DEEP REINFORCEMENT LEARNING FOR TRANSIT SIGNAL PRIORITY IN A CONNECTED ENVIRONMENT

Meng LONG

PhD

The Hong Kong Polytechnic University

2024

The Hong Kong Polytechnic University

Department of Electrical and Electronic Engineering

Deep Reinforcement Learning for Transit Signal Priority in A Connected Environment

Meng LONG

A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy

August 2023

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____ (Signed)

Meng LONG (Name of student)

Abstract

The growing urbanization and increasing population break the balance of travel demand and road capacity in metropolises, causing heavy traffic congestion. Public transit can transport more people than private vehicles while occupying fewer road resources. Improving transit service can effectively alleviate traffic congestion as better transit service can attract more travelers by buses from private cars. Moreover, traffic lights at intersections bring signal delays, which even become the primary source of transit delays and harm bus efficiency and reliability. Hence, transit signal priority (TSP) is an essential measure to reduce traffic congestion and promote transit reliability at signalized intersections. The emerging connected vehicle technology and reinforcement learning (RL) algorithms provide the opportunity for more intelligent TSP strategies due to more detailed and accurate information and more robust algorithms.

The first work proposes an *extended Dueling Double Deep Q-learning with Invalid action masking* (eD3QNI) algorithm for TSP strategy at isolated intersections in a connected environment. The algorithm considers multiple conflicting bus priority requests and the constraints on the traffic light and phase skipping rule, aiming to improve the person delay of buses. Simulation results demonstrate that eD3QNI produces lower average person delay and schedule delay than other methods. It also shows that the invalid action masking (IAM) method is superior to the usual variable decision points (VDP) method in terms of high convergence speed, effective performance improvement, and application of domain knowledge on the RL algorithm.

To extend the above work into a multi-intersection environment, the second work develops a *Cooperative TSP strategy of Variable phase* (CTSPV) by MARL to improve transit schedule adherence at arterial roads. Agents determine the phase of the next step so that the phase sequence and duration are varied with the real-time traffic conditions considering the trade-off between transits and non-transits, the multiple conflicting bus requests, and the cooperation between different agents. Three kinds of traffic constraints are tested, and their results verified the necessity of proper restrictions on RL to guarantee experience quality and training efficiency. This work analyzes the signal timing pattern difference between CTSPV and fixed-time signal and proves the good performance of RL-learned knowledge.

As headway regularity is another essential indicator for transit reliability besides schedule adherence, the third work develops a *Cooperative TSP strategy of Variable phase for Headway adherence* (CTSPVH) to improve transit headway adherence at arterial roads. The proposed approach considers four critical aspects, i.e., complicated states with multiple conflicting bus requests, rational actions constrained by domain knowledge, comprehensive rewards balancing buses and cars, and a collaborative training scheme among agents. They are correspondingly addressed by proper state representation, IAM algorithms to mask out irrational actions, and reward functions formulated by general traffic queue and transit headway deviation.

Those three strategies solve the key problems of TSP approaches and incorporate traffic domain knowledge with RL to ensure action rationality, avoiding severe congestion and serious accidents caused by irrational actions. Therefore, those three intelligent and safe strategies have bright prospects in practical applications to improve transit efficiency and reliability.

Publications

The following papers have been generated during the my PhD candidacy.

Peer-Reviewed Journal Papers:

Long, M., Zou, X., Zhou, Y., & Chung, E. (2022). “Deep reinforcement learning for transit signal priority in a connected environment”. Transportation Research Part C: Emerging Technologies, 142.

Long, M., Chung, E, Sulejic, D., & Sabarc, N.R. (2024). “A Cooperative Longitudinal Lane-changing Distributions Advisory for a Freeway Weaving Segment”. Journal of Intelligent Transportation Systems: Technology, Planning, and Operations.

Yao, R.H., Zhang, W.S., & **Long, M**. (2022). “DLW-Net model for traffic flow prediction under adverse weather”. Transportmetrica B: transport dynamics 10 (1), 499-524.

Zou, X.X., Chung, E., Zhou, Y., & **Long, M.**, Lam, WHK. (2023). “A feature extraction and deep learning approach for network traffic volume prediction considering detector reliability”. Computer-Aided Civil and Infrastructure Engineering.

Conference:

Long, M., & Chung, E. “Transit Signal Priority for Signalized Intersections with Deep Reinforcement Learning", present on the 26th International Conference of Hong Kong Society for Transportation Studies, Hong Kong, 12 December, 2022.

Long, M., & Chung, E. “Transit Signal Priority for Arterial Road with Multi-Agent Reinforcement Learning”. The 8th International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS 2023), Nice, French, 14-16 June, 2023.

Long, M., & Chung, E. (2024). “Cooperative Transit Signal Priority for Headway Adherence on Arterial Roads with Multi-Agent Reinforcement Learning”. The 24th COTA International Conference of Transportation Professionals (CICTP). (Accepted for presentation)

Acknowledgments

I would like to express my deepest gratitude to my supervisor Prof. Edward Chung. It is he who offered me an opportunity to pursue my Ph.D. study at PolyU and also a chance to exchange for half of a year in UTokyo. I am extremely grateful for his patient guidance, continuous encouragement, and kind assistance. He is an esteemed leader in my research life, leading me to find interesting and meaningful research topics, encouraging me to program from simple to complicated models, and helping me to find the primary research significance and contributions. He not only assisted in study conduction and paper writing, but also encouraged me to practice my presentation skills and taught me how to present my studies for different audiences and listen to their comments. He has been my role model in research and life as his enthusiastic research attitude and regular physical exercise motivate me to work hard with a healthy body. Moreover, special thanks to his genial wife Yoko for leaving pleasant memories of my study and life in both Hong Kong and Japan.

I would like to extend my sincere thanks to my co-supervisor Dr. Weihua Gu. He generously gives me professional advice every time in my presentation, recommends me high-quality papers related to my research problems. I sincerely appreciate his willingness to help me.

I am also thankful to Prof. Takashi Oguchi, my supervisor at UTokyo. Firstly, I would like to thank him for giving me the opportunity to join his lab, which has given me a precious memory of being in a foreign country. Also, thank him for arranging time to discuss with me during his busy hours and providing me with valuable suggestions.

My thanks should go to my research teams and colleagues in PolyU and UTokyo, including Dr. Hongbo Ye, Dr. Wei Ma, Dr. Yue Zhou, Dr. Xiaowen Bi, Dr. Mingyu Shen, Dr. Qiaolin Hu, Dr. Xiao Yang, Dr. Jiaxin Wen, Dr. Tianyang Han, Xiexin Zou, Rong Cheng, Jieming Chen, Zhixian Tang, Chaoyun Wang, Ruoheng Wang, Jiangfeng Zhang, Hailong Shi, Zeren Xing, Ruoxuan Zhong, Jaya Varshini Kala, Somporn Sahachaisaree, Noriko Asahara and all others in EF113 and Cw504. They help me a lot in my research and life. Many thanks also to my other friends Wandong Xu, Lejing Li, and Dr. Xin Jin. Their company encourages me to go through difficulties and enjoy life in Hong Kong. In addition, I also would like to mention my previous research teams and online old friends, including Prof. Shengchuan Zhao, Prof. Ronghan Yao, Dr. Wensong Zhang, Yalin Liang, Wenting Ma, Lan Yang, Hongming Jia, Liqiong Zheng, Mingli Wang, Qiuxia Luo, Lu Zhang, Longze Cong. Their continuous care and heartfelt advice gave me the courage to pursue PhD and walk to the end.

I want to thank PolyU for providing a monthly stipend, conference grant, and financial support for the research attached student program (RASP), with which I can concentrate on my study and broaden my international horizon. And many thanks to all reviewers and examiners for their valuable comments on the improvement of my thesis.

Finally, I would like to express my gratitude to my family. Although we were separated for a long time because of the pandemic and RASP, they have always been my strong backing. No matter where I am or what I am going through, just thinking about them fills my heart with strength and warmth, which gives me the courage and ability to overcome any difficulties.

Contents

List of Figures

List of Tables

List of Abbreviations

TSP	Transit Signal Priority
TSC	Traffic Signal Control
RL	Reinforcement Learning
DRL	Deep Reinforcement Learning
DNN	deep neural networks
MARL	Multi-agent Reinforcement Learning
ETA	estimated time of arrival
CV	connected vehicle
V2I	vehicle to infrastructure
OBU	onboard units
RSU	roadside unit
APC	automatic passenger counters
MDP	Markov Decision Process
DTDE	distributed training decentralized execution
CTCE	centralized training centralized execution
CTDE	centralized training decentralized execution
EB, WB, SB, and NB	eastbound, westbound, southbound, and northbound
FS	fixed sequence
VP	variable phase
APD	average person delay
APDB	average person delay of buses
APDC	average person delay of cars
PER	Prioritized Experience Replay
IAM	Invalid Action Masking
VDP	variable decision point
D3QN	Dueling double deep Q-learning
eD3QNI	extended Dueling double deep Q-learning with invalid action masking
SUMO	Simulation of Urban MObility
FT	fixed-time signal
ATSP	active transit signal priority
ATSPF	active transit signal priority of fixed sequence scheme
ATSPV	active transit signal priority of variable phase scheme
A2C	Advantage Actor-Critic
DDPG	Deep Deterministic Policy Gradient
CTSPV	Cooperative TSP strategy of Variable phase
OD	Origin-Destination
CFT	coordinated fixed-time signal
IFTPS	independent fixed time with phase skipping
LQFA	Long Queue First Algorithm
IQL	Independent Q Learning
VDN	Value Decomposition Network
FTRBS	Fixed Time with Rule-Based Skipping
ADRBS	Average Duration with Rule-Based Skipping
FRBS	F-value with Rule-Based Skipping
BH	Bus Holding strategy
CTSPVH	Cooperative TSP strategy of Variable phase for Headway adherence
AHD	average headway deviation

ABHD	average bi-headway difference
APRDB	average person running delay on bus
TPDB	total person delay on bus

List of Notations

s_t	state at time t
s'_t	next state for time t
a_t	action at time t
r_t	reward at time t
p	phase
P	full phase set
π	policy
π^*	optimal policy
G_t	expected return
γ	discounting factor
$V_\pi(s)$	state-value function under policy π when starting from state s
$Q_\pi(s,a)$	action-value function of choosing action a in state s under policy π
Q^{masked}	masked action-value function
$Q_{tot}(S_t, \Lambda_t; \Theta)$	total Q-value of joint action Λ_t by main networks with parameters Θ given the global state S_t at time t
$Q^i(s_t^i, a_t^i; \theta^i)$	Q-value of action a_t^i for agent i by the network with parameters θ^i given the state s_t^i at time t
S_A	state matrix of average speed
$S_{Al\varphi}$	the average speed on the lane φ of the approach link l
S_{Alm}	average speed of movement m at the approach link l
L_Q	state matrix of queue length
$L_{Ql\varphi}$	queue length on the lane φ of the approach link l
L_{Qlm}	queue length of movement m at the approach link l
L_{Qp}	maximum queue length of phase p's movements
$L_{Ql\varphi t}$	queue length on the lane φ of the approach link l at time t
Φ_l	lane set of approach link l
M_l	movement set of approach link l
Ψ	approach set
P_C	encoding vector of current phase
Γ	bus set for bus states information
O	bus occupancy
O_{kt}	passenger occupancy of bus k at time t
O_{pb}	passenger occupancy of bus b which requests the green time of phase p
D	distance to the stop line
D_{pb}	distance to the stop line of bus b which requests the green time of phase p
SD	schedule delay
SD_{kt}	schedule delay of bus k at time t
SD_{pb}	schedule delay of bus b which requests the green time of phase p
Å	action space
A_t	valid action space at time t
k	index of the bus in the whole intersection
$\hat{d}_{kt}$	good performance delay of bus k comparing to the velocity of 5.56 m/s
d_{kt}	normal delay of bus k comparing to the velocity of 16.67 m/s
NP_{ct}	number of passengers in the car c
d_{ct}	normal delay of car c comparing to the velocity of 16.67 m/s
$APDB$	average person delay of buses

$APDC$	average person delay of cars
APD	average person delay
$Que_{l\varphi t}$	queue length on the lane φ of the approach link l at time t
T	total simulation time
S_{mask}	mask states
G_{min}	minimum green time
G_{max}	maximum green time
$Q_{threshold}$	queue length threshold for skipping rule
Δt_{λ}	time step length at λ step
ϵ_{max}	maximum epsilon
ϵ_{min}	minimum epsilon
ϵ_{dec}	epsilon decrement
α, β	learning rate
S_{mem}	replay memory size
S_{rep}	size of replay buffer
S_{bat}	batch size
N_{rep}	the number of iterations to replace parameters
N_e	the number of episodes
$Twarm$	simulation warm-up time
θ	main network parameters
θ^-	target network parameters
w, w_1, w_2	weights of reward functions
D_{Cross}	length of the pedestrian crossing
V_{Ped}	average velocity of pedestrians
C	cycle length
q	traffic volume
f	saturation flow
$Det_{l\varphi}$	detector length on the lane φ of the approach link l
N	the number of intersections
$Speed_p$	average speed of phase p movements
$Queue_p$	average queue length of phase p movements
$Prio_p$	bus urgency of phase p movements
F_p	urgency of phase p
ξ, ϱ, η	parameters of F-value function
FH_t^k and BH_t^k	forward and backward headway of bus k at time t
FHD_t^k and BHD_t^k	the forward and backward headway deviation of bus k at time t
δ_b, δ_c and δ_f	the backward, current, and forward buses in Δt

Chapter 1

Introduction

1.1 Background

The development of public transportation is a common and effective measure to alleviate traffic congestion caused by the growing urbanization and increasing population in metropolises worldwide (Chu et al., 2020). It can transport more people than private vehicles while occupying fewer road resources. Specifically, a private car can only carry up to 5 travelers at a time; in contrast, an ordinary single-deck bus can carry up to 80 passengers. In a sense, one bus can replace at least 16 cars to transport people. Improving the transit service can attract more travelers to use it rather than private cars in the city with a small bus ratio, but also can effectively improve the bus efficiency and reliability in the city with an already high bus ratio (Xu et al., 2022b). In addition, Traffic lights are widely used to control conflicting movements at intersections by assigning their right of way over time, e.g., there are already 1916 signalized intersections in Hong Kong by the end of 2019, according to the statistics of the transport department. It can improve traffic efficiency but also block some movements to wait for the green time at intersections, including buses. The signal delays caused by traffic lights are the primary source of transit delays, accounting for over 20% of total travel time in dense areas (Shalaby et al., 2021), and would lead to schedule delays. As a consequence, traffic signals highly affect transit efficiency and reliability.

Hence, various measures have been implemented to enhance the transit service quality by prioritizing buses these years. Those measures are divided into two kinds: one is the spatial approach to provide bus lanes, such as dedicated or intermittent bus lanes, and flexible space-sharing methods (Guler and Cassidy, 2012; Haitao et al., 2018); the other is the temporal method to prioritize buses by a traffic signal, i.e., transit signal priority (TSP) (Bie et al., 2020; Ghaffari et al., 2020). The latter method is more cost-effective than the former, owing to the fewer changes to the infrastructure and greater flexibility to real-time traffic. Additionally, some latest studies cooperated the holding control at bus stops (Abdelhalim and Abbas, 2021) and the speed advisory for buses (Lee and Wang, 2022) with TSP strategies to provide signal priority for more buses at once.

Public transport researchers developed various TSP strategies to prioritize buses going through intersections, which can significantly decrease signal delays and improve schedule adherence (Sheffield et al., 2021; Xiao et al., 2022). The advanced TSP system typically comprises four fundamental parts (Smith et al., 2005). First, the *Detection System* utilized a range of technology to collect information on general traffic and transit vehicles, like queue length at approaches and buses' location and passenger occupancy. Those detecting technologies include loop detectors, infrared detectors, radio frequency tag readers, GPS, connected vehicle (CV) technology, etc. Second, the *Priority Request Generator* generates single or multiple priority requests. Third, the *Priority Request Server* receives and triages those requests as necessary. Fourth, the *Signal Controller* responds to those requests through various

TSP strategies and executes signal control.

Existing TSP strategies fall into three types, i.e., passive, active, and adaptive strategies (Bagherian et al., 2015; Truong et al., 2017). ***Passive strategy.*** The transit signal should be pre-set based on the arrival time of buses. Therefore, its efficiency is highly reliant on the stability and accuracy of traffic demands, and it will not be an intelligent choice for unpredictable and variable traffic. ***Active strategy.*** There need several detectors at intersections to detect the approaching of the bus, and then signal timing is updated. Various approaches have been proposed to respond to a transit priority request, such as 1) green extension, which extends the green time to allow the detected bus to pass through the intersection, 2) red truncation, also called early green, which truncates red time earlier to shorten the waiting time for the green time serving detected buses, 3) phase insertion which inserts one transit phase into the normal signal sequence, 4) phase skipping which skips non-priority phases to let detected bus clear the intersection faster, and so on. And the results of this strategy also mainly depend on the accuracy of some variables. Active TSP can also be subdivided into unconditional (Shi et al., 2017) and conditional methods (Cvijovic et al., 2022). Unconditional active TSP would prioritize buses whenever requested; conversely, conditional active TSP provides bus priority only when some criteria are satisfied, such as large schedule delay, heavy passenger occupancy, etc. ***Adaptive strategy.*** It should be implemented based on the traffic states to optimize some general performance measures, such as maximizing headway regularity (Ling and Shalaby, 2003), minimizing the vehicular delay of all vehicles (Mirchandani et al., 2001), person delay (Christofa et al., 2013), and other weighted combinations of person delay or schedule delay (Yang et al., 2019; Zeng et al., 2021).

1.2 Motivations and Objectives

The development of public transportation has emerged as a widely adopted solution to tackle the escalating traffic congestion resulting from rapid urbanization and population growth. Among the various operational strategies, TSP stands out as one of the most effective approaches to enhance the efficiency and reliability of transit vehicles, thereby fostering the advancement of public transportation. In this context, the **motivations** for this research are as follows.

(1) Opportunities brought by the RL methods:

Reinforcement learning (RL) algorithms have been widely applied to traffic signal control (TSC) and have achieved great success in real-time signal control due to the following two advantages: 1) ***Flexibility***: RL can be model-free, and it does not require a predefined analytical model of the traffic system. It learns directly from interactions with the environment and can capture the non-linear relationships between traffic signal settings and traffic flow, making it suitable for situations where the traffic dynamics are complex or unknown. This flexibility is particularly advantageous in real-world traffic scenarios, which often involve non-linear and stochastic behavior. Therefore, RL-based TSC is more robust as it can adapt to various complex and dynamic traffic scenarios more effectively compared to traditional methods that rely on fixed control rules. 2) ***Optimization***: RL can optimize traffic signal control policies based on long-term cumulative rewards. By considering the overall system performance, RL algorithms can optimize traffic signal control policies to minimize congestion, reduce travel times, and improve traffic flow efficiency.

The existing TSP methods, like traditional TSC, also have certain limitations in dealing with complex and dynamic traffic situations and long-term optimization. Therefore, it is an opportunity to develop smarter and more efficient TSP with the deep reinforcement learning (DRL) algorithm and CV

technology, adapting to complicated traffic situations and optimizing long-term performance.

(2) Challenges posed by the RL methods

However, the application of RL in TSC also has some challenges, including 1) ***Safety***: RL algorithms must be carefully designed and trained to ensure that they do not inadvertently create unsafe traffic conditions or violate traffic regulations; 2) ***Scalability***: RL algorithms may struggle to scale to large, complex traffic networks with varying traffic patterns and multiple intersections.

