–561, 2008.

[159] Azzedine Boukerche and Caron Dzermajko. Performance evaluation of data distribution management strategies. *Concurrency and Computation: Practice and Experience*, 16(15):1545–1573, 2004.

[160] Azzedine Boukerche and Xin Fei. A voronoi approach for coverage protocols in wireless sensor networks. In *Proc. IEEE GLOBECOM*, pages 5190–5194, 2007.

[161] Rodolfo WL Coutinho, Azzedine Boukerche, Luiz FM Vieira, and Antonio AF Loureiro. Design guidelines for opportunistic routing in underwater networks. *IEEE Communications Magazine*, 54(2):40–48, 2016.

[162] Samer Samarah, Muhannad Al-Hajri, and Azzedine Boukerche. A predictive energy-efficient technique to support object-tracking sensor networks. *IEEE Trans. Veh. Technol.*, 60(2):656–663, 2010.

[163] Azzedine Boukerche and Xu Li. An agent-based trust and reputation management scheme for wireless sensor networks. In *Proc. IEEE GLOBECOM*, volume 3, pages 5–pp, 2005.

[164] Azzedine Boukerche and Xin Fei. A coverage-preserving scheme for wireless sensor network with irregular sensing range. *Ad Hoc Networks*, 5(8):1303–1316, 2007.

[165] Mourad Elhadef, Azzedine Boukerche, and Hisham Elkadiki. Diagnosing mobile ad-hoc networks: two distributed comparison-based self-diagnosis protocols. In *Proc. ACM MobiWac*, pages 18–27, 2006.

[166] Noura Aljeri and Azzedine Boukerche. A novel online machine learning based RSU prediction scheme for intelligent vehicular networks. In *Proc. IEEE AICCSA*, pages 1–8, 2019.

[167] Richard WN Pazzi and Azzedine Boukerche. Mobile data collector strategy for delay-sensitive applications over wireless sensor networks. *Computer Communications*, 31 (5):1028–1039, 2008.

[168] Azzedine Boukerche. A simulation based study of on-demand routing protocols for ad hoc wireless networks. In *Proc. IEEE ANSS*, pages 85–92, 2001.

[169] Azzedine Boukerche and Samer Samarah. A novel algorithm for mining association rules in wireless ad hoc sensor networks. *IEEE Trans. Parallel Distrib. Syst.*, 19(7): 865–877, 2008.

[170] Azzedine Boukerche, Richard Werner Nelem Pazzi, and Regina Borges Araujo. A fast and reliable protocol for wireless sensor networks in critical conditions monitoring applications. In *Proc. ACM MSWiM*, pages 157–164, 2004.

[171] Azzedine Boukerche, Richard Werner Nelem Pazzi, and Regina Borges Araujo. Fault-tolerant wireless sensor network routing protocols for the supervision of context-aware physical environments. *JPDC*, 66(4):586–599, 2006.

[172] Noura Aljeri and Azzedine Boukerche. Mobility and handoff management in connected vehicular networks. In *Proc. ACM MobiWac*, page 82–88, 2018.

[173] Hadi Habibzadeh, Tolga Soyata, Burak Kantarci, Azzedine Boukerche, and Cem Kaptan. Sensing, communication and security planes: A new challenge for a smart city system design. *Computer Networks*, 144:163–200, 2018.

[174] Abdul Jabbar Siddiqui, Abdelhamid Mammeri, and Azzedine Boukerche. Real-time vehicle make and model recognition based on a bag of SURF features. *IEEE Trans. Intell. Transp. Syst.*, 17(11):3205–3219, 2016.

[175] Cristiano Rezende, Abdelhamid Mammeri, Azzedine Boukerche, and Antonio AF Loureiro. A receiver-based video dissemination solution for vehicular networks with content transmissions decoupled from relay node selection. *Ad Hoc Networks*, 17: 1–17, 2014.

[176] Azzedine Boukerche, Ioannis Chatzigiannakis, and Sotiris Nikoletseas. A new energy efficient and fault-tolerant protocol for data propagation in smart dust networks using varying transmission range. *Computer Communications*, 29(4):477–489, 2006.

[177] Pravin Varaiya. Max pressure control of a network of signalized intersections. *Transp. Res. Part C: Emerging Technologies*, 36:177–195, 2013.

[178] Jennie Lioris, Alex Kurzhanskiy, and Pravin Varaiya. Adaptive max pressure control of network of signalized intersections. *IFAC-PapersOnLine*, 49(22):19–24, 2016.

[179] Pedro Mercader, Wasim Uwayid, and Jack Haddad. Max-pressure traffic controller based on travel times: An experimental analysis. *Transp. Res. Part C: Emerging Technologies*, 110:275–290, 2020.

[180] Azzedine Boukerche, Anahit Martirosyan, and Richard Pazzi. An inter-cluster communication based energy aware and fault tolerant protocol for wireless sensor networks. *Mobile Networks and Applications*, 13(6):614–626, 2008.

[181] Noura Aljeri and Azzedine Boukerche. An optimized link duration-based mobility management scheme for connected vehicular networks. In *Proc. ACM PE-WASUN*, pages 7–14, 2019.

[182] Noura Aljeri and Azzedine Boukerche. A predictive collision detection protocol using vehicular networks. In *Proc. IEEE PIMRC*, pages 1–5, 2017.

[183] Noura Aljeri and Azzedine Boukerche. A performance evaluation of time-series mobility prediction for connected vehicular networks. In *Proc. ACM MSWiM*, pages 127–131, 2020.

[184] Renê Oliveira, Carlos Montez, Azzedine Boukerche, and Michelle S Wangham. Reliable data dissemination protocol for VANET traffic safety applications. *Ad Hoc Networks*, 63:30–44, 2017.