Most researchers in computer science devote themselves to improving the RL algorithm and verifying the efficiency of their algorithm in the application of TSC. Still, the formulation of their RL framework lacks the transportation domain knowledge, including the minimum and maximum green time constraints and the phase-skipping rules, to consider traffic problems and ensure safety. For example, most existing RL methods allow for fully flexible skips decided by agents without any constraint, which would confuse drivers and frustrate drivers getting skipped multiple times. Considering safety and equity issues, we should restrict irrational actions that would cause severe problems in actual traffic. Hence, it is necessary to develop a rational DRL-based TSP that combines transportation engineers' knowledge.

Even though some works consider minimum green time constraints, they fulfill this requirement by setting a very long decision step, which exceeds the minimum green time (El-Tantawy et al., 2014; Kai et al., 2019; Wan and Hwang, 2018). As an example, the agent is only allowed to modify signals every 15 s, but for TSP control, this large step length will cause signal timing not to be modified in time to give transit priority, as after 15 s, the buses may have been queuing for some time already. Therefore, the proper method to consider transportation domain knowledge for TSP should be proposed.

The second scalability challenge of RL mentioned above encourages us to think about how to apply RL to the environment with dynamic traffic patterns and multiple intersections. Hence, it is also crucial to consider scenarios with various traffic volumes, bus schedules, etc., and extend the application of RL from isolated intersections into multiple intersections.

(3) Other issues related to TSP strategies

Other practical issues need to be addressed for the TSP strategies. Firstly, giving priority to buses will deprive some private cars of benefits and thus affect their performance. Hence, the trade-off between transit and non-transits in TSP strategies is somewhat tricky. Secondly, in complex real-world environments, there are multiple bus lines so that many buses may approach an intersection from conflicting directions at the same time and request signal priority. Effectively handling multiple conflicting bus priority requests is a crucial problem and highly influences the performance of TSP strategies. Thirdly, traffic flows between adjacent intersections can affect each other, so when optimizing TSC, researchers consider the cooperation between them, aiming to create a steady flow. Cooperative optimization becomes much more complicated for intersections with TSP strategies, but it is still critical as it can amplify the benefits of TSP strategies. Fourthly, TSP strategies can be designed to realize different operational goals, including person delay, bus schedule adherence, and bus headway adherence. It is important to think of the corresponding RL framework to satisfy different needs.

Based on these motivations, the **objectives** of this study are as follows:

- Propose efficient TSP strategies with DRL methods that can adapt to complex traffic conditions and optimize performance by long-term rewards.
- Develop TSP strategies that incorporate the transportation domain knowledge about traffic

light constraints (i.e., the minimum and maximum green time) and phase skipping rule into the DRL framework so that the actions chosen by the target methods are guaranteed to be rational and effective.

- Propose TSP strategies for isolated signalized intersections and then extend to arterial roads with multiple signalized intersections, considering various traffic conditions.
- Effectively solve the trade-off between transit and non-transits, multiple conflicting bus priority requests, and cooperation among agents for arterial roads, and significantly improve the person delay, bus schedule adherence, or headway adherence.

1.3 Organization of thesis

Based on the above research backgrounds and objectives, the rest of this thesis concentrates on answering the question of how to implement transit signal priority by the reinforcement learning approaches efficiently. The remainder of this thesis is structured as follows:

- Chapter 2 presents a comprehensive literature review. It begins with a description of existing transit signal priority strategies, followed by a brief demonstration of the data collection of CV technology and elementary knowledge of RL algorithm. These facilitate more intelligent TSP strategies owing to providing detailed and accurate information and robust algorithms for traffic signal control. Then the recent research on the application of RL algorithms in traffic signal control is also reviewed. The last part of the literature review describes transit reliability and summarizes its related research as it is a vital aspect for evaluating bus operation.

- Chapter 3 proposes an RL-based TSP strategy for the **isolated intersection** in a connected environment. It considers multiple conflicting bus priority requests and the constraints on the traffic light and phase skipping rule, aiming to improve the person delay of buses. Simulation experiments are conducted to demonstrate the good performance of the proposed method. This chapter also studies the effects of connected buses' penetration rates on the proposed method and the performance of different specific reward functions.

- Chapter 4 develops a MARL-based TSP strategy for the arterial road with **multiple intersections** to improve **transit schedule adherence**. It considers the trade-off between transits and non-transits, the multiple conflicting bus requests, the constraints on the traffic light and phase skipping rule, and the cooperation between different agents. Experiments are conducted, and the effects of traffic light constraints on the RL algorithm are analyzed. To figure out how the proposed RL-based method works, this chapter also analyzes the signal timing pattern from cycle length, phase duration, skipping pattern, and action-choosing pattern. Rule-based strategies are generalized from those patterns and are tested to prove the good performance of RL-learned knowledge.

- Chapter 5 extends the work of Chapter 4 to improve **transit headway adherence**. It considers all the factors of Chapter 4 and also adds 1s all-red time to consider the loss of phase transition. Simulation experiments are conducted to show the results compared with traditional fixed-time signal and bus holding strategy. In the end, this chapter also analyzes bus holding's limitations and discusses the proposed method's benefits.

- Chapter 6 concludes the thesis by summarizing those studies, specifying contributions, and discussing future works.

Chapter 2

Literature review

2.1 Transit signal priority

Transit signal priority (TSP) seeks to modify the existing traffic lights in real-time to facilitate bus movements to pass through intersections for improving bus service. Relevant research has proven a wide range of benefits, such as reducing bus travel time (Ghaffari et al., 2020), decreasing bus or passenger delay (Ma et al., 2014), improving bus schedule adherence (Yang et al., 2019; Zeng et al., 2021). Literature related to TSP strategies mainly focuses on the following issues:

1) Optimizing objectives. Numerous researchers have dedicated themselves to seeking the TSP strategies with considerable improvement on bus service and less damage on private vehicles by formulating the proper objectives and constraints of optimization algorithm (Ma et al., 2014; Thodi et al., 2021; Yang et al., 2019; Zeng et al., 2021). The objectives must be suitable for finding the trade-off between prioritized buses and non-prioritized vehicles. Person-based optimizing objectives are more commonly used than vehicle-based ones, like maximizing person capacity (Ma et al., 2014), minimizing person delay (Christofa et al., 2016; Christofa et al., 2013; Thodi et al., 2021). They aim at system efficiency, but those approaches do not give transit absolute priority to attract travelers from other modes. In addition, these methods are model-based, and the model formulation and variable selection are highly dependent on the objective setting. As a result, the model has to be redesigned if we want to achieve other different objectives.

2) Accuracy of variables. The effectiveness of active TSP depends on the accurate information of buses, e.g., the estimated time of arrival (ETA) of buses, which is of great significance to respond to the priority request of buses. Thus, we not only need to come up with an efficient model to implement the active TSP but also need to work out more precise indicators for this active strategy. In recent years, many efforts have been made to predict more accurate ETA, like traffic-based calculation, regression models, and machine learning methods (Zhang et al., 2020). Hence, the requirement of variables' accuracy limits the effectiveness of TSP and increases the difficulty of proposing a better strategy.

3) Conflicting priority requests. Cities always have intertwined bus routes, leading to conflicting priority requests. The serving sequence has tremendous significance on traffic efficiency. In the past, the first-come-first-serve policy was commonly used to arrange multiple bus priority requests, but it was proved to lead to extra delays by later studies (Head et al., 2006). Ma et al. (2013) firstly interpreted the serving sequence of multiple requests as a multistage decision process. They proposed a dynamic programming model for an isolated intersection under different traffic demand and bus states, i.e., bus occupancy and schedule deviation. Christofa et al. (2013) and Hu et al. (2016) developed different person-delay-based optimization methods to schedule multiple conflicting priority requests at an isolated intersection. These optimization processes are mixed-integer nonlinear program and binary mixed-integer linear program, respectively, of which objective functions minimize the total and per

person delay. However, as all of them are model-based approaches, the performance is dependent on the model assumption, and models only are applicable to realize the designed specific goals. Xu et al. (2022a) calculated the priority value for each bus by a fuzzy controller to handle the multiple conflicting TSP requests. Xu et al. (2022b) integrated TSP into max-pressure control policy and determined the serving sequence of conflicting requests by the optimal solution of a mixed-integer linear program that maximizes the total pressure; however, their method only works for bus rapid transit with exclusive bus lanes.

4) Traffic signal cycle. Many studies have assumed fixed cycle length when designing TSP strategies to easily formulate the optimization objective, e.g. minimizing cumulative delay in one cycle (Christofa et al., 2016; Christofa et al., 2013; Thodi et al., 2021). Although the fixed cycle length assumption can facilitate the implementation of arterial traffic signal coordination, it will limit the performance of TSP strategies at isolated intersections. Yu et al. (2017) realized this issue and worked out the TSP strategy with flexible cycle length by formulating objectives as the cumulative delay within a fixed planning horizon including two cycles. Later, they proposed implementing phase rotation in the person-based TSP in 2018 and continued integrating phase sequence optimization into the person-based framework in 2022 to provide TSP, which can optimize phase sequences, cycle lengths, and phase splits simultaneously by minimizing the cumulative delay within a fixed time span. However, research on TSP with flexible cycles is still limited. Most of the existing methods with flexible cycles still calculate signal timings by cycles or make adjustments to one base signal timing scheme; however, it is not intelligent enough to react immediately to the real-time traffic state during a cycle.

5) Cooperation of multiple intersections. Existing research developed a range of TSP strategies at isolated intersections using rule-based logic (Zlatkovic et al., 2012), mathematical programming model (Liu et al., 2021), and artificial intelligence (Alizadeh Shabestray and Abdulhai, 2019; Stevanovic et al., 2008). However, researchers also realized that cooperation between adjacent intersections is essential to ensure that the gain of TSP at the current intersection would not be wasted at downstream intersections. For coordinated corridors, Christofa et al. (2016) presented a real-time signal control system by one mixed integer linear program to minimize the person delay of two consecutive intersections during a design cycle so as to provide priority to transits with high occupancy. Zeng et al. (2021) devised route-based TSP by linear programming to provide bus priority at the route level, aiming to use the least signal deviations to improve bus schedule adherence for all considered buses in any route, not only in arterial. From the perspective of a network level, Li et al. (2022) proposed a regionally coordinated bus priority signal control approach considering pedestrian and passenger delays. It contains two stages solved by genetic algorithms: 1) to obtain the basic regionally coordinated signal timing and 2) to implement transit signal priority based on bus arrival time. Xu et al. (2022a) described a multi-agent TSP control method using fuzzy logic for traffic networks considering conflicting priority requests. In this approach, one agent controlled one intersection and determined the signal timing in the next cycle based on collected local information and communicated neighbor information. In summary, those existing methods are all model-based and suffer from common limitations of model-based methods, such as optimizing objectives, accuracy of bus arrival time, and model assumptions. They use cycle-based optimization or modify one base signal timing, which cannot respond to real-time traffic in a timely.

To summarize the above five aspects, most existing methods are model-based, and their performance highly relies on objective settings, variable accuracy, and model assumptions. There is a need to develop more robust methods that can consistently achieve desirable outcomes without

significant performance degradation in the face of changing objectives, inaccurately predicted variables, and discrepancies between actual situations and model assumptions. In addition, most existing methods calculate signal timings by cycles or make adjustments to one base signal timing scheme when considering flexible traffic signal cycles and multi-intersection cooperation. However, it would be more flexible and intelligent to determine the signal at each time step according to the current traffic state of general traffic and buses. The CV technology and intelligent algorithm provide new opportunities for the advanced TSP strategy to better tackle the problems mentioned above.

2.2 Connected vehicle technology

CV technology makes communication among vehicles and vehicles/infrastructures possible. Vehicles equipped with the vehicle to infrastructure (V2I) Onboard Units (OBU) can transmit route data and current operating states to the Roadside Unit (RSU), which could be connected with the traffic signal control system (Zeng et al., 2021). More accurate and detailed traffic information can be collected in such an environment, like the real-time bus location from GPS, bus schedule deviation, bus occupancy from electronic payment systems or automatic passenger counters (APC), which are crucial variables for TSP. More innovative TSP strategies can be developed as more information can be obtained from advanced communication technologies than sparse road detectors and be transmitted quickly (Cvijovic et al., 2022).

2.3 Reinforcement learning

(1) Single-agent RL

Reinforcement learning (RL) is a subfield of machine learning and an effective tool for solving problems formulated as the Markov Decision Process (MDP). Given the process is in one state s_t at step t, the decision-maker chooses an action a_t and the process would move to another state s_{t+1} at the next step $t+1$, finally obtaining a reward r_t correspondingly. State, action, reward, and next state form one transition $< s_t, a_t, r_t, s_{t+1} >$.

Figure 2.1 shows the structure of the RL method. In RL, we call the decision-maker the *agent*, and the thing outside the agent is called the *environment*, which can interact with the agent to execute actions and provide the next state and immediate reward (Sutton and Barto, 2018). After numerous repeated interactions with the environment, the RL agent strives to find the optimal policy π^* which can maximize the expected return G_t. Some terminologies used are explained as follows.

Policy. It maps states to probabilities of selecting each action. For example, if the agent is following policy π to choose action, then $\pi(a|s)$ denotes the probability of choosing action a given state s.

Return. This is a long-term reward, given by the sum of future discounted rewards:

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \tag{2.1}$$

where γ is the discounting factor, $0 \leq \gamma \leq 1$, which determines the present value of future rewards, e.g., if a reward r_{t+k} will be received after k time steps, the present value of the reward worth only γ^k times r_{t+k}. Hence, if $\gamma = 0$, the agent only focuses on maximizing immediate rewards; in contrast, if $\gamma < 1$, the agent will consider both immediate and future rewards. The closer γ is to 1, the stronger the agent takes future rewards into account.

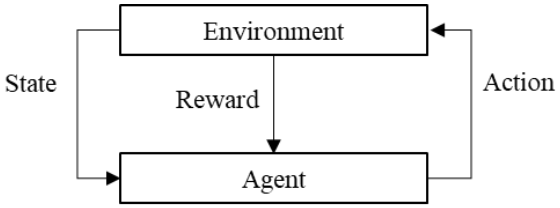


Figure 2.1. The structure of the RL method.

Value functions. RL involves state-value functions (or action-value functions) to estimate the goodness of being in a given state (or of performing a given action in a given state) for the agent. The goodness is valued by the expected return.

The state-value function $V_\pi(s)$ is the expected return under policy π when starting from state s, given by

$$V_\pi(s) = \mathbb{E}_\pi[G_t|s_t] = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \,|s_t = s] \tag{2.2}$$

where $\mathbb{E}_\pi[*]$ denotes the expected value of a variable under policy π, and t is the time step.

Similarly, the action-value function $Q_\pi(s,a)$ is the expected return of choosing action a in state s under policy π, given by

$$Q_\pi(s,a) = \mathbb{E}_\pi[G_t|s_t = s, a_t = a] = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \,|s_t = s, a_t = a] \tag{2.3}$$

We call the RL method, which approximates value functions using deep neural networks (DNNs) as deep reinforcement learning (DRL).

RL algorithms fall into different types, as Figure 2.2 shows. Below describes the details.

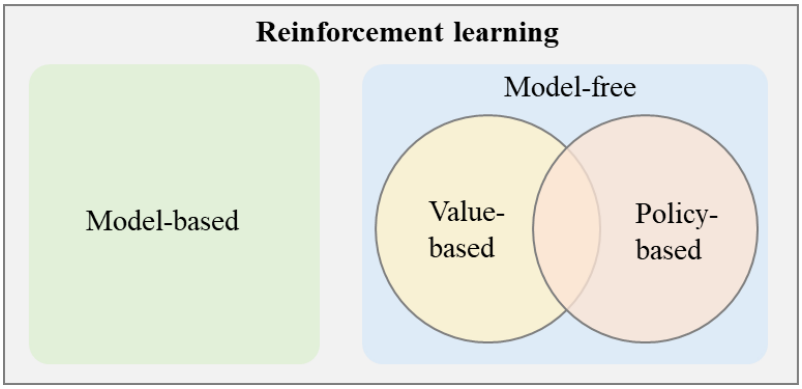


Figure 2.2. RL algorithms.

Model-based and Model-free: Model-based RL algorithms require a model of the environment to estimate the distribution of state transition probability and help the agent learn, while model-free RL algorithms only rely on trial-and-error learning, which is a significant difference from common dynamic programming methods. The model-free RL algorithms can also be classified into two types, i.e., value-based and policy-based algorithms.

Value-based and policy-based: Value-based methods are to learn the state-value or action-value function, while policy-based algorithms are to learn the policy function directly. Hence, it is more like that policy-based methods learn the actor, and value-based methods learn one critic which can evaluate how good the actor is and find the best actor. Table 2.1 lists some classical model-free RL algorithms in recent years.

On-policy and off-policy: Those are different techniques to learn the optimal policy. On-policy methods evaluate and improve the policy used for taking actions, whereas off-policy methods evaluate and improve a policy different from that used to take actions. In other words, off-policy learning has two policies, *target policy* being learned about and more exploratory *behavior policy* used to generate behavior. It facilitates finding the optimal policy, unlike the near-optimal policy from on-policy learning using the same policy to learn and take actions. Almost all value-based methods are off-policy because they allow to use the experience data from other policies to train agents, and they even have the replay buffer to store the experience from different policies and randomly replay them in training process, such as DQN. However, most policy-based methods are on-policy.

Table 2.1. Classical model-free RL algorithms.

Algorithms	Descriptions	Types	Policy
SARSA (Rummery and Niranjan, 1994)	State–Action–Reward–State–Action	Value-based	On-policy
DQN (Mnih et al., 2015)	Deep Q network	Value-based	Off-policy
D3QN (Wang et al., 2016)	Double dueling deep Q network	Value-based	Off-policy
REINFORCE (Williams, 1992)	Monte-Carlo Policy-Gradient Control	Policy-based	On-policy
A2C (Sutton et al., 1999)	Advantage Actor-Critic	Value-based and Policy-based	On-policy
DDPG (Timothy et al., 2019)	Deep Deterministic Policy Gradient	Value-based and Policy-based	Off-policy

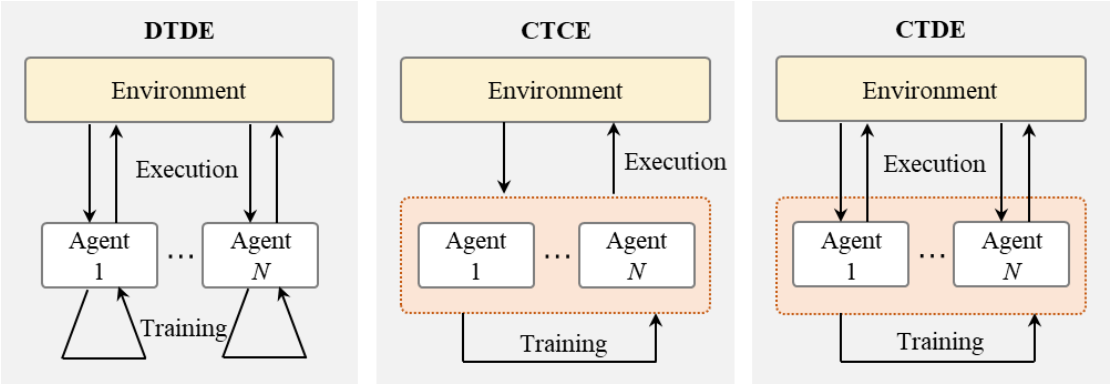


Figure 2.3. Training and execution schemes of MARL.

(2) Multi-agent RL

Muti-agent reinforcement learning (MARL) is an extension of the single-agent RL for more complex environments. The key to MARL is the relationship between agents during the training and execution process. Current MARL methods can be divided into three types from the perspective of the training and execution schemes (Gronauer and Diepold, 2021) as Figure 2.3 shows: 1) distributed training decentralized execution (DTDE) (Tan, 1993), 2) centralized training centralized execution (CTCE) (Gupta et al., 2017), and 3) centralized training decentralized execution (CTDE) (Rashid et al., 2018; Sunehag et al., 2018). DTDE treats all agents independently, and agents' learning does not rely on any shared knowledge; on the contrary, CTCE requires communications between different agents

and integrates all agents as a joint agent, which trains and executes together considering mutual conditions. Hence, the flaw of DTDE is that the environment is non-stationary from the perspective of a single agent with partial observability, reducing learning efficiency. CTCE also suffers the drawback of the curse of dimensionality and heavy reliance on communication infrastructures.

In contrast, CTDE uses mutual information during training but allows independent decision-making for agents during execution. More specifically for value-based methods, CTDE incorporates extra information such as observations, joint actions, and agents' policies to the centralized action-value function during learning. The action value (Q-value) estimates the goodness (expected return) of taking one action according to a policy given a specific state. In online execution, agents can behave independently yet cooperatively without the centralized action-value function for information sharing, facilitating a decentralized implementation. The shared information during training can help agents to overcome the environmental non-stationarity caused by multiple varying policies of agents to improve learning performance. In one centralized Q function, poor agents are likely to be eclipsed by those well-performed agents. Like multi-person collaboration, lazy people may be hidden among hardworking people because there is only one overall evaluation for the task. Therefore, researchers of the CTDE method are devoted to proposing different formulations of the centralized value function to properly decompose the gains resulting from each agent and tackle the credit assignment problems.

2.4 RL-based traffic signal control

In recent years, significant success has been achieved in traffic signal control (TSC) by artificial intelligent techniques, especially RL methods from early tabular Q-learning algorithm to current DRL approaches (El-Tantawy et al., 2014; Genders and Razavi, 2019; La and Bhatnagar, 2011; Liang et al., 2019; Rasheed et al., 2020; Wang et al., 2021). According to the concept of RL, in our TSC field, the environment is the road network, and the agent is usually the signal controller. The agent will perceive traffic conditions and learn to determine the traffic signals that maximize total network performance by training from numerous experienced transitions. Existing research results show that significant advantages of RL methods on TSC problem over other approaches include below two aspects (Genders and Razavi, 2019):

(1) The TSC methods commonly used in the past require models to establish the relationship between signal timing and other traffic states to represent reality, like Australian SCATS and English SCOOTS systems. In contrast, well-trained model-free RL methods are more robust to dynamic traffic states as it learns a policy directly from interactions with the environment, not relying on explicit models for dynamic or transition probabilities. It just needs sample sequences of states, actions, and rewards from actual or simulated interaction with an environment to train. (2) RL leverages the structure of the problem and the information obtained from the interactions between the agent and the environment to learn which actions in specific states lead to rewards. This is in contrast to evolutionary methods, which do not explicitly consider such individual environment interactions but rely on genetic operators and fitness evaluation for evolution.

(1) RL-based TSP

Similarly, the RL algorithm can also be one preferred approach to realize a more efficient TSP. This method builds the interaction framework between the environment and agents rather than analyzing different situations and their optimization programs. The agent senses the environment state, including information about buses, private cars, and signal timing, and chooses the action accordingly.

The environment conducts the action, modifies the traffic signal, and obtains the next state and immediate reward for the agent. Using experiences about states, actions, and rewards, the agent learns to choose the action which brings the largest cumulative reward given a state. Therefore, the model-free RL approach benefits from robustly adapting to dynamic traffic states without assumptions on the optimization model.

The first work of RL for TSP is by Ling and Shalaby (2003), which realized adaptive TSP for the single intersection along the King Streetcar route through a tabular Q-learning algorithm to minimize the headway deviation. However, the studied case with a single streetcar route is somewhat simplified than the real traffic with numerous intertwined bus routes at intersections. Later DRL algorithms also were used to propose TSP strategies for states with high dimensionalities, like DQN (Alizadeh Shabestray and Abdulhai, 2019) and proximal policy optimization with model-based acceleration (PPOMA) (Guo and Wang, 2021). They can extract critical features from the raw state information by DNNs, so it is not prerequisite to provide completed data, such as ETA.

Although the above DRL methods show good performance on traffic efficiency, they have limitations: 1) Both works just considered limited bus/tram routes, which is far simple than actual traffic conditions of cities with high bus ratios. 2) Both works used variable phase sequences, but no mechanism was designed to consider the rationality for phase skipping. Hence, irrational actions may appear in model-free DRL and would cause damage in the real environment, such as two continuous skips for one phase and extremely disordered phase sequence, which may lead to driver anxiety and confusion, respectively.

(2) MARL-based TSC

A range of RL algorithms have achieved great success in traffic signal control at isolated intersections (Alegre et al., 2022; Han et al., 2021; Mao et al., 2022; Wan and Hwang, 2018), but those single-agent RL methods are not scalable to the large network with an extremely high dimension of states and actions (La and Bhatnagar, 2011). Hence, Multi-agent RL methods (MARL) have also been applied to arterial roads or larger networks, treating each intersection (Chu et al., 2020) or a combination of several intersections (Wang et al., 2021) as one agent.

As aforementioned, MARL with CTDE scheme performs better than other MARL due to without environment stationary and curse of dimensionality. Therefore, below summarizes the related works of MARL-based TSC under the CTDE scheme on assigning credits to different agents. Tan et al. (2020) used dense layers to approximate the global Q-value based on other agents' Q-values, which is an extension of the Value Decomposition Network (Sunehag et al., 2018) to decompose the total Q-value as the sum of individual Q-value. Similarly, Zhang et al. (2021) considered the contributions of local agents to global traffic and then formulated the total Q-value as the contribution-weighted sum of other agents' Q-value. Ma and Wu (2022) deployed the QMIX method (Rashid et al., 2018), which added one mixing network taking individual Q-value and global states as input to generate the total Q-value.

Although various works show the effectiveness of MARL approaches on traffic signal control, neither considers the priority strategy for transits. Hence, it is one opportunity to design real-time cooperative TSP strategies using MARL to improve bus reliability.

2.5 Transit reliability

Reliability is a crucial attribute in evaluating the quality of transit service and determining the travel experience of transit users. A basic definition of bus reliability is the stability of the transit

performance over time, such as travel time, waiting time, boarding time, seat availability, and so on (Turnquist, 1981). Hence, it is directly related to the attractiveness of public transit to current and potential riders (Chen et al., 2009). Besides transit users, transit reliability is also significant to transit operators as operational costs are tied to transit service levels.

Schedule adherence is one of the essential elements for the reliability of transit service. It is also an important indicator to characterize on-time performance. The early or late arrival of buses would result in transit users missing the bus or increasing the waiting time, influencing transferring if any, and causing uncertainty of arrival time at destination. And for bus operators, the failure to meet on-time targets may result in penalties or reduced compensation. Therefore, many existing studies propose various priority measures to decrease the bus schedule delay, which is the delay time compared to the bus schedule. TSP is one of the most effective priority measures, as mentioned in *Section 2.1 Transit signal priority* (Chow et al., 2017; Liang et al., 2022).

Another element significantly affecting bus reliability is headway adherence. Transit users will focus more on schedule adherence when bus schedules are rare, while they will focus more on headway adherence when bus schedules are frequent. The variability of traffic states and passenger demand always leads to headway instability. Once the headway between the target bus and the bus in front increases, there will be more people waiting at bus stations, and then the longer boarding time caused by increasing waiting passengers would increase bus dwell time and enlarge the headway more. Then the headway between the target bus and the bus behind decreases, and the fewer waiting persons and less boarding time would cause the bus behind run faster and finally catch on the target bus after a period of time. This phenomenon is well-known as bus bunching. In this situation, more transit users experience crowded buses and wait for a long time, deteriorating travel comfort.

To better understand the effects of headway adherence on the person waiting phenomenon, the arrival and departure of persons at a stop are shown in Figure 2.4 and Figure 2.5 under late arrival and early arrival. Schedule headway is always set evenly, but actual headway can be uneven due to many factors, such as traffic conditions and passenger demand. In Figure 2.4 (a), the black line denotes person arrivals, and the red and blue lines represent actual and expected person departure, respectively. One bus arrives late after time t', so there are more arrived person waiting for this late bus. Blue area is the cumulative waiting time when one bus comes late, while the red area is the cumulative waiting time when buses arrive on schedule. The first blue triangle area after time t' is much bigger than the red one, showing a much larger waiting time for persons. If there is a passenger capacity control for buses like in Figure 2.4 (b), the first bus after time t' does not have enough capacity for all waiting persons, then the rest who fail to board are required to wait for the second bus. Assume that the second bus has enough capacity for all the waiting people when it arrives at the stop. Compared to the blue area of Figure 2.4 (a) and (b), we can find that capacity control cause waiting time worse, and the dark blue area of Figure 2.4 (b) is the increased waiting time under capacity control compared to no capacity control. Fewer people arrive for the early arrival situation, so it is not easy to overpass bus capacity, which is simpler than late arrival situation. Figure 2.5 shows the case that the first bus after time t' arrives early but following buses arrive following the scheduled headway, so the blue area is small. However, if the first bus arrives early, it is more likely that the following bus comes with a large headway (larger than scheduled headway), which will cause a situation like Figure 2.4 (a). Then the cumulative waiting time would increase a lot.

Therefore, nonadherence to headway has a significant negative influence. To even out headway and also solve bus bunching problems, a variety of strategies have been developed, i.e., bus holding

strategy (Daganzo, 2009), speed control (Daganzo and Pilachowski, 2011), transit signal priority (Anderson and Daganzo, 2020; Chow et al., 2017; Liang et al., 2022), etc. As their name suggests, the bus holding strategy is to hold early arrival buses at stations, and speed control uses an adaptive control scheme to accelerate the late buses and decelerate the early buses. The former mainly works for decreasing headway to delay early arrival buses, and other methods for increasing headway, like stop-skipping and bus insertion, would deteriorate the experience of waiting people and increase costs of bus operators as it requires many spare buses and spaces for placing those buses. Moreover, speed control must always trace and control all buses, which is more complicated than TSP strategies. TSP strategies are usually used to decrease bus delay and improve schedule adherence; however, they are also effective for headway adherence, e.g., prioritizing buses arriving late with large headway and not prioritizing buses arriving early with small headway.

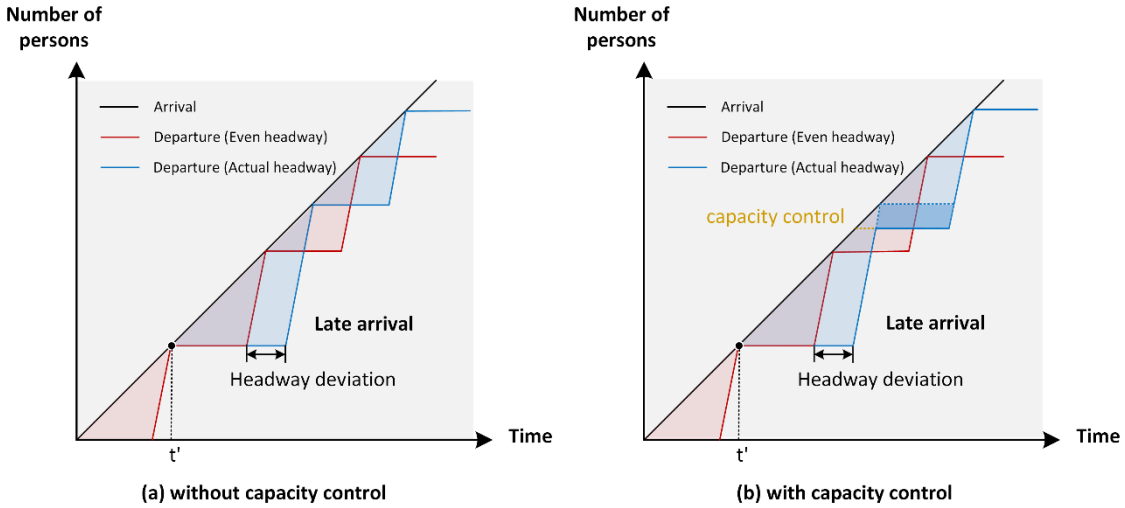


Figure 2.4. The cumulative person waiting time when the bus arrives late.

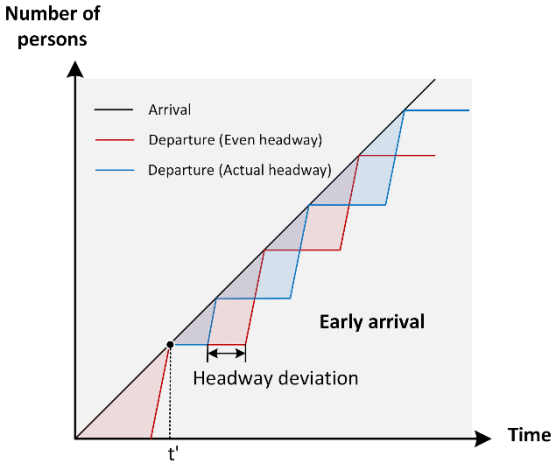


Figure 2.5. The cumulative person waiting time when the bus arrives early.

Chapter 3

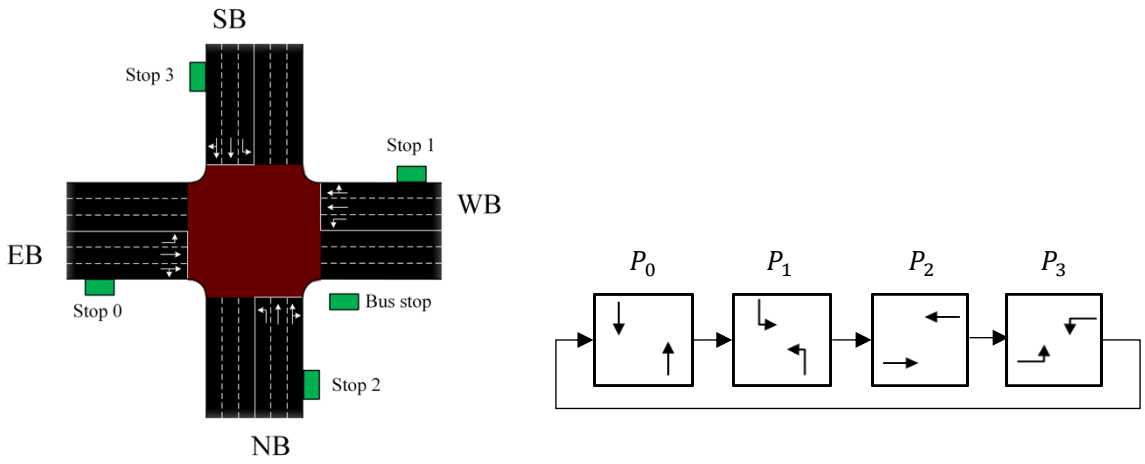
TSP at isolated intersection

This chapter proposes a TSP strategy at the isolated intersection based on the DRL framework in a connected environment to improve bus efficiency and reduce the person delay of buses. This chapter is orgainized as follows. Section 3.1 introduces the methodology of this chapter and describes the proposed eD3QNI algorithm. The experiment framework are presented in Section 3.2. Experiment results are showed in Section 3.3. Section 3.4 discusses the performance of the proposed approach. Section 3.5 summarizes this chapter.

3.1 Methodology

3.1.1 Problem statement

RL is a promising alternative approach to determine the optimal relationships between actions and their cumulative effects (rewards) on the given traffic environment (states). Thus, it can adapt to complicated changes in traffic and make the best traffic control actions (Rasheed et al., 2020). TSP process can be defined as MDP by state, action, reward and unobservable state transition probability. The TSP strategy of a typical four-leg signalized intersection with four bus stops is studied. Figure 3.1 shows the intersection layout and phase configuration.



(a) Intersection layout (b) Phase sequence

Figure 3.1. The studied environment.

Twelve vehicle movements and bus routes are considered, i.e., right-turn, through, left-turn right-hand traffic of eastbound (EB), westbound (WB), southbound (SB), and northbound (NB) approaches. A standard 4-phase design is adopted which is applied in many countries, e.g., Australia. The adoption

of this phasing scheme hides a lane setup requirement that there be separate lanes for left turns. The problem is formulated based on concepts from MDP, and the intersection is treated as the agent to control the signal timing. More specifically, the state s_t, the action a_t, and the reward r_t are required to be defined at discrete time step t, $t = 0,1,2,3$ ...

(1) State

To realize the TSP strategy, sufficient state information should be transmitted to the agent, including the information about the whole traffic and the bus movements. The first one shows the entire traffic conditions at the signalized intersection, and the second bus information provides the current state of the bus.

a. *The whole traffic*

This part contains three elements, i.e., average speed S_A and queue length L_Q of approach lanes, and current phase P_C. Both average speed and queue length can be measured or estimated by conventional detection technologies such as loop detectors and cameras at intersections, and the CV technology is not needed.

As there are four approach links, each of which is composed of three lanes, the matrix S_A can be expressed as:

$$S_A = \begin{bmatrix} S_{A11} & S_{A12} & S_{A13} \\ S_{A21} & S_{A22} & S_{A23} \\ S_{A31} & S_{A32} & S_{A33} \\ S_{A41} & S_{A42} & S_{A43} \end{bmatrix} \tag{3.1}$$

where $S_{Al\varphi}$ represents the average speed on the lane φ of the approach link l; $\varphi \in \Phi_l$, and Φ_l is the lane set of approach link l; $l \in \Psi$, and Ψ is the approach set. In this studied 4-leg intersection, $l = 1,2,3,4$; $\varphi = 1,2,3$. The actual form of the above matrix can vary depending on real lane settings.

The same as the matrix S_A, matrix L_Q can be expressed as:

$$L_Q = \begin{bmatrix} L_{Q11} & L_{Q12} & L_{Q13} \\ L_{Q21} & L_{Q22} & L_{Q23} \\ L_{Q31} & L_{Q32} & L_{Q33} \\ L_{Q41} & L_{Q42} & L_{Q43} \end{bmatrix} \tag{3.2}$$

where $L_{Ql\varphi}$ represents the queue length on the lane φ ($\varphi \in \Phi_l$) of the approach link l ($l \in \Psi$).

Figure 3.1(b) shows the phase combinations, and 4 phases are considered. For better processing in the neural network, phases are encoded by one-hot encoding (Zhou et al., 2020). That is P_p ($p = 0,1,2,3$), where $P_p = 1$ if phase p shows green light, otherwise $P_p = 0$. Therefore, the current phase P_C is given by the below vector:

$$P_C = [P_0 \quad P_1 \quad P_2 \quad P_3]^T \tag{3.3}$$

b. *The bus movements*

Three indicators are selected to represent the states of buses, i.e., distance to the stop line D, bus schedule delay SD, and bus occupancy O. D can be detected by the camera, loop detectors, or calculated by GPS information; the latter two information can be obtained from the bus operator, who needs V2I Onboard Units of buses to transmit the location captured by GPS and the occupancy collected by APC to them. Only the approaching buses within 100 m from the stop line will be recorded.

Distance to the stop line D can be easily calculated based on the current location of the approaching bus and the location of the stop line to show the distance from the ego bus to the stop line of its approach.

Bus schedule delay SD reflects the delay of a bus as compared against its schedule. Bus schedule deviation SDV_{ϖ} can be obtained based on the GPS when the buses arrive at bus stop ϖ. And the schedule delay of a bus that is driving between two stops can be calculated by

$$SD = \max(0, SDV_{\varpi} + SD_t) \tag{3.4}$$

where SD_t is the schedule delay from the nearest stop ϖ passed by the bus to the current location at simulation step t, which is formulated as $SD_t = TT_t - \frac{TD_t}{V_s}$. TT_t and TD_t are the travel time and distance from stop ϖ until simulation step t, respectively. V_s is the bus schedule speed, set to be 4.0 m/s. If buses have not arrived at any stops in the experiment, we will regard the start point of the network as a stop. SDV_{ϖ} will be the given initial schedule deviation of this bus and SD_t is calculated from entering the network to the current location at step t. It should be noted that when the bus is not late than its schedule, that is $(SDV_s + SD_t)$ is a negative value, SD should be 0. This is because we set that only late buses need to consider different priorities based on the corresponding state. Thus, all the specific early arrival time of non-late buses are neglected.