[185] Azzedine Boukerche, Xuzhen Cheng, and Joseph Linus. A performance evaluation of a novel energy-aware data-centric routing algorithm in wireless sensor networks. *Wireless Networks*, 11(5):619–635, 2005.

[186] Azzedine Boukerche, Sajal K Das, and Alessandro Fabbri. Analysis of a randomized congestion control scheme with dsdv routing in ad Hoc wireless networks. *JPDC*, 61 (7):967–995, 2001.

[187] Azzedine Boukerche and E Robson. Vehicular cloud computing: Architectures, applications, and mobility. *Computer Networks*, 135:171–189, 2018.

[188] Noura Aljeri and Azzedine Boukerche. A dynamic MAP discovery and selection scheme for predictive hierarchical MIPv6 in vehicular networks. *IEEE Trans. Veh. Technol.*, 69(1):793–806, 2019.

[189] David Tse and Pramod Viswanath. *Fundamentals of wireless communication*. Cambridge university press, 2005.

[190] Azzedine Boukerche. *Algorithms and protocols for wireless and mobile ad hoc networks*, volume 77. John Wiley & Sons, 2008.

[191] *Handbook of algorithms for wireless networking and mobile computing*. Chapman & Hall/CRC, Boca Raton, FL, 2006.

[192] Azzedine Boukerche and Steve Rogers. Performance of GZRP ad hoc routing protocol. *Journal of Interconnection Networks*, 2(01):31–48, 2001.

[193] A Boukerch, L Xu, and K EL-Khatib. Trust-based security for wireless ad hoc and sensor networks. *Computer Communications*, 30(11):2413–2427, 2007.

[194] Azzedine Boukerche, Xiuzhen Cheng, and Joseph Linus. Energy-aware data-centric routing in microsensor networks. In *Proc. ACM MSWiM Workshops*, pages 42–49, 2003.

[195] Noura Aljeri and Azzedine Boukerche. Advice-loc: An adaptive vehicle-centric location management scheme for intelligent connected cars. *Ad Hoc Networks*, 107: 102223, 2020.

[196] Noura Aljeri and Azzedine Boukerche. A two-tier machine learning-based handover management scheme for intelligent vehicular networks. *Ad Hoc Networks*, 94:101930, 2019.

[197] Peng Sun, Noura AlJeri, and Azzedine Boukerche. An energy-efficient proactive handover scheme for vehicular networks based on passive RSU detection. *IEEE Trans. Sustain. Comput.*, 5(1):37–47, 2018.

[198] Noura Aljeri and Azzedine Boukerche. An efficient handover trigger scheme for vehicular networks using recurrent neural networks. In *Proc. ACM MSWiM*, pages 85–91, 2019.

[199] Lili Du and Hoang Dao. Information dissemination delay in vehicle-to-vehicle communication networks in a traffic stream. *IEEE Trans. Intell. Transp. Syst.*, 16(1): 66–80, 2014.

[200] Anastasios Kouvelas, Jennie Lioris, S Alireza Fayazi, and Pravin Varaiya. Maximum pressure controller for stabilizing queues in signalized arterial networks. *Transp. Res. Rec.*, 2421(1):133–141, 2014.

[201] Azzedine Boukerche, Renato B Machado, Kathia RL Jucá, João Bosco M Sobral, and Mirela SMA Notare. An agent based and biological inspired real-time intrusion detection and security model for computer network operations. *Computer Communications*, 30(13):2649–2660, 2007.

[202] Azzedine Boukerche and Carl Tropper. A static partitioning and mapping algorithm for conservative parallel simulations. In *Proc. PADS Workshops*, pages 164–172, 1994.

[203] Azzedine Boukerche, Horacio ABF Oliveira, Eduardo Freire Nakamura, and Antonio AF Loureiro. DV-Loc: a scalable localization protocol using voronoi diagrams for wireless sensor networks. *IEEE Wireless Communications*, 16(2):50–55, 2009.

[204] Azzedine Boukerche and Amir Darehshoorzadeh. Opportunistic routing in wireless networks: Models, algorithms, and classifications. *ACM CSUR*, 47(2):1–36, 2014.

[205] Rodolfo WL Coutinho, Azzedine Boukerche, Luiz FM Vieira, and Antonio AF Loureiro. GEDAR: geographic and opportunistic routing protocol with depth adjustment for mobile underwater sensor networks. In *Proc. IEEE ICC*, pages 251–256, 2014.

[206] Azzedine Boukerche and Yonglin Ren. A secure mobile healthcare system using trust-based multicast scheme. *IEEE J. Sel. Areas Commun.*, 27(4):387–399, 2009.

[207] Azzedine Boukerche and Mirela Sechi M Annoni Notare. Behavior-based intrusion detection in mobile phone systems. *JPDC*, 62(9):1476–1490, 2002.

[208] Rodolfo Wanderson Lima Coutinho, Azzedine Boukerche, Luiz Filipe Menezes Vieira, and Antonio Alfredo Ferreira Loureiro. Geographic and opportunistic routing for underwater sensor networks. *IEEE Trans. Comput.*, 65(2):548–561, 2016.

[209] Kaouther Abrougui, Azzedine Boukerche, and Richard Werner Nelem Pazzi. Design and evaluation of context-aware and location-based service discovery protocols for vehicular networks. *IEEE Trans. Intell. Transp. Syst.*, 12(3):717–735, 2011.

[210] Sergio Correia, Azzedine Boukerche, and Rodolfo I Meneguette. An architecture for hierarchical software-defined vehicular networks. *IEEE Communications Magazine*, 55(7):80–86, 2017.