Bus occupancy O should be retrieved from the bus company by APC. If the exact number of passengers is not easy to obtain, we can just set several levels to represent the bus occupancy based on the e-payment record or in-car camera.

These three indicators are formed as a matrix S_{BUS} to represent the states of buses. In order to fix the size of states for neural network training, the matrix is designated to be 12×3 (as shown in Table 3.1), in which three buses' information will be considered for the movements of each phase.

Table 3.1. The matrix of buses' states.

	Distance D	Schedule delay SD	Occupancy O
P_0			
P_1			
P_2			
P_3			

The buses of north-south through (P_0), north-south left-turn (P_1), west-east through (P_2), and west-east left-turn (P_3) are recorded in the first, second, third, and last three rows of the matrix. If there are less than three buses detected at approaches of one phase flow, the blank space in the matrix would be 0. If there are more than three buses detected at approaches of one phase flow, calculate the indicator $prio = SD * O/(D + \varepsilon)$ where $\varepsilon = 10^{-5}$ to avoid the case that the denominator becomes 0. Shorter distance to stop line, larger schedule delay, and larger occupancy, respectively, will result in higher value of $prio$. Hence, information of three buses with the largest 3 $prio$ values will be filled into the matrix for each phase flow. Moreover, we emphasize that information of buses at all the 4 phases are contained in the state, rather than only the bus whose route is served by the current phase.

In summary, the state s_t is defined as $s_t \triangleq (S_A, L_Q, P_C, S_{Bus})$.

(2) Action

This study considers two schemes: 1) the fixed sequence (FS) scheme without phase-skipping, 2) the variable phase (VP) scheme with phase-skipping. At each time step, the agent will select one action a_t based on the current state s_t to receive a new state s_{t+1}.

For the FS scheme, the action space Å = (0,1). $a_t = 0$ when keeping the current phase; $a_t = 1$ when switching to the next phase. For the VP scheme, the action space Å = (0,1,2,3). That is, a_t is the phase of the next step. If the next phase turns out to be the same as the current phase, keep the current phase and run one more step, else run out of the yellow time and switch to the next phase.

(3) Reward

The reward function is the key for the agent to learn to take the best action gradually. It is also one element that differentiates RL methods from other types of machine learning approaches. The agent will seek to obtain a policy that maximizes the expected return. In this experiment, the expected return can be derived from Eq (2.1) to $G_t = E\left[\sum_{\tilde{t}=t}^{T} \gamma^{\tilde{t}-t} r_{\tilde{t}}\right]$ where T is the time step at which one episode terminates (Mnih et al., 2013). The goal of each agent is to give late bus priorities at the intersection to improve the efficiency of buses to a good performance level by minimizing the person delay of buses at the intersection. Therefore, we define the reward as:

$$r_t = -\sum_k (\hat{d}_{kt} * O_{kt}) / \sum_k O_{kt} \tag{3.5}$$

where k is the index of the bus in the whole intersection; O_{kt} is the occupancy of bus k at time t; and $\hat{d}_{kt}$ is the good performance delay of bus k at time t, which means the delay compared to good performance, formulated as $\hat{d}_{kt} = TT_{kt} - \frac{TD_{kt}}{GV_{Bus}}$; TT_{kt} and TD_{kt} are travel time and travel distance of bus k at time t; GV_{Bus} is the good performance velocity of bus, set to be 5.56 m/s. Note that all the buses in this intersection will be considered.

Here good performance delay $\hat{d}_{kt}$ is used to formulate the reward functions, rather than the normal delay d_k, given by $d_{kt} = TT_{kt} - \frac{TD_{kt}}{DV_{Bus}}$ where DV_{Bus} is the desired velocity of the bus, set to be 16.67 m/s. The reason is that bus generally cannot run at desired speed due to the traffic situation and dwell time, so more positive immediate rewards can be collected by $\hat{d}_{kt}$, and more meaningful samples are saved to let agents learn better.

In addition, two alternative reward functions designed for other operational goals are discussed in *Section 3.4.2 Performance of different reward functions*. Considering this study's purpose and practical feasibility, they are not introduced here.

(4) Basic constraints

We should intervene partly in the interaction between the agent and the environment to meet the restrictions of traffic safety, efficiency, and comfort. According to traffic engineering knowledge, two types of constraints are required to be satisfied in the RL structure, i.e., the constraints of minimum and maximum green time and the phase skipping rule.

a. The constraints of minimum and maximum green time

The green time should satisfy the minimum green time $Gmin$ to guarantee the safety of pedestrians to cross an intersection. It should also not exceed the maximum green time $Gmax$ to avoid too long

waiting for other approaches' movements.

b. The constraints for the phase skipping rule

This part only applies to the VP scheme. Because it is undesired to continuously skip a certain phase multiple times or skip the phase with a long queue, which will lead to drivers' frustration, we can set the phase skipping rule as:

- When the queue length of one phase flow exceeds the predefined threshold, this phase cannot be skipped.
- Each phase cannot be skipped twice in a row if there are vehicles detected in its corresponding approach lane.

Such a phase skipping rule is designed to avoid inappropriate skipping and provide a valid action space for the agent to choose the action from.

(5) Evaluation metrics

Person-based metrics are widely used to evaluate the impacts of control strategies on travellers, like the person delay and person lateness. Road-based metrics also can describe the whole traffic situation, such as the queue length. Hence, we can use the following metrics to assess the performance of TSP strategies, including person-based metrics and road-based metrics.

a. Average person delay of buses (APDB)

$$APDB = \sum_t \sum_k (d_{kt} * O_{kt}) / \sum_t \sum_k O_{kt} \tag{3.6}$$

b. Average person delay of cars (APDC)

$$APDC = \sum_t \sum_c (d_{ct} * NP_{ct}) / \sum_t \sum_c NP_{ct} \tag{3.7}$$

where d_{ct} is the normal delay of car c at time t, given by $d_{ct} = TT_{ct} - \frac{TD_{ct}}{DV_{Car}}$, DV_{Car} is the desired velocity of cars, set to be 16.67 m/s; NP_c is the number of passengers in the car c.

c. Average person delay (APD)

$$APD = \sum_t (\sum_k d_{kt} * O_{kt} + \sum_c d_{ct} * NP_{ct}) / \sum_t (\sum_k O_{kt} + \sum_c NP_{ct}) \tag{3.8}$$

where b and c is the bus and car index, respectively.

d. Queue

$$Queue = \frac{\sum_t \sum_l \sum_\varphi L_{Ql\varphi t}}{T * \sum_l |\Phi_l|} \tag{3.9}$$

where $L_{Ql\varphi t}$ is the queue length on the lane φ ($\varphi \in \Phi_l$) of the approach link l ($l \in \Psi$) at time t ($t \in [0, \mathrm{T}]$). $|\Phi_l|$ is the size of the approach set, e.g., number of lanes of the approach link l.

e. Lateness

The lateness means the average time incurred by waiting persons due to bus schedule delays. It is formulated as

$$Lateness = \sum_t \sum_k (SD_{kt} * O_{kt}) / \sum_t \sum_k O_{kt} \tag{3.10}$$

where SD_{kt} and O_{kt} are the schedule delay and passenger occupancy of bus k at time t.

3.1.2 eD3QNI algorithm

The proposed DRL for TSP strategy is an *extended Dueling double deep Q-learning with Invalid action masking* (eD3QNI) algorithm, which uses prioritized experience replay (PER) to sample batches, and ϵ -greedy method to explore experiences. D3QN has the advantages of high efficiency of samples usage and good convergence performance. Given that the reward function is formulated as the negative value of delay, many negative immediate rewards would be received within one period, but the agent can not learn useful knowledge from experiences with very negative rewards. PER can replay more essential transitions, such as samples with a positive reward, thereby helping the agent learn more efficiently (Schaul et al., 2016).

The eD3QNI's framework, as displayed in Figure 3.2, is the same as the typical framework of D3QN. First, it randomly initializes the phase for the environment, and after a duration of warm-up time, the initial state is received from the environment. The agent receives the state and chooses the best action. Then the environment conducts this action and obtains the next state and the associated reward. The tuple composed of the state, the action, the reward, and the next state is stored in replay memory. Its TD-error, computed as the priority term, and its index are stored into the sum-tree (Schaul et al., 2016). Sum-tree is a binary tree data structure where the value of a node is the sum value of its left and right children, and it can be used to conduct proportional prioritization. When the number of the tuples reaches the batch size, one mini-batch will be sampled by sum-tree to train the main network. After the training, new TD-errors are obtained, and the priority in the sum-tree should be updated. Parameters of the target network will be updated every certain number of learning steps.

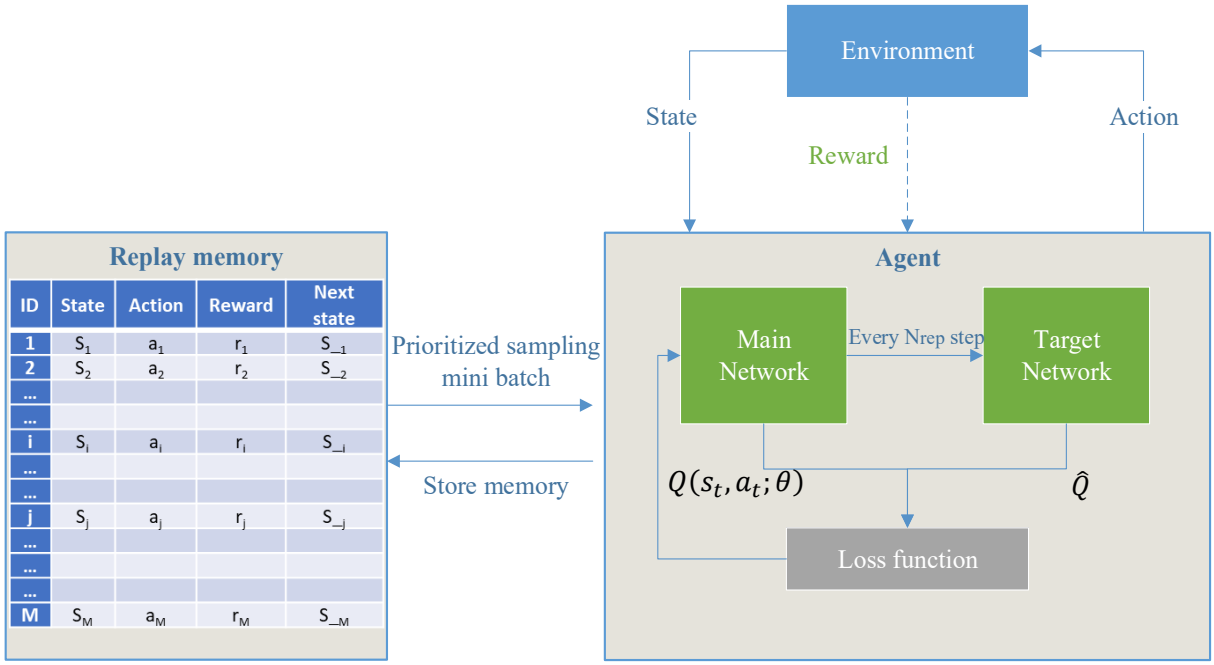


Figure 3.2. The framework of eD3QNI.

Figure 3.3 shows the architecture of DNN of the above eD3QNI, taking the VP scheme as an example. The state information obtained from the environment is input to this DNN to extract features firstly. After concatenation, the features will be input into two fully connected layers to get the state-value and the advantages for each action (Wang et al., 2016). Finally, the action-value for all actions is output from this network by summarizing those two terms. This value is the expected cumulative reward received when taking the corresponding action based on the given state. The only difference between

the DNN of the FS and VP schemes is the output size of the advantages and the final action-value, i.e., 2 for the FS scheme and 4 for the VP scheme. LeakyReLU is a type of activation function based on the ReLU, but it has a small slope for negative values, rather than making them equal to zero. Its use introduces the nonlinear characteristics into DNN.

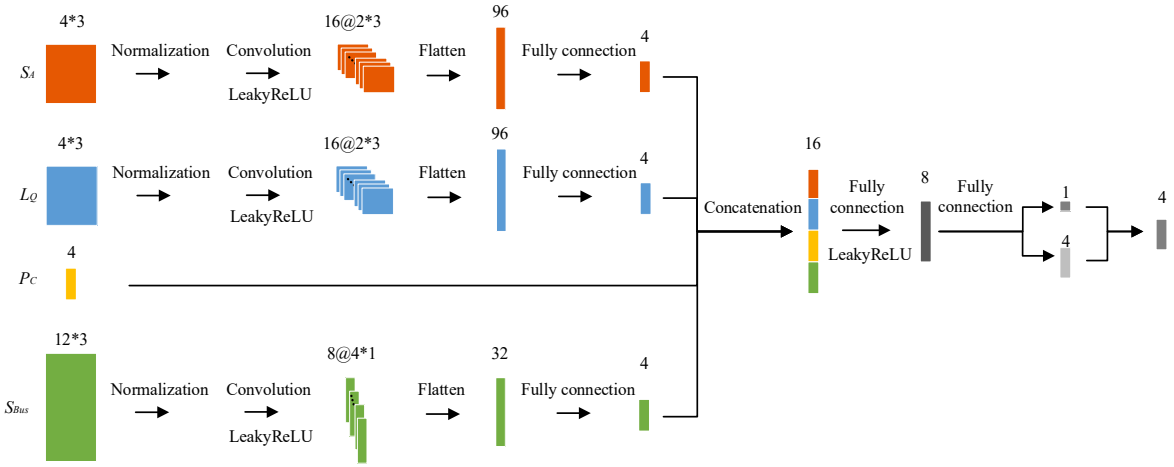


Figure 3.3. The architecture of DNN of the eD3QNI agent for the VP scheme.

As for the basic constraints, two methods can be used, i.e., variable decision point (VDP) and invalid action masking (IAM).

(1) Variable decision point (VDP)

This approach has been employed in several existing studies of RL-based TSP strategies (El-Tantawy et al., 2014; Kai et al., 2019; Wan and Hwang, 2018) for the consideration of the minimum green time. The agent only selects actions at decision points instead of at every time step. For each phase, the decision point is between its minimum and maximum green times. Figure 3.4 shows one example. In this example, the time between two actions is variable. When the action is keeping the current phase and the green time duration satisfies the minimum and maximum green times, the environment extends the current phase by one time step, of which step length $\Delta t_\lambda = 1$s; when the action is switching to another phase or is keeping the current phase but the green time duration exceeds the maximum green time, then the environment should go through the yellow time Y and G_{min_p} at this step, thus $\Delta t_\lambda = Y + G_{min_p}$.

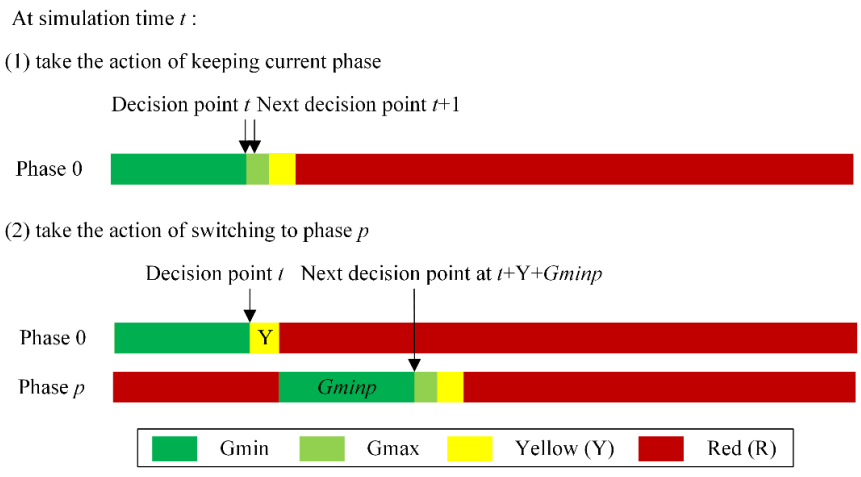


Figure 3.4. The decision points for the agent.

(2) Invalid action masking (IAM)

IAM is an approach to mask the invalid actions sampled from the entire action space. It has been proven to work well in Policy Gradient method, even in a situation with large-space invalid actions (Huang and Ontañón, 2020). In this work, we adopted IAM to handle the constraints for green time and phase skipping rule, masking the invalid actions during the acting and training process.

For identifying the invalid actions, the states will add one more tuple $S_{mask} \triangleq (G_{min}, G_{max}, Duration, SkipCntr)$ as the states for masking, which shows the minimum and maximum green time, the current duration of this phase, and the skipped times of each phase. Hence $s_t \triangleq (S_A, L_Q, P_C, S_{Bus}, S_{mask})$. G_{min}, G_{max}, and $Duration$ is one integer value individually, and $SkipCntr$ is a 4×1 matrix only filled with the number of 0 or 1. Invalid actions are judged by the following principles:

- When the action does not satisfy the minimum and maximum green time constraint, it is regarded as an invalid action.
- When the queue length of one phase flow is not less than the threshold $\boldsymbol{Q_{threshold}}$, phases after this phase in a signal cycle will be the invalid actions. This signal cycle refers to the cycle from the current phase to the next 3 phases in this study. For example, if the current phase is phase 1, the signal cycle will be phases 1-2-3-0. If the queue length of phase 2 exceeds $\boldsymbol{Q_{threshold}}$, phase 3 and 0 will be invalid actions, and the action space only includes phases 1 and 2. The agent will determine keeping the current phase 1 or switching to phase 2 with a long queue length directly.
- When there is one phase flow with queue less than $\boldsymbol{Q_{threshold}}$, and has been skipped once in signal control, phases after this phase in a signal cycle will be the invalid actions to guarantee the phase not be skipped the second time. For example, if the current phase is phase 0, i.e., the signal cycle will be phase 0-1-2-3, and phase 3 has been skipped once, the action space will include phase 0 to 3 since there is no phase after phase 3 in this cycle.

For masking actions, we add a large negative value $Value\_$ into the logits of invalid actions outputted from the training network of the PG method. Similarly, the considerable negative value $Value\_$ can be added into the Q values of invalid actions outputted from eD3QN (see **Pseudocode of IAM)**. In this case, the step length is also variable as step length $\Delta t_\lambda = 1$ when the action is keeping current phase, and $\Delta t_\lambda = Y + 1$ when the action is switching to another phase.

The whole process of the proposed method can be explained with its pseudocode (see **Algorithm eD3QNI**). The ϵ of ϵ-greedy method is designed to decay exponentially with the simulation steps.

3.2 Experiment

3.2.1 Simulated environment

(1) Simulation platform

The experiment is built in a widely used simulation platform, SUMO (Simulation of Urban MObility) (Krajzewicz et al., 2002). It is open-source software with abundant packages to efficiently conduct large-scale microscopic traffic simulation, developed mainly by the Institute of Transportation Systems of German Aerospace Center. In this software, an API (Application Programming Interface),

named TraCI (Traffic Control Interface), allows users to manipulate the traffic simulation online and retrieve various values of objects in the simulation.

Pseudocode of IAM
Given $Q_{threshold}$, $Value\_$, $(s_\lambda, a_\lambda, R_\lambda, \Delta t_\lambda)$
$s_k \triangleq (S_A, L_Q, P_C, S_{Bus}, S_{mask})$, $S_{mask} \triangleq (G_{min}, G_{max}, Duration, SkipCntr)$
Initialize 1×4 matrix M_λ with all zeros to represent mask values of all phases
Calculate average queue length of each phase AL_Q based on L_Q
if $Duration < G_{min}$ **then**
$M_\lambda[j] = Value\_$ for $j \in$ {the phase-switching actions}
elif $Duration \geq G_{max}$ **then**
$M_\lambda[j] = Value\_$ for $j \in$ {the phase-keeping actions}
end if

if Variable Phase scheme with phase skipping **then**
for phase p = 0 **to** 3 **do**
if $0<AL_{Q_p}<Q_{threshold}$ **then**
if $SkipCntr_p$=1 **then**
$M_\lambda[j] = Value\_$ for $j \in$ {the phase-skipping actions}
end if
else $AL_{Q_p} \geq Q_{threshold}$ **then**
$M_\lambda[j] = Value\_$ for $j \in$ {the phase-skipping actions}
end if
end for
end if
Obtaining the masked Q values $Q^{masked} = M_\lambda + Q$
choose action based on $a_\lambda = argmax_a Q^{masked}(s_\lambda, a_\lambda; \theta)$

(2) Environment layout

The simulated intersection is 301 m × 301 m area, with four links. Each link has one right-turn and through lane, one through lane, and one left-turn lane, of which the width is 3.5 m. And there have been set 10-meters-long bus stops at the most-right lane of each link, 100 m far from the stop line. Both twelve car movements and bus movements are set in this simulation from different approaches. Lane area detectors (E2) are placed in every approach lane with 150-meters in length.

(3) Traffic demands

Two traffic demands (shown in Table 3.2) are considered, of which the summation of critical flow ratios at this intersection are about 0.6 and 0.8 to represent normal and high traffic demand, respectively. The flow ratio means *flow volume*/*saturation flow rate*. The probability method is used to generate the traffic flow.

Table 3.2. Traffic demands in simulation.

Demand	Vehicle type	North-south (veh/h)			West-east (veh/h)		
		Through	Right-turn	Left-turn	Through	Right-turn	Left-turn
Normal	Car	270	38	150	450	48	230
	Bus	30	12	30	30	12	30
High	Car	340	36	180	580	46	290
	Bus	60	24	60	60	24	60

(4) Traffic signal

The phase considered in the simulation is the same as the example of Figure 3.1(b). The phase of

each step, excluding the warm-up period, is decided by the action from the proposed algorithm. By webster's method, we can get the fixed-time signal (FT) (P_0, P_1, P_2, P_3) for the two demands are 9 s, 10 s, 15 s, 14 s, and 21 s, 22 s, 33 s, 32 s respectively, with 3 s yellow time after each phase. The minimum green time $Gmin$ for each phase is calculated by D_{Cross}/V_{Ped}. As the refuge islands are set in the middle of crosswalks, the crosswalk width D_{Cross} can be 3*3.5=10.5 m. Considering pedestrian walking speed V_{Ped}=1.2 m/s, $Gmin$=10.5/1.2=9 s for each phase. The maximum green time for each phase is set to be 10 s more than FT, i.e., 19 s, 20 s, 25 s, 24 s for normal demand and 31 s, 32 s, 43 s, and 42 s for high demand.

Algorithm eD3QNI

Input: Discount factor γ, maximum epsilon ϵ_{max}, minimum epsilon ϵ_{min}, epsilon decrement ϵ_{dec}, learning rate α, replay memory size S_{rep}, batch size S_{bat}, the number of iterations to replace parameters N_{rep}, the number of episodes *N*, simulation time *T*, simulation warm-up time *Twarm*

Initialize replay memory Ω with zeros and sum-tree with $p_1 = 1, \Delta = 0$

Initialize main network θ and target network $\theta^- = \theta$

for *episode* = 1 **to** N_e **do**

Initialize environment

for *t* = 1 **to** *T* **do**

if *t* < *Twarm* **then**

run simulation one step, $count$ = 0, λ = 0

return

end if

obtain state s_1 and action a_1, $count$ += 1, λ += 1, $index = cout\%S_{mem}$

epsilon $\epsilon = \epsilon_{max} * e^{-\epsilon_{dec} * count}$ if $\epsilon > \epsilon_{min}$ else ϵ_{min}

Choose action $a_t = argmax_a Q^{masked}(s_k, a_k; \theta)$ with probability $(1 - \epsilon)$ or randomly choose one with probability ϵ

Simulate one step Δt_λ, calculate reward r_λ by $r_\lambda = \sum_{t=1}^{\Delta t_\lambda} \gamma^{t-1} r_t$ and get next state $s_{\lambda+1}$

Calculate TD-error $\delta_\lambda = \hat{Q} - Q(s_\lambda, a_\lambda)$ by

$$Q(s_\lambda, a_\lambda) = V(s_\lambda, a_\lambda) + (A(s_\lambda, a_\lambda) - \frac{1}{|\mathcal{A}|}\sum_{a'} A(s_\lambda, a'))$$

$$\hat{Q} = \begin{cases} r_\lambda & , \sum_\lambda \Delta t_\lambda = T \\ r_\lambda + \gamma^{\Delta t_\lambda} \hat{Q}(s_{\lambda+1}, argmax_{a' \in A} Q^{masked}(s_{\lambda+1}, a'; \theta); \theta^-), otherwise \end{cases}$$

$t = t + \Delta t_\lambda$

if replay memory reaches the size S_{rep} **then**

Delete the oldest memory tuples

end if

Store tuple $(s_\lambda, a_\lambda, r_\lambda, s_{\lambda+1}, \Delta t_\lambda)$ to replay memory and add $(\delta_\lambda, index)$ into the sum-tree

if the number of stored memories > S_{bat} **then**

for *i*=1 **to** S_{bat} **do**

Sample transition $i \backsim P(i) = (p_i + \xi)^\lambda / \sum_j (p_j + \xi)^\lambda$, $\xi = 0.01$, $\lambda = 0.6$

Calculate importance-sampling weight $\omega_i = (S_{mem} * P(i))^{-\mu} / \max \omega_i$, $\beta = \min(1, \mu + \sigma)$, $\mu_0 = 0.4$, $\sigma = 0.001$

Calculate TD-error δ_i and update priority $p_i \leftarrow |\delta_i|$

Accumulate weight changes $\Delta \leftarrow \Delta + \omega_i \delta_i \nabla_\theta Q(s_i, a_i)$

end for

Update weights $\theta \leftarrow \theta + \Delta$ and reset $\Delta = 0$

Update target network $\theta^- \leftarrow \theta$ every N_{rep} step

end if

end for

(5) Bus information

Like car flows, 12 routes for buses are fully implemented in the network, i.e., EB/WB/NB/SB

through/right-turn/left-turn. Bus lines of through and right-turn at EB, WB, NB, SB approach will stop at Stops 0, 1, 2, 3, respectively. All the left-turn bus lines will not stop at any bus stops. However, one total bus flow is specified for each route rather than individually considering the flow of each bus line in one route because several bus lines may share the same route at an intersection in real traffic. As Table 3.2 shows, the bus flow of North-south/West-east through, right-turn, left-turn routes are 30, 12, 30 veh/h under normal demand, and 60, 24, 60 veh/h under high demand. To account for the stochasticity of bus arrivals as a result of a combination of different bus lines that share the same given route, the above bus flows are generated by a binomial distribution with probability of 0.0083, 0.0033, 0.0083 under normal demand, and by a binomial distribution with probability of 0.0167, 0.0067, 0.0167 under high demand. Since buses would have schedule delays in their running, each bus will be given an initial schedule deviation when entering the intersection.

The distances of approaching buses to the stop line are retrieved from E2 detectors. In reality, the bus schedule deviation and occupancy can be received by CV technology. However, in the simulation, the bus occupancy is generated by *randint* syntax from 1 to 70, and the initial bus schedule deviation is generated by multiplying 120 with the 200 values from a standard normal distribution. Different sets of numbers will be generated for various episodes by changing the random seed. In this way, we can generally control the number of passengers and schedule delays in one episode simulation and duplicate the number settings for other methods in the same episode, making it easy to compare the results of different methods.

In the simulation, cars are set to carry 1.2 person per vehicle. After setting the above information, each episode will simulate 1500 s, of which the first 300 s is for warm-up, and normal and high demand simulate for 600 s successively. Numbers are generated randomly to be the SUMO seeds of different episodes.

3.2.2 Compared methods

To evaluate the performance of the proposed method, we compare it with the following baseline approaches. The detailed calculations and algorithms are presented in Appendix A.

(1) Fixed-time signal (FT): The signal timing is pre-set and fixed (see Appendix A-I). Phases are cycled in a fixed sequence, and their durations are calculated by the webster's method in this experiment. It is a commonly used control strategy due to its simplicity.

(2) Active TSP (ATSP): Two kinds of active TSP strategies are modelled as the benchmarks of fixed sequence and variable phase scheme, ATSPF and ATSPV, which can compare with eD3QNI method of fixed sequence and variable phase scheme, respectively (see Appendix A-II). ATSPF considers only green extension and red truncation under the fixed sequence scheme. It is the modification of Thodi et al. (2021) 's work and prioritizes the buses with the larger total person lateness (*Total person lateness = Bus occupancy*Bus schedule delay*). ATSPV prioritizes the phases with the largest sum value of $prio$ and follows phase skipping rule under the variable phase scheme.

(3) eD3QNV, eD3QNI: The eD3QNI and eD3QNV are eD3QN methods with IAM and VDP, respectively. The eD3QNV algorithm can only satisfy the constraints for green time, while the eD3QNI algorithm can satisfy constraints for green times and the phase skipping rule in the variable phase scheme. The eD3QNI can store more experience than eD3QNV due to the selecting actions at each time step rather than certain decision points. These two approaches are analyzed under fixed sequence and variable phase schemes.

(4) REINFORCE: This policy-based approach is a modification of the work of Williams (1992) by adding the IAM in the algorithm (see Appendix A-III). Given that the action space in this research is discrete and not more than 4, the stochastic policy has been taken, meaning actions are chosen based on the probability. In this case, a large negative value $Value\_$ will be added into the logits of invalid actions outputted from Policy Gradient method to mask invalid actions.

(5) Advantage Actor-Critic (A2C): This is another Policy Gradient method with the central aspect of n-step updating, which is the crucial difference from REINFORCE updated until the end of one episode (see Appendix A-IV). We adopt a one-step updating algorithm with IAM here to value its performance.

(6) Deep Deterministic Policy Gradient (DDPG): This algorithm was proposed to realize off-policy learning with continuous action by combing the ideas of DQN and Deep Policy Gradient (DPG) (Timothy et al., 2019) (see Appendix A-V). We also add IAM into it here to implement constraints.

For REINFORCE method, its DNN also firstly extract features, like Figure 3.3, but the concatenated features after LeakyReLU activation will be input to a fully connected layer and output the logits of all actions. Then we can compute the probability of selecting different actions by categorical distributions based on obtained logit values, and choose the action. The other two gradient policy methods have two networks: actor network to select action and critic network to assess that action. Their actor networks are the same as the REINFORCE, while their critic networks will finally output one state-value, instead of the logits value of all actions, by inputting LeakyReLU activated concatenated features into one fully connected layer. The action is selected with a certain probability, which means these methods already can explore. Because the feature extraction part of actor and critic networks is the same, these two networks' feature extraction layers can share the same parameters. Hence, those parameters will be updated cumulatively by the loss of these two networks.

The learning parameters of all models are set in Table 3.3. These parameters have been tried in many combinations to get the best one.

Table 3.3. Model parameters.

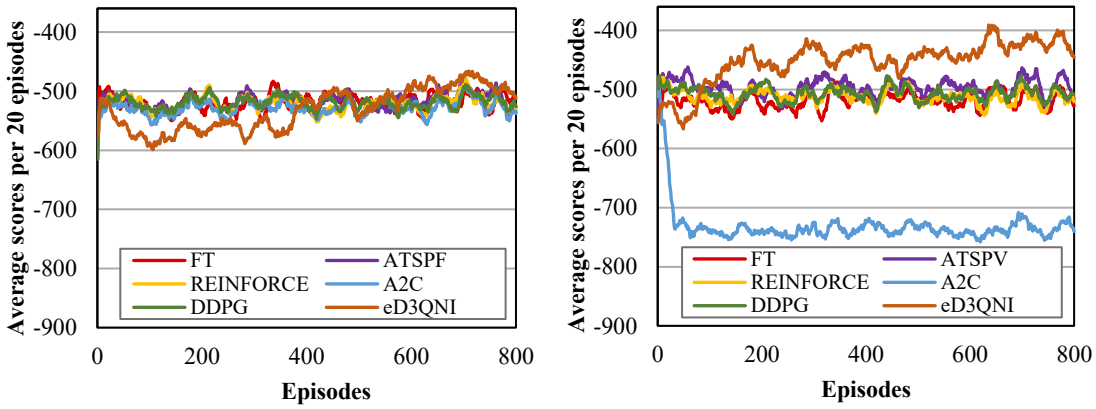
Parameters	eD3QN	REINFORCE/A2C	DDPG
Discount factor γ	0.99	0.99	0.99
Learning rate α	0.0001	0.0001	0.0001
Learning rate β	-	-	0.0001
The number of iterations to replace Parameters N_{rep}	2000	-	-
Network updating parameter τ	-	-	0.001
Replay memory size S_{rep}	20000	-	20000
Batch size S_{bat}	64	-	64
Maximum epsilon ϵ_{max}	0.6	-	-
Minimum epsilon ϵ_{min}	0.1	-	-
Epsilon decrement ϵ_{dec}	5e-5	-	-
Size of action space	2 (for FS) / 4 (for VP)		
SUMO simulation step length	1 s / step		
Simulation time T	1500 s		
Simulation warm-up time T_{warm}	300 s		
Queue length threshold $Q_{threshold}$	30 m		
The number of episodes N	800		

3.3 Results

In this section, the proposed method is evaluated by microscopic traffic simulation. The simulation results reveal the proposed method's efficiency.

3.3.1 The comparison of proposed methods and baselines

The value of cumulative reward of each episode, also called the score, is computed to plot the learning curve, which can represent the efficiency of methods. Since each episode's score constantly oscillates, average scores per 20 episodes are computed to plot the learning curve to make the curve smoother and easier to observe learning trends. The learning curves of all methods are shown in Figure 3.5. It can be seen from the figure that eD3QNI converges to the best results after about 550 and 400 episodes for the FS scheme and VP scheme, respectively. In comparison, other methods do not significantly improve during the training process, and the A2C algorithm even drops down dramatically in the first 50 episodes for the VP scheme. Though eD3QNI does not perform better than other methods before 550 episodes for the FS scheme, the converged score is the highest one, also obviously higher than ATSPF's score. The fixed sequence of phases and satisfaction of minimum and maximum green times limit the flexibility of the proposed algorithm to make decisions, so the differences between scores of all methods are not very obvious. On the contrary, for the variable phase scheme, the benefits of eD3QNI over ATSPV are fully reflected as about 50 more scores are received by eD3QNI than ATSPV on average.



(a) Fixed sequence (FS) scheme (b) Variable phase (VP) scheme

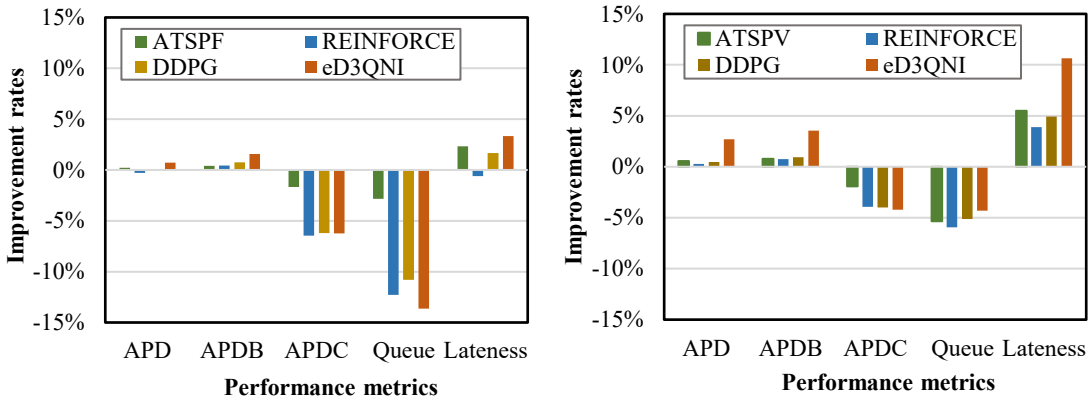
Figure 3.5. The learning curve of different methods.

Such efficiency difference between different algorithms is because REINFORCE method updates every episode without guaranteeing the improvement of each update, and the sample utilization is much lower than eD3QNI with experience replay buffer, resulting in low learning efficiency. Moreover, the A2C method updates every step, which is very likely to cause erroneous parameter updates. The reason is that worse actions may be chosen during exploration, especially at the beginning of the training, as the policy network is just initialized, and incorrect logits value will be outputted from DNN. Though the DDPG method combines the advantages of DQN and PG algorithms, it reaches remarkable benefits on the continuous and large action space. In this study, the actions are discrete with only 2 or 4 terms, so the agent can directly approximate the action value or state value with little computation.

Figure 3.6 depicts the performance results of different methods for the FS scheme and VP scheme, except for poor-performing A2C. Performance indicators are calculated by averaging each model's last

100 episodes data because of high fluctuations within a small number of episodes. They include average person delay (APD), average person delay of buses (APDB), average person delay of cars (APDC), queue per lane, and lateness per person. The improvement rates are based on FT's performance. The positive value means improvement, that is, decrease for the delay, queue, and lateness, and vice versa. It can be seen from Figure 3.6 that eD3QNI outperforms active TSP strategy and other RL methods in terms of the reduced average person delay (0.7% for FS scheme and 2.7% for VP scheme), reduced average person delay of buses (1.6% for FS scheme and 3.5% for VP scheme) and decreased lateness (3.3% for FS scheme and 10.6% for VP scheme). Compared with FT and ATSP, though the queue gets worse in the eD3QNI method, the queue only increases 3.9 and 1.3 m/lane for FS and VP compared to the FT strategy, 3.0 m/lane for FS scheme compared to ATSPF strategy. The simulation results also reveal the trade-off between buses and private vehicles in the TSP strategy; that is, the more the average person delay of buses decreased, the more the average person delay of cars increased.

In overall, we see that the proposed TSP strategy by the eD3QNI algorithm has benefits as it improves the average person delay and lateness with negligible adverse effects on the queue.



(a) Fixed sequence (FS) scheme (b) Variable phase (VP) scheme

Figure 3.6. Performance improvement rates of different methods.

3.3.2 The comparison of VDP and IAM

Both VDP and IAM are used in the eD3QN algorithm to analyze their performance, i.e., eD3QNI and eD3QNV. VDP just can satisfy the green time of basic constraints, while IAM can also consider the phase skipping rule and avoid long waiting situations of drivers, which considers more transportation domain knowledge. The left plot of Figure 3.7 shows the learning curve of the scores. From this, we can know performances of these four algorithms get improved with the episodes. However, they oscillate highly, and it is not easy to observe the convergence efficiency. Hence, the more stable metric, approximated action-value Q by DNN, is also measured to display the learning curve. We run the simulation with the random policy before training and collect 64 states to track their maximum predicted Q value in the training process. The metric average Q value is calculated by averaging the tracked value of those states, of which the result is shown in the right plot of Figure 3.7 The reason why the average Q value does not get increased like theory is as follows. As the DNN is initialized at the beginning of the training, the initial Q value will be estimated to be 0. And the reward is formulated by the delay indicator in this study. Therefore, the Q value will be negative, and the average Q value must gradually decrease with episodes.

As presented in Figure 3.7, eD3QNI converges at about 400 and 600 episodes for FS and VP

schemes, respectively, while eD3QNV converges at almost 750 episodes for two schemes. After converging, the IAM strategy can reach a higher average Q value than the VDP strategy. Hence, the IAM strategy converges faster than the VDP strategy. Table 3.4 lists the performance comparison of the VDP and IAM strategies. IAM strategy has more significant improvements on average person delay, average person delay of bus, and lateness, which is also better than the ATSP approach.

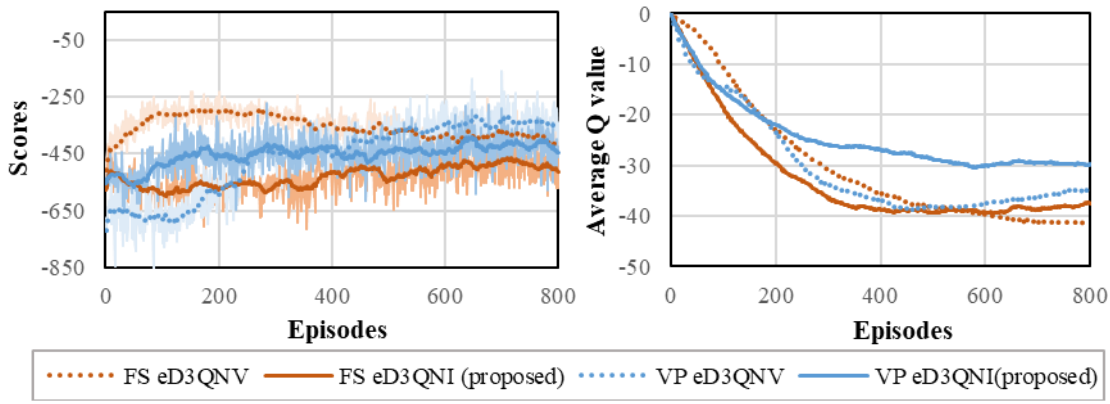


Figure 3.7. The learning curve comparison of VDP and IAM.

Table 3.4. The performance improvement rates' comparison of VDP and IAM.

Method		Delay (s/person)			Queue (m/lane)	Lateness (s/person)
		Total	Bus	Car		
FS	ATSPF	0.2%	0.4%	-1.7%	-2.9%	2.3%
	eD3QNV	-0.4%	0.3%	-6.0%	-13.4%	0.0%
	eD3QNI	0.7%	1.6%	-6.2%	-13.6%	3.3%
VP	ATSPV	0.6%	0.8%	-1.9%	-5.3%	5.5%
	eD3QNV	1.7%	3.2%	-8.3%	-18.5%	-3.7%
	eD3QNI	2.7%	3.5%	-4.2%	-4.3%	10.6%

Hence, the IAM method is superior to VDP in terms of high convergence speed and effective performance improvement. Most importantly, it offers a way to add transportation domain knowledge to the RL algorithm by masking the invalid actions of violating corresponding principles.

3.3.3 The comparison of FS and VP

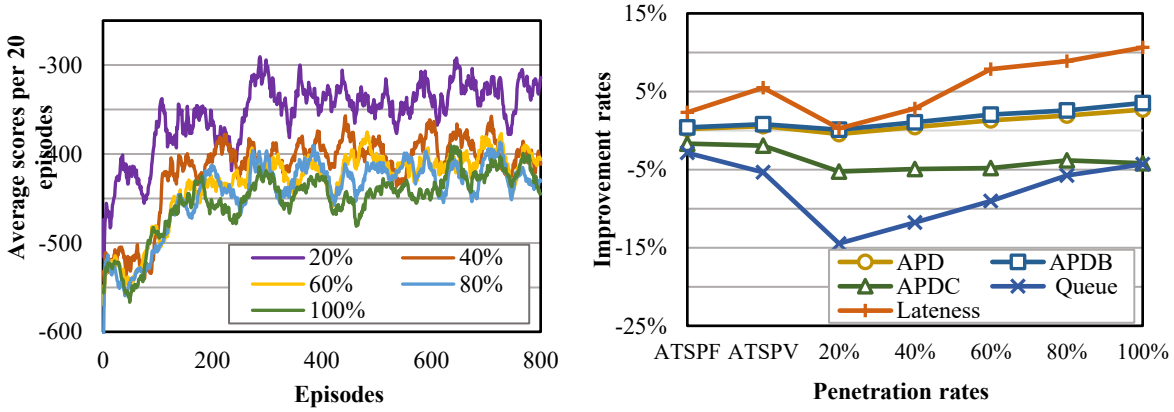
The VP scheme brings more flexibility than the FS scheme for RL to train to choose the best actions. Buses can get more priorities to get the right of way at the intersection so the bus delay can be reduced more. Although the increased delay may be more severe for cars in the meantime, the person delay still gets more significant improvement than the FS scheme.

3.4 Discussions

3.4.1 Impacts of CV penetration rates

As the implementation of the proposed RL algorithm does not need any individual car information, only buses are considered to be equipped with CV technology. However, in reality, not all buses are equipped with V2I OBU to transmit their information to traffic signal control systems. Hence, the agent

can only receive states of partial buses. In this case, it is essential to discuss the impacts of CVs' penetration rates of buses on the performance of the proposed algorithm. Penetration rates mean the proportion of buses with CV technology. If the penetration rate equals 0, no bus information is sent to the agent, and the agent will fail as no reward can be returned. Thus, we just evaluate the impacts of penetration rates from 20% to 100% with the increment of 20%.Figure 3.8 presents the results of the proposed algorithm for the VP scheme with different penetration rates. To better compare the effect of penetration rates on performance, results of two ATSP strategies are added as the benchmark at the leftmost side in Figure 3.8(b).



(a) Learning curve (b) Performance improvement rates

Figure 3.8. Comparing results of different penetration rates.

From Figure 3.8(a), it can be seen that the penetration rate does not affect the convergence efficiency of the eD3QNI algorithm, as they all get converged at around 400 episodes. However, in Figure 3.8(b), the scenario with a higher penetration rate shows better performance, including the lower average person delay and lateness, and the smaller queue. Moreover, the average delay reduces steadily with the increase of penetration rates, while the lateness decreases vastly before 60% of connected buses' penetration and decreases less after that. When not more than 40% and 60% of buses are equipped with CV technology, the performance of the eD3QNI algorithm will be worse than the performance of ATSPF and ATSPV respectively in terms of average person delay and lateness.

3.4.2 Performance of different reward functions

As the RL method is to maximize the expected cumulative reward, the effect of the proposed eD3QNI highly depends on the reward formulation. The aforementioned results are all based on the reward function given by the average person delay of buses as Eq (3.5), called Reward 1. In order to analyze the impacts of reward functions on the efficiency of the proposed algorithm, two more reward functions are formulated as:

Reward 2:

$$r_t = -\sum_k(\hat{d}_{kt} * SD_{kt} * O_{kt})/\sum_k(SD_{kt} * O_{kt}) \tag{3.11}$$

Reward 3:

$$r_t = R_{Bus} + wR_{Car} = -\sum_k(\hat{d}_{kt} * O_{kt})/\sum_k O_{kt} - w * \sum_c(d_{ct} * NP_{ct})/\sum_c NP_{ct} \tag{3.12}$$

where w is a weight.

As shown in Eq (3.11)-(3.12), Reward 2 is formulated as the average delay of buses weighted by

total lateness, and Reward 3 is the weighted sum of average person delay of buses and cars. On-time performance is an indicator of the ability to be on time in transportation, and it becomes the critical target of public transportation service. The function of Reward 2 is one way to improve the on-time performance. In addition, many current TSP strategies are conditional as the performance of private vehicles will be considered to determine whether give bus priority. The formulation of reward 3 is the method to implement conditional TSP strategy in the RL algorithm. Therefore, Reward 3 can consider the operational efficiency of both cars and buses when conducting TSP, which is different from Reward 1 for the unconditional TSP strategy aiming to prioritize late buses.

Using Reward 3 requires knowledge of both bus occupancy and car occupancy. In the simulation, we have collected the actual passenger occupancy of each bus every time step, but assumed all cars have the same average passenger occupancy 1.2 person/veh to simply evaluate the performance of Reward 3, because APC systems are installed on transit vehicles but not yet for cars. The weight of Reward 3 will influence the proposed TSP strategy results. For example, a larger weight for cars will lead the agent to focus more on cars than buses, and then the benefits of buses will be reduced. Note that buses are set to carry about 35 person/veh on average, while cars are set to carry only 1.2 person/veh, so the number of passengers carried by a bus dominates that carried by a car. If the weight for cars is not properly assigned, the reward that considers the benefits of both bus and cars (i.e., total person delay) will be outperformed by the reward that only considers the benefit of buses (i.e., person delay of buses).

Thus, sensitivity analysis of weights has been conducted by eD3QNI under the VP scheme, considering weights of 0.01, 0.03, 0.15, and 1.5. Additionally, Reward 1 can be regarded as Reward 3 with a weight of 0. Note that the ratio of passenger occupancies of a bus and a car is $35/1.2 \approx 30$. According to the number of passengers carried, one bus is equivalent to 30 cars. Therefore, the weight being $1/30 = 0.03$ implies that the reward function equally considers the benefits of one bus and one car. Figure 3.9 shows performance improvement rates of Reward 1, Reward 2 and Reward 3 with different weights based on FT. From the figure, we can see that when Reward 3's weight is less than 0.03, APDC is nearly unchanged, but the improvement of APDB is reduced with the increase of weight so that the improvement of APD is also reduced. Moreover, as the weight increases greater than 0.03, APDC and APDB increase, and the improvement of APD is reduced. Therefore, when Reward 3's weight is 0.03, which means that one bus and one car are given nearly equal preference, it achieves the maximal improvement on APDB and minimal adverse effect on APDC, resulting in the highest improvement of APD.

Choosing 0.03 as the weight of Reward 3, we evaluate the performance of three reward functions by eD3QNI under the VP scheme. From Figure 3.9, when the delay weighted by lateness (Reward 2) is the reward function, the lateness will decrease more than Reward 1; when the reward function considers the delay of cars (Reward 3), the TSP will have a less negative effect on cars than Reward 1.

The results show that specific reward functions can be designed based on the proposed RL framework to develop TSP strategies with users' different preferences, like minimizing bus passenger delay, improving on-time performance, or decreasing car delay. However, when adopting the reward function, which improves the operational efficiency of both cars and buses, all cars are required to install occupant classification systems and connected technology to collect and transmit car passenger occupancy, and the weight of the reward function should be appropriately adopted. The weight is recommended to be the ratio of passenger occupancies of cars and buses to consider the benefits of one bus and one car equally. To clarify different use cases of those reward functions, Table 3.5 lists their comparisons. It is clearer to identify the applicable case for each reward function, considering the goal

and the technology required to obtain needed information.

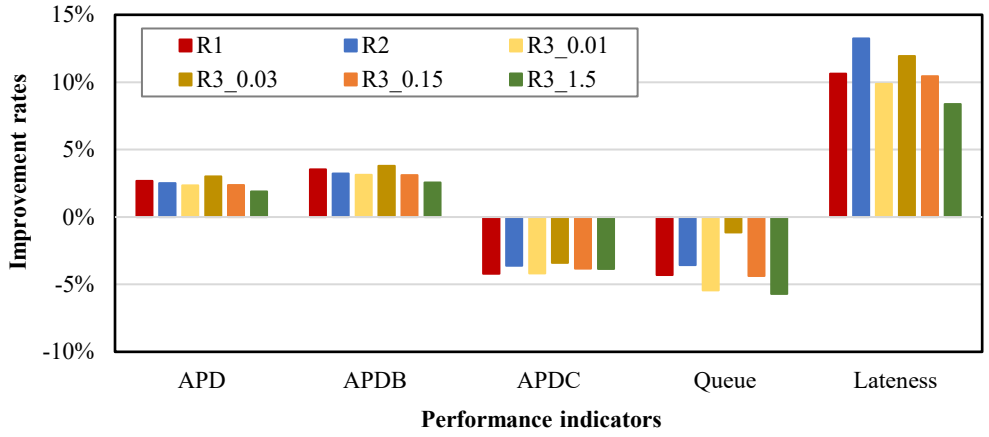


Figure 3.9. Performance comparison between different reward functions. (Note: R1 and R2 are Reward 1 and Reward 2, respectively; R3_0.01, R3_0.03, R3_0.15, R3_1.5 are Reward 3 with weight of 0.01, 0.03, 0.15, 1.5, respectively.)

Table 3.5. Comparisons between different reward functions.

Details		Reward functions		
		Reward 1 (Eq (3.5))	Reward 2 (Eq (3.11))	Reward 3 (Eq (3.12))
Goals		To reduce the person delay of buses	To improve the on-time performance of buses	To reduce the person delay of both cars and buses
Information required to transmit to traffic signal systems by	Conventional detection technologies	Average speed and queue length measured by cameras at intersections		
	V2I technology	▪ Bus location captured by GPS ▪ Bus schedule deviation measured when arriving at bus stops ▪ Bus passenger occupancy by electronic payment systems or automatic passenger counters installed in buses		
				Car passenger occupancy by occupant classification systems installed in connected cars
Advantages		It is practically feasible and can significantly improve transit efficiency.	It will decrease lateness more than the TSP strategy with other reward functions.	It has a less negative effect on cars and even achieves the best performance in total person delay when a proper weight of the function is adopted.
Disadvantages and challenges		It will have a relatively more negative effect on the car's efficiency.	The benefits on average person delay are cut compared to other reward functions.	▪ Every car needs to be equipped with occupant classification systems and connected technology to collect and transmit car passenger occupancy, which is not feasible yet. ▪ The performance is very susceptible to the weight value of the function, which is not easy to find a proper one.

3.5 Summary

This chapter proposes the eD3QNI algorithm for the TSP strategy at isolated intersections to improve bus operational efficiency. Compared to existing studies, the proposed method utilizes invalid action masking for traffic signal constraints and phase skipping rule, adopts a person-based reward function, and considers multiple conflicting bus priority requests. Hence, it is a driver-friendly and person-based strategy and is more robust to dynamic traffic with complex bus routes than existing research with model assumptions, fixed cycle length, or limited bus lines.

The proposed algorithm is evaluated by embedding it into a microscopic traffic simulation platform. The simulation results show that the eD3QNI method can effectively exploit the unknown deep-buried relationship between the environment and the agent and achieve better results of bus operational efficiency than fixed-time signal, active TSP strategy, and other common RL methods. The eD3QNI could become an effective TSP strategy when the method's parameters are trained well. Note that, in this algorithm, it is the employment of the IAM strategy that provides a way to incorporate traffic engineers' knowledge into the development of an RL scheme for rational TSP strategy. Furthermore, the evaluation results prove the improved performance of the proposed method over the traditional VDP method in terms of the training process, convergence speed, and improvement in traffic-related metrics. The penetration rates of connected buses do not affect the convergence efficiency of the proposed algorithm, although the larger it is, the better performance this algorithm achieves. Finally, the proposed framework can accommodate different specific reward functions to develop TSP strategies to realize different operation goals.

Chapter 4

TSP at multiple intersections to improve schedule adherence

This chapter extends the work from an isolated intersection to an arterial road with multiple intersections. With the increase in the number of intersections, more difficulties arise, including 1) how to handle cooperation among intersections when they may make decisions asynchronously, 2) how to ensure the experience quality and training efficiency when the environment becomes much more complicated, and 3) how to formulate proper reward functions by feasible obtained data to promote schedule adherence while considering the trade-off between transit and non-transit vehicles. This chapter will address those issues and propose a cooperative TSP strategy at multiple intersections based on the MARL framework to improve bus schedule adherence.

This chapter is organized as follows. Section 4.1 introduces the methodology of this chapter and describes the proposed CTSPV algorithm. The experiment framework is presented in Section 4.2. Experiment results are shown in Section 4.3. Section 4.4 discusses the signal results of the proposed approach. Section 4.5 summarizes this chapter.

4.1 Methodology

4.1.1 Problem statement

This work considers an arterial road with N signalized intersections, and several conflicting bus routes pass through those intersections. The traffic signal controller needs to determine the phase at the next step for each intersection according to the observed real-time traffic, e.g., detected vehicle speed, queue length, and information on approaching buses. Once the signal timings are executed, the detection system will observe new traffic conditions, and we can calculate some performance metrics to obtain immediate feedback for those signal timings. Hence, it can be described as a MDP and solved by MARL. Below are detailed descriptions of this problem.

(1) Agents

Each intersection along the arterial is controlled by an agent, which takes traffic conditions as the state s_t, traffic signal as an action a_t, and performance feedback as a joint reward r_t for each step t. It is constructed by the RL algorithm and can learn from transition experiences to choose actions maximizing discounted cumulative reward $G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$ where γ is a discounting factor, $0 < \gamma < 1$ (Sutton and Barto, 2018).

(2) State

At each step t, the state s_t composes four parts of the local intersection, including average speed S_A, queue length L_Q, current phase P_C, and bus state S_{Bus}.

a. Average speed S_A

$$S_A = \{S_{Alm}\}, l \in \Psi, m \in M_l \tag{4.1}$$

where S_{Alm} represents the average speed of movement m at the approach link l; Ψ is the approach set, usually containing southbound, northbound, eastbound, and westbound; M_l is the movement set for approach link l, including through and left-turn movements for right-hand traffic, in which right-turns are permitted to pass on red.

b. Queue length L_Q

$$L_Q = \{L_{Qlm}\}, l \in \Psi, m \in M_l \tag{4.2}$$

where L_{Qlm} represents the queue length of movement m at the approach link l. Here we take the maximum queue length for the movement with more than one lane.

c. Current phase P_C

It is a one-hot encoding vector where the value of one refers to the current phase. If we consider a signal scheme with four phases, the vector would have eight values, successively meaning the green and the yellow of those four phases.

d. Bus state S_{Bus}

A proper representation of the bus state will allow agents to better serve bus priority requests, even in cases with multiple conflicting priority requests. In this work, we also record the distance to the stop line D, schedule delay SD, and passenger occupancy O of buses approaching the intersection as in Chapter 3. Schedule delay is the delay of one bus compared to its schedule, and we set it to be 0 if the bus is not late. Passenger occupancy is the number of bus passengers that the onboard APCs can obtain. Hence, the bus state is denoted by

$$S_{Bus} = \{D_{pb}, SD_{pb}, O_{pb}\}, p \in \mathrm{P}, b \in \Gamma \tag{4.3}$$

where D_{pb}, SD_{pb}, and O_{pb} represent the distance to the stop line, schedule delay, and passenger occupancy of bus b, which requests the green time of phase p. P is the phase set, e.g., $P = \{0,1,2,3\}$ if four phases are considered in the signal scheme. Γ is the bus set with a fixed total number, and here we set $|\Gamma| = 3$ so we only put the information of the three most urgent buses in S_{Bus} for each phase p. The urgency is measured by $\frac{SD_{pb}O_{pb}}{D_{pb}+\delta}$ where $\delta = 10^{-5}$ to avoid invalid calculations. Therefore, we can simultaneously consider 12 bus requests from different directions at each intersection.

(3) Action

In this study, the action a_t of agents is to choose one phase for the next step $t+1$ from a set of valid signal phases A_t at each step t, and thus agents can decide the phase duration and sequence. To tackle the asynchronous decision-making between different agents, we also treat the yellow time as an action so that each agent can choose one action at every step. Then the universal action space is a set of the green and the yellow of all considered phases. In contrast, the valid action space is a subset of them.

We deploy Invalid Action Masking (Huang and Ontañón, 2020) to mask the actions that do not satisfy basic constraints based on traffic engineering knowledge. The masking method is to add a large negative value on the original action-value Q and then choose the action maximizing the masked action-value Q^{masked}. Phases can only be valid when they satisfy the basic constraints.

Basic constraints consist of two aspects: signal constraints and skipping rules.

a. Signal constraints

Signal constraints contain two parts, i.e., the minimum green time G_{min} and maximum green time G_{max}, which are essential for traffic signal control. This study considers three constraint settings, namely Cons_1, Cons_2, and Cons_3. We use the Cons_3 to evaluate the performance of the proposed method compared to other approaches. Still, we also analyze the results of these three constraints in *Section 4.3.3 Effects of Gmin and Gmax constraints* and explain why Cons_3 is chosen.

- Cons_1:

$$G_{min} = \frac{D_{Cross}}{2V_{Ped}},\ G_{max} = G_{min} + 100 \tag{4.4}$$

where D_{Cross} is the length of the pedestrian crossing (unit: m); V_{Ped} is the average velocity of pedestrians, set as 1.2 m/s.

- Cons_2:

$$G_{min} = \max\left(\frac{D_{Cross}}{2V_{Ped}}, \frac{C*q}{f}\right),\ G_{max} = G_{min} + 30 \tag{4.5}$$

where C is the cycle length of baseline fixed-time signal (unit: s); q is the traffic volume (veh/h); f is the saturation flow (veh/h).

- Cons_3:

$$G_{min} = \max\left(\frac{D_{Cross}}{2V_{Ped}}, 3 + 2\frac{L_{Qp} - \frac{\sum_{\dot{p}\in P, \dot{p}\neq p} L_{Q\dot{p}}}{|P|-1}}{7.62}\right),\ G_{max} = \begin{cases} 120, & \text{for the major through phase} \\ 90, & \text{for the minor through phase} \\ 60, & \text{for the left-turn phase} \end{cases} \tag{4.6}$$

where L_{Qp} is the maximum queue length of phase p's movements (unit: m), e.g., if phase p is minor through phase, then we take the maximum of two-way minor through movements' queue. $|P|$ is the size of phase set P.

It should be noted that we set 3 seconds for yellow time. Finally, the valid action space A_t will have the following situations: 1) When G_{min} is not satisfied, the agent can only keep the current phase, i.e., A_t = *current green*; 2) When both G_{min} and G_{max} are satisfied, the agent can choose to keep the current phase or switch to yellow, i.e., A_t = *current green and yellow*; 3) When it is during a yellow time, or the phase duration reaches G_{max}, the agent only can choose yellow, i.e., A_t = *yellow*; 4) When it is the end of yellow time, the agent can choose any green phase, i.e., A_t = *any green*.

b. Skipping rules

We use the same phase skipping rules in isolated work: 1) phases cannot be skipped twice in a row when a queue is detected for the phase; or 2) phases cannot be skipped when the phase has a queue over certain threshold. Therefore, once one phase cannot be skipped, phases after that phase in the defined sequence will be invalid actions.

(4) Reward function

After executing action a_t, agents receive a joint reward r_t from the environment. We expect the proposed strategy can 1) prioritize buses to improve bus schedule adherence and bring less detrimental effects on non-prioritized traffic, and 2) improve the general traffic efficiency when there are few buses. Hence, the reward is defined as the weighted sum of queue density and bus lateness, given by Eq (4.7). In practice, these two terms also are more feasible to obtain by the detection system than the total person delay, which is usually used.

$$r_t = w_1 \frac{\sum_l \sum_\varphi L_{Ql\varphi t}}{\sum_l \sum_\varphi Det_{l\varphi}} + w_2 \frac{\sum_k (SD_{kt} * O_{kt})}{\sum_k O_{kt}} \tag{4.7}$$

where $L_{Ql\varphi t}$ and $Det_{l\varphi}$ represent the queue length at time t and detector length on the lane φ of the approach link l in the whole network, respectively; SD_{kt} and O_{kt} represent the schedule delay and passenger occupancy of bus k in the road network at time t; w_1 and w_2 are two negative weights for queues of general traffic and lateness of buses, respectively.

(5) Evaluation metrics

We introduce various metrics to assess the performance of MARL-based TSP strategies on learning efficiency and traffic conditions.

(1) Algorithm learning metrics:

- *Scores*: It is the sum of immediate rewards in one episode. After observing the scores with the training episodes, we can plot one learning curve to show the learning efficiency and convergence point.
- *Qtot*: This value reflects how well agents perform joint actions under a specific state and policy. We expect it to increase with the training steps. Like scores, their evolution during training can be used to represent the learning curve, and we can observe the learning performance. MARL networks need many steps to train; therefore, we extract this value per 1000 training steps.

(2) Traffic performance metrics:

- *Average person delay*: This belongs to person-based metrics. In this study, we compute the Average Person Delay for general traffic, Cars, and Buses, denoted as APD, APDC, and APDB by Eq (3.6) -(3.8).
- *Queue*: This is a road-based metric that can be calculated by detected queue length on the road. It is the queue length averaged by lanes and time, given by Eq (3.9).
- *Lateness*: It is the average lateness of each bus passenger per time step, computed by Eq (3.10).

4.1.2 CTSPV algorithm

This study proposes the *Cooperative TSP strategy of Variable phase* (CTSPV) based on QMIX (Rashid et al., 2018) and Dueling Double Deep Q Network (D3QN) (Wang et al., 2016). It masks invalid actions by the IAM algorithm (Huang and Ontañón, 2020) to satisfy basic constraints, and samples minibatch by PER (Schaul et al., 2016) to improve sample efficiency and speed learning. It also explores experiences by the ϵ-greedy approach, which randomly chooses actions with a probability of ϵ. Algorithm 1 provides the pesudocode to implement the proposed CTSPV approach, and Figure 4.1 illustrates its framework to approximate individual and total action-value given the state, action and reward.

Algorithm 1: CTSPV algorithm

Input: Discount factor γ, learning rate α, epsilon of ϵ-greedy ϵ, target networks update iterations N_{rep}, replay buffer B and its size S_{rep}, minibatch size S_{bat}, number of episodes N_e, traffic simulation time T, simulation warm-up time T_w, number of agents N

Output: trained parameters of the CTSPV algorithm

Initialize replay buffer B, minibatch MB, main network parameters Θ, and target network parameters $\Theta^- \leftarrow \Theta$, training step $\tau \leftarrow 0$
for $episode = 1$ **to** N_e **do**
  for $t = 1$ **to** T **do**
    Run one simulation step
    if $t \geq T_w$ **then**
      // Execute and store experience
      Obtain state information $S_t = \{s_t^i\}$ for all agents $i \in \{1, \dots, N\}$
      for agent $i = 1$ **to** N **do**
        Obtain valid action space A_t^i, generate $rand \leftarrow \text{Uniform}(0,1)$
        Choose action $a_t^i = \begin{cases} \text{argmax}_a Q^{masked}(s_t^i, a_t^i; \theta), & rand > \epsilon \\ \text{random } a \in A_t^i, & otherwise \end{cases}$
      end for
      Execute action $\Lambda_t = \{a_t^i\}$ for all $i \in \{1, \dots, N\}$ and receive the latest state $S'_t = \{s'^i_t\}$ for all $i \in \{1, \dots, N\}$ and reward r_t (Eq (4.7))
      if replay buffer reaches the size S_{rep} **then**
        Delete the oldest experience tuples in B
      end if
      $B \leftarrow B \cup \{S_t, \Lambda_t, S'_t, r_t\}$
      Calculate and store the importance weight of each experience for priority experience replay
      // Train networks
      if the number of stored experiences > S_{bat} **then**
        Sample a minibatch MB from B by PER
        for j in MB **do**
          Calculate TD-error $\sigma_j = y_{tot_j} - Q_{tot_j}$ and update the importance weight of sample j
        end for
        Update main network parameters $\Theta, \tau \leftarrow \tau + 1$
        if τ % $N_{rep} = 0$ **then**
          Update target network parameters $\Theta^- \leftarrow \Theta$
        end if
      end if
    end if
  end for
end for

The proposed method conforms to CTDE. Agents are trained by the loss functions formulated by the total Q-value, Q_{tot} (also called centralized Q-value) but execute solely by choosing actions after observing a local state in accordance with their own Q. We define $Q_{tot}(S_t, \Lambda_t; \Theta)$ as the total Q-value of joint action Λ_t by main networks with parameters Θ given the global state S_t at time t, and $Q^i(s_t^i, a_t^i; \theta^i)$ as the Q-value of action a_t^i for agent i ($i \in N$) by the network with parameters θ^i given the state s_t^i at time t. N is the number of intersections.

The loss function for training is computed by

$$\mathcal{L}(\Theta) = \sum_{i=1}^{S_{bat}} (y_{tot} - Q_{tot}(S_t, \Lambda_t; \Theta))^2 \tag{4.8}$$

where $y_{tot} = r_t + \gamma \hat{Q}_{tot}(S'_t, \Lambda'_t; \Theta^-)$, $\hat{Q}_{tot}(S'_t, \Lambda'_t; \Theta^-)$ is the total Q-value of the target network, given by $\hat{Q}_{tot} = Q_{tot}\left(S_t', arg\text{max}_{\Lambda_t'} Q_{tot}(S_t', \Lambda_t'; \Theta); \Theta^-\right)$; S_t' is the next global state at time t; Θ^- is the

parameters of target networks for all agents.

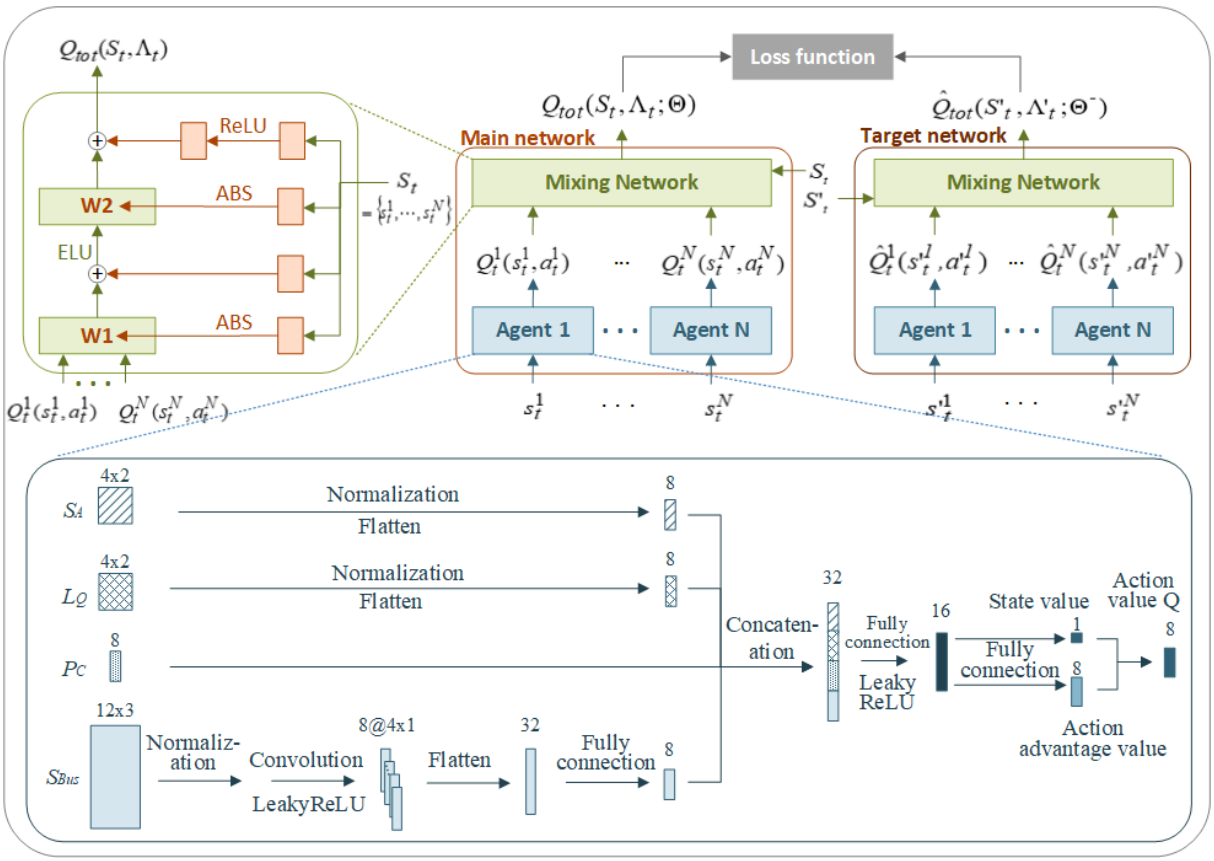


Figure 4.1. The QMIX framework of the proposed method.

In the proposed method, each agent selects action a_t^i given its policy and s_t^i, and then calculate their own $Q^i(s_t^i, a_t^i; \theta^i)$, which is then fed into the mixing network along with the global state S_t to approximate $Q_{tot}(S_t, \Lambda_t; \Theta)$ for centralized training. Hence, the centralized value function relies on the cooperation of all agents to maximize its outputs. The mixing network ensures that the joint action from performing *argmax* Q_{tot} is equal to the set of individual actions from performing *argmax* Q of each agent, which is realized by constraining the non-negative monotonicity between Q_{tot} and each Q, i.e., Eq (4.9). The monotonicity is guaranteed by restricting parameters to be non-negative, as the ReLU function of the mixing network illustrated in Figure 4.1.

$$\frac{\partial Q_{tot}(S_t, \Lambda_t; \Theta)}{\partial Q^i(s_t^i, a_t^i; \theta^i)} \geq 0, i \in N \tag{4.9}$$

Additionally, the credit assignment among agents can be learned in the mixing network during the training procedure by having Q_{tot} conditioned on the extra information, like the global state. Overall, the proposed method is trained with a centralized value function, which needs agents' cooperation to maximize their shared reward and learn a proper credit assignment; accordingly, the well-trained agent of each intersection can yield cooperative actions on traffic signals when performing independently, which can achieve the minimum queue length and bus schedule delay.

4.2 Experiment

4.2.1 Simulated environment

To evaluate the performance of the proposed CTSPV strategy, we simulated an arterial in the SUMO platform as the environment to interact with agents. As shown in Figure 4.2, three intersections are considered in the simulated environment, of which the east-west direction is the major road, and the three north-south roads are minor. The spacing between two adjacent intersections is 450 m. There are a total of eight entrances and exits, composing 56 Origin-Destination (OD) pairs.

Traffic demands: We set five periods for such OD-pairs traffic demands, in which the flow ratios of arterial through movements are about 0.35, 0.5, 0.55, 0.45, and 0.3, respectively. Those five periods (S1~S5) are set for 15, 15, 30, 15, and 15 mins to simulate the traffic volume increases from off-peak to peak and then decreases to off-peak. As for transit, we consider six two-way bus lines, as illustrated in Figure 4.2. The headway of Line E0/W0 is scheduled to 10 mins for S1 and S5, and 5 mins for the other three periods, while the headway of rest lines is scheduled to 20 mins for S1 and S5, and 10 mins for the other three periods. We also set random seeds for SUMO to generate vehicles differently with episodes.

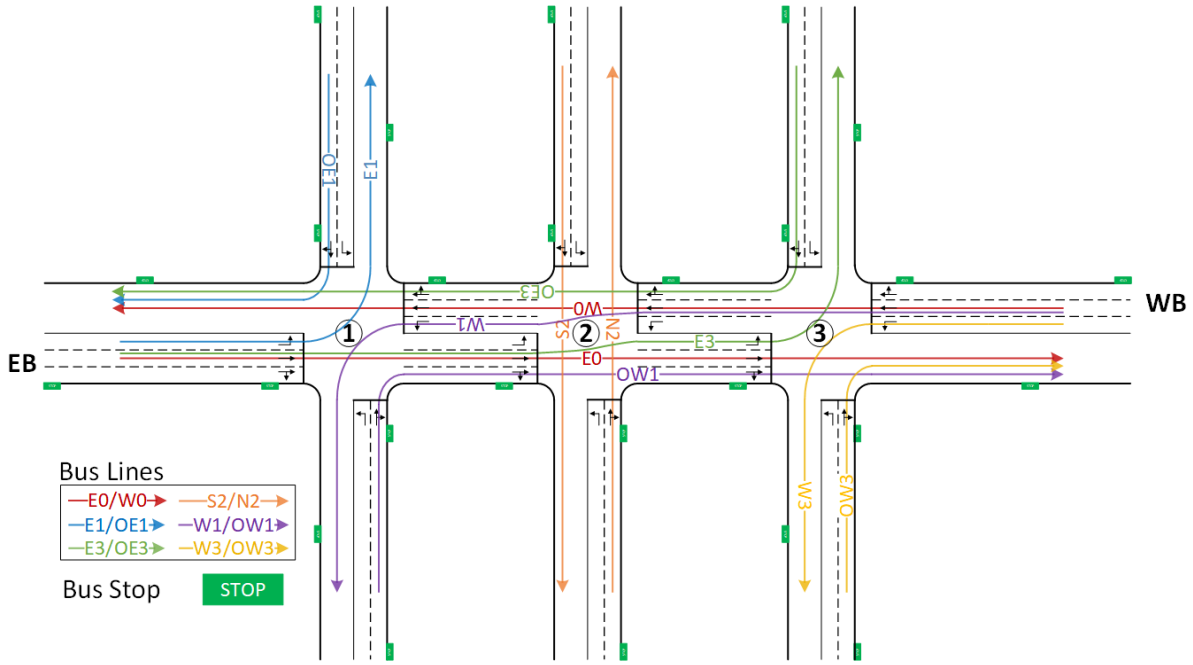


Figure 4.2. The layout of the simulated environment.

Traffic signal phase: Each intersection is controlled by the traffic signal with a typical four-phase scheme, i.e., major through phase P_0, major left-turn phase P_1, minor through phase P_2, and minor left-turn phase P_3. Agents can decide the phase duration and skip some phases in each step.

Bus settings: Bus will be inserted into this network according to its schedule headway and a random delay, which is produced by a normal distribution $N(0,120^2)$. We also simulate passengers of different bus routes getting on at each bus stop and getting off three stops later. At each stop, the passenger arrival rate of Line E0/W0 is 0.0167 person/s for S1 and S5, and 0.033 person/s for other periods, while the passenger arrival rate of rest lines is 0.0083 person/s for S1 and S5, and 0.0167 person/s for other periods. Therefore, the bus occupancy changes with the passenger, and we also set

an occupancy capacity of 70 pax for each bus.

4.2.2 Compared methods

We compare our proposed CTSPV method with the following baselines.

(1) *Coordinated Fixed-Time signal* (CFT): We obtain the optimal fixed signal timing considering arterial coordination by SIDRA INTERSECTION, a microanalysis software to aid in designing and evaluating isolated or network of intersections. To show the coordination effect of CFT, we plot the time-space diagram of major through movements taking signal timings for demand of period 1 as an example, as shown in Figure 4.3. CFT can provide sufficient green bandwidth for through movements to pass three intersections efficiently.

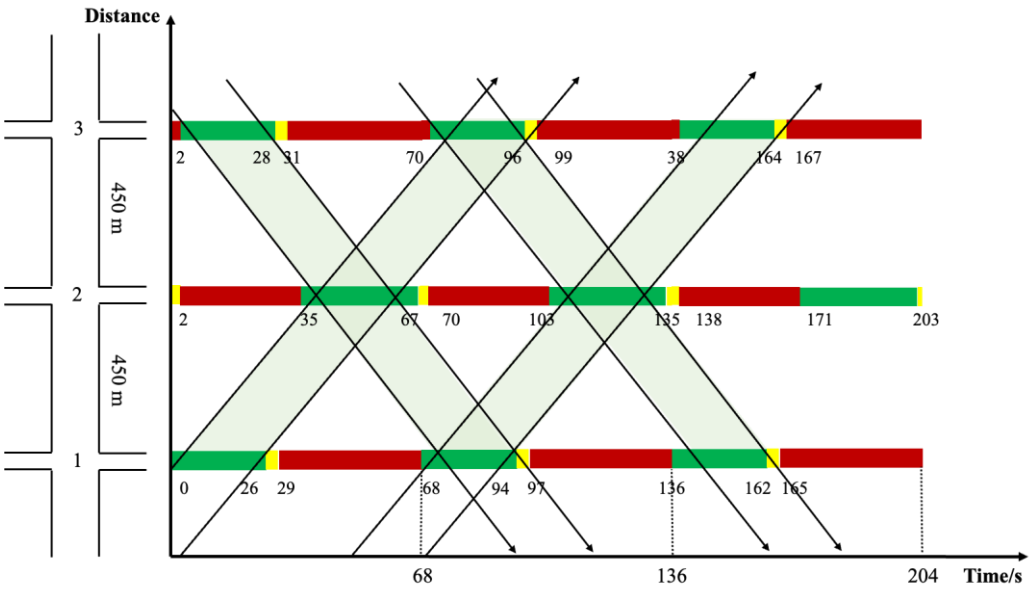


Figure 4.3. The time-space diagram of through movement at major road under S1 demand.

(2) *Independent Fixed Time with Phase Skipping* (IFTPS): We use SIDRA software to obtain the base signal timing for each intersection independently. Then, at the end of yellow, we allow skipping to the most urgent valid phase following the skipping rules mentioned in *Section 4.1.1(3) Action*. The urgency of phase p is determined by F value

$$F_p = \xi Speed_p + \varrho Queue_p + \eta Prio_p \tag{4.10}$$

where $Speed_p$, $Queue_p$ and $Prio_p$ are the average speed, average queue length, and bus urgency of phase p movements, $Prio_p = \sum_b \frac{SD_{pb} O_{pb}}{D_{pb}+\delta}$, $\xi = -0.1$, $\varrho = 0.01$, $\eta = 0.001$. The variables of this F value are calculated by RL methods' input information (states).

(2) *Long Queue First Algorithm* (LQFA): It is an online adaptive signal control strategy (Ahmed et al., 2023), which prioritizes lanes with longer queue lengths.

(3) MARL methods:

- *Independent Q Learning* (IQL): Each intersection is controlled by an independent eD3QNI agent, as proposed in *Section 3.1.2 eD3QNI algorithm*. Hence it is a kind of DTDE method.
- *Value Decomposition Network* (VDN): It is also a CTDE method, introducing a total Q to realize the cooperation between different agents. Its only difference from the proposed method is that the

total Q-value is calculated by $Q_{tot}(S_t, \Lambda_t; \Theta) = \sum_{i=0}^{N} Q_i(s_t^i, a_t^i; \theta^i)$, instead of by the mixing network.

The parameters of the proposed method and other MARL methods are described in Table 4.1.

Table 4.1. The parameters settings.

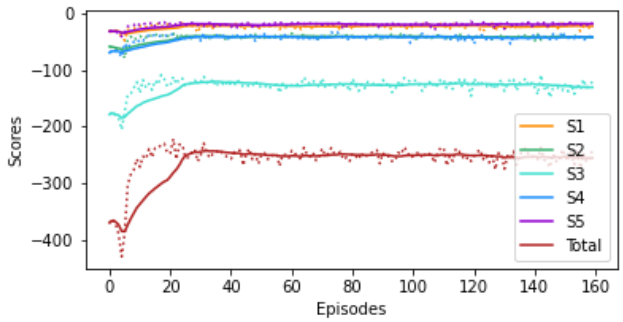
Parameters	Value
Discount factor	0.99
Learning rate	0.0001
Epsilon of ϵ-greedy	Exponentially decade from 0.6 to 0.1 with a rate 5e-5
Target networks replace iterations	2000
Size of the replay buffer	20000
Size of minibatch	64
Number of episodes	160
Traffic simulation time	5700 s (the first 300 s for warm-up)
w_1 and w_2 for the reward	-1 and -1/6000

4.3 Results

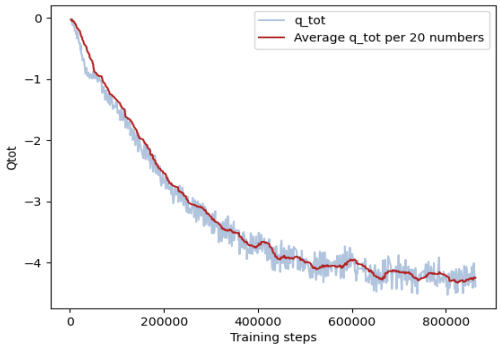
In this section, the proposed CTSPV method and other baselines are evaluated with Cons_3 by microscopic traffic simulation. For each episode, we take different random seeds to start the simulation. The simulation results reveal the proposed method's efficiency.

4.3.1 Algorithm learning

The learning curve can clearly illustrate the learning efficiency and convergence performance. In this study, both scores and Qtot are used to plot the learning curve, as shown in Figure 4.4. To observe the learning trends easily, we also compute the average score and Qtot per 20 values. The left figure (a) is the curve of scores with episodes for a total and five individual periods, in which solid lines are the averaged scores. It shows all six learning curves increase and basically flatten after 30 episodes. And the most significant improvement is contributed to the learned knowledge for periods with heavy traffic demand, i.e., S3. The periods with similar traffic demand shows similar learning performance, i.e., S1 & S5, and S2 & S4. In addition, the right figure (b) plots the curve of Qtot with episodes, displaying that the proposed CTSPV strategy finally converges after 0.7 million training steps. Therefore, the proposed algorithm can learn well from interactions with the environment.



(a) Scores. (S1~S5 refers to the five periods of traffic demands, and the Total is the sum of all five periods)



(b) Qtot.

Figure 4.4. The learning curve of the CTSPV strategy.

4.3.2 Performance comparison

We average the last converged 60 episodes to calculate evaluation metrics. Based on the metrics value of the CFT method, we compute the improvement rate of other approaches, as illustrated in Figure 4.5. The improvements of different periods are also given. Looking at the total value for each metric, we can find the CTSPV strategy improves most for all performance indices, in which queue, lateness, and average person delay (APD) decrease by 31.6%, 17.0%, and 8.7%, respectively. Moreover, focusing on car-dominated indicators (such as APDC and queue), the worst to best methods are IFTPS, IQL, VDN, CTSPV, and LQFA; for example, their queue lengths compared to CFT significantly decrease by 2.0%, 27.2%, 30.0%, and 32.2%, respectively. As LQFA is specifically designed to reduce queue length, it improves car-dominated metrics the most. However, focusing on bus-related metrics (such as APDB and lateness), the worst to best methods are IFTPS, IQL, LQFA, VDN, and CTSPV. In detail, their corresponding lateness compared to CFT significantly reduces by 4.9%, 12.2%, 13.7%, 16.0%, and 17.0%, respectively. It proves that the proposed CTSPV can reduce schedule delay and improve schedule adherence the most. Comparing them in pairs, the slightly better performance of IFTPS than CFT shows the benefits of skipping unurgent phases; VDN yielding better performance than IQL shows the advantages of cooperation among agents; the improvement of CTSPV over the VDN proves the significance of the proper credit assignment among agents, i.e., specifying the contributions of each agent's action on outcomes. Consequently, the proposed strategy with the above three superiorities achieves the best results.

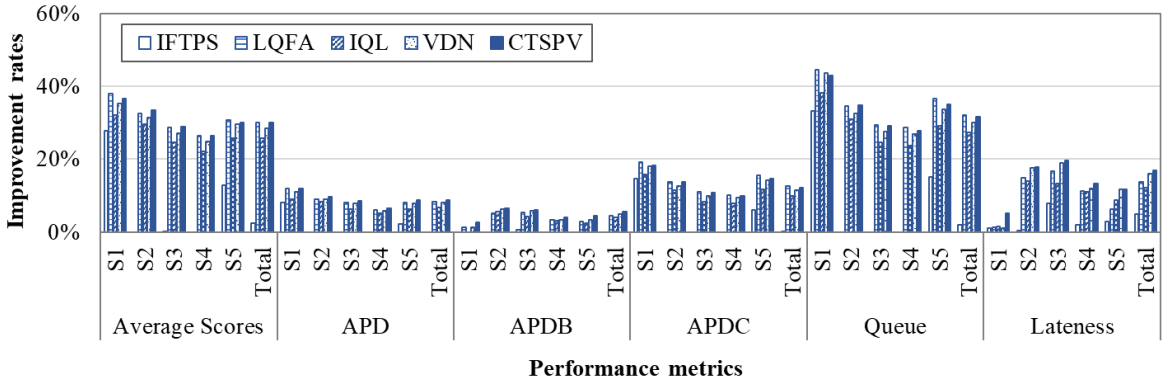


Figure 4.5. The performance comparison of the CTSPV strategy with baselines.

From the detailed five-period view, CTSPV has the largest improvement rates on all performance metrics for all traffic demands. Additionally, the CTSPV improves more on scores, APD, APDC, and queue during periods with less traffic demand, while the metrics related to buses, i.e., APDB and Lateness, improve most during S3 with heavy demand. Therefore, under higher demand, the CTSPV strategy can improve bus on-time performance more significantly.

4.3.3 Effects of Gmin and Gmax constraints

In this study, we integrate signal constraints into the RL algorithm to constrain the action-choosing behavior of agents. Hence, such restrictions of G_{min} and G_{max} directly influence the proposed algorithm's effect. We calculate the performance metrics under three signal constraints mentioned in *Section 4.1.1(3) Signal constraints*. As given by Eq (4.4)-(4.6), G_{min} of Cons_1, roughly referring to pedestrians' crossing time, is very small so that agents have fewer limitations on choosing actions; G_{min}

of Cons_2 aims to clear all queues for each phase, and such large G_{min} make final signal outcomes close to the CFT; lastly, G_{min} of Cons_3 is formulated by a normalized queue length and guarantees the current phase's queue can be cleared to the average queue of other phases. Cons_1 and Cons_2 are preset before the simulation, but Cons_3 updates at the yellow end of every phase during the simulation, which is the time for agents to decide the next phase.

Figure 4.6 illustrates improvement rates under these three constraints compared to the CFT. The improvement results show Cons_3 > Cons_2 > Cons_1. Results of Cons_1 demonstrate it is not as expected that RL algorithms with larger flexibility to explore and exploit will learn better knowledge. Conversely, if we do not constrain a proper G_{min}, agents will switch phases frequently with a small phase duration during initial exploration. Then with running of simulation and increase of traffic demand, more vehicles will queue and even queue up to the start of the road network causing the generated vehicles fail to insert into the simulated network. Meanwhile, many bad experiences will be stored in the replay buffer. The worse experience would cause bad training, leading to choosing bad actions and storing poor experiences. Finally, it forms a vicious cycle. Therefore, with the increase in simulation time, the scenario of Cons_1 gets worse and worse, more than 80% worse than CFT in most indices. In addition, G_{min} of Cons_2 is too large and limits much on the agents. The proposed strategy under Cons_2 can perform well in off-peak periods but is still worse than CFT for heavy traffic flow. Eventually, we find that Cons_3 is the most proper constraint to avoid the vicious circle of bad actions and leave enough flexibility for agents to decide the phase duration and sequence. Such restrictions on RL algorithms can guarantee learning efficiency and effectiveness.

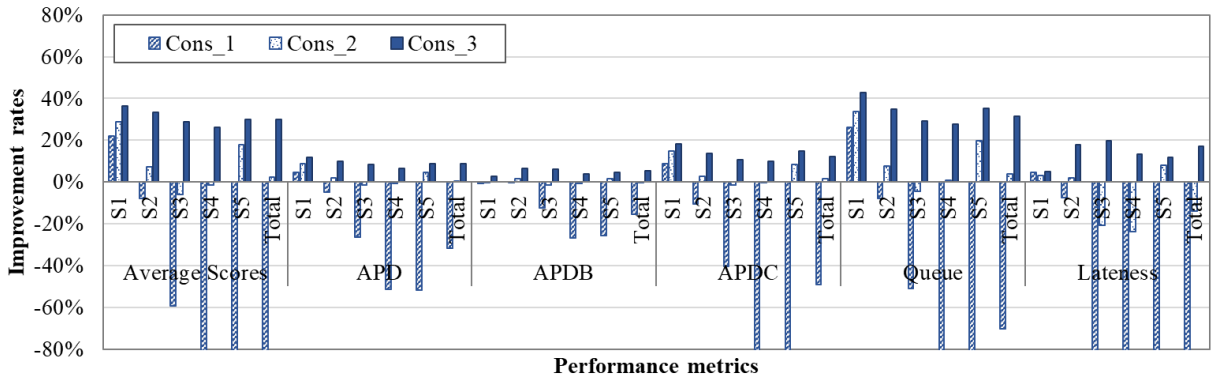


Figure 4.6. The comparison of the CTSPV strategy under different signal constraints.

4.4 Discussions

To clearly demonstrate the proposed algorithm's advantages, we record the signal timing outcomes of the last 10 episodes to analyze its signal timing pattern from cycle length, phase duration, skipping pattern, and action-choosing pattern. To compute the cycle length and skipping counters, we define a base phase sequence, i.e., major through phase P_0 → major left-turn phase P_1 → minor through phase P_2 → minor left-turn phase P_3. Once we observe the phase number descending with the simulation, that phase will belong to a new cycle.

4.4.1 Signal timing pattern

(1) Cycle length

We plot the cycle length of the last episode with the simulation time in Figure 4.7. The cycle length changes all the time due to the phase skipping and varied phase duration. To easily observe the difference between CFT and the proposed method's signal results, we calculated the average cycle length of 10 episodes in each period for the CTSPV strategy, as listed in Table 4.2. It shows that the cycle length of the CTSPV algorithm becomes larger as the traffic demand increase. The cycle lengths of CFT are also added to the last column of Table 4.2. Due to the signal coordination, different intersections share the same length for CFT. Compared with CFT, the cycle length of CTSPV for various periods always is about half. Such smaller cycle lengths may be brought by the shorter phase duration or phase skipping.

(2) Phase duration

For each phase, phase duration is the duration of green time. We calculate the average duration of each phase under different periods. If one phase is skipped, we do not consider it as 0 duration but just neglect it when taking the average. Table 4.3 summarizes the phase duration of different intersections for both CTSPV and CFT strategies under five periods. From the table, the duration of all phases for the CTSPV is smaller than that of the CFT, and both CTSPV and CFT tend to set longer green time for phases with larger traffic volume, for example, the duration of major through P_0 is larger than that of minor through P_2 under the same period; the duration of P_0 under period S3 is larger than under other periods with lower traffic demand. We can also see that the sum duration of all phases, including 12s yellow time for different periods, is always smaller than that period's cycle length. The reason is that there are frequent phase skippings. Finally, we can conclude that the proposed CTSPV tends to make shorter green time and skip phase frequently.

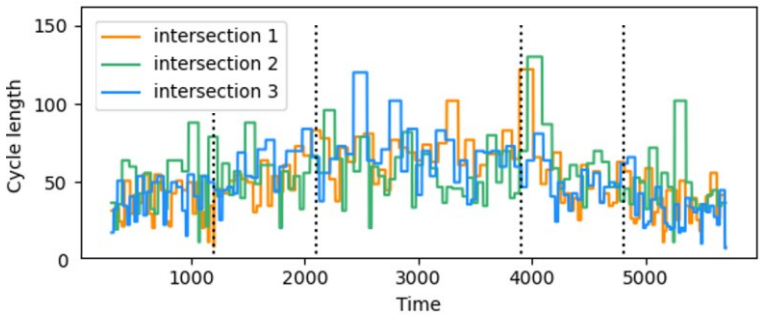


Figure 4.7. The cycle length with simulation time for the last episode.

Table 4.2. The cycle length of the CTSPV and CFT strategies under different periods.

Periods	CTSPV			CFT
	Intersection 1	Intersection 2	Intersection 3	
S1	34	47	34	68
S2	49	53	53	142
S3	67	56	69	146
S4	54	55	52	86
S5	34	47	33	62

Table 4.3. The phase duration of the CTSPV and CFT strategies under different periods.

Intersection	Periods	CTSPV				CFT			
		P_0	P_1	P_2	P_3	P_0	P_1	P_2	P_3
1	S1	11	10	9	9	26	12	9	9
	S2	18	15	9	9	69	35	17	9
	S3	27	18	10	9	71	36	18	9
	S4	21	15	9	9	38	18	9	9
	S5	11	9	9	9	22	10	9	9
2	S1	25	7	9	9	32	6	9	9
	S2	27	7	10	9	89	12	20	9
	S3	29	7	10	9	92	12	21	9
	S4	28	7	10	9	48	6	11	9
	S5	26	7	10	9	26	6	9	9
3	S1	12	10	9	9	26	12	9	9
	S2	20	15	9	9	69	35	17	9
	S3	29	18	10	9	71	36	18	9
	S4	20	14	9	9	38	18	9	9
	S5	11	9	9	9	22	10	9	9

(3) Skipping pattern

a. Number of phase skippings

We count the number of phase skippings in 10 episodes and compute the average number, as demonstrated in Table 4.4. Phases P_0, P_1, P_2, and P_3 account for about 4%, 22%, 31% and 43%, respectively. Hence, agents tend to skip minor roads and left-turn phases, with less traffic flow.

Table 4.4. The number of phase skippings.

Intersection	Number of phase skipping			
	P_0	P_1	P_2	P_3
1	5	25	41	57
2	4	31	34	49
3	5	25	41	55

b. Number of type skippings

In accordance with the queue length and bus priority requests, we can divide the environment states into four types: *Type1*: without queue and bus request, *Type2*: without queue and with bus request, *Type3*: with queue and without bus request, *Type4*: with queue and bus request. Then we also count the number of skips and non-skips for each type. Note that only the situation that can be skipped but not skipped is counted, and the situation that is constrained, i.e., cannot be skipped is ignored. From Table 4.5, the total number of different types shows that most states are in Type 1, and very few are in Type 2. Ignoring Type 2, we can obtain that the frequency of skips is 91%, 56%, and 34% for Type 1, Type 3, and Type 4. Therefore, agents have learned to skip the phases without queue length and bus priority requests (Type 1).

Table 4.5. The number of skips and non-skips for different types.

Intersection	Number of skips				Number of non-skips				Total number			
	Type 1	Type 2	Type 3	Type 4	Type 1	Type 2	Type 3	Type 4	Type 1	Type 2	Type 3	Type 4
1	43	1	74	9	5	1	54	12				
2	31	0	85	2	4	0	75	11	133	3	416	53
3	47	1	72	7	3	0	56	12				

(4) Action-choosing pattern

The phase-changing data is collected to analyze the action-choosing pattern of the CTSPV strategy. Situations that are constrained to only one valid action would not be considered. Hence, the analyzing data records the current phase, and the agent chooses which action from what kind of valid action space. In summary, here are eight situations as illustrated in Figure 4.8. Taking situation I as an example, the current phase is the yellow end of P_0. When P_2 cannot be skipped due to the observed skipping times or queue length, the valid action space is $[P_1, P_2]$; conversely, when P_2 can be skipped, P_3 would also be valid action, so that the valid action space is $[P_1, P_2, P_3]$.

Table 4.6 lists the total number of actions under different situations in 10 episodes, and also gives the corresponding ratio of choosing each valid action. From Table 4.6, we can know that (1) ***Situation I*** P_1 and P_2 are roughly chosen with the same large ratio, while there is only about a 10% chance to select P_3; (2) ***Situation II*** Three phases are equally to be selected; hence we separately observe the choosing ratio under two action spaces to analyze its pattern. As all intersections show a similar pattern, so we just take Intersection 1 as one example to explain here, and its choosing ratios of all situations are listed in Figure 4.8. As the figure shows, the ratio for II-1 is $P_2 = 0.31, P_3 = 0.69$, and for II-2 is $P_0 = 0.57, P_2 = 0.27, P_3 = 0.16$. Therefore, we can find that agents want to go to P_0 as soon as possible so that they will choose the last phase P_3 which is closer to P_0 in II-1, or directly choose P_0 in II-2; (3) ***Situation III*** Agents will not skip P_0 in about 80% of cases; (4) ***Situation IV*** Agents will just switch to P_0 with a likelihood of about 70%.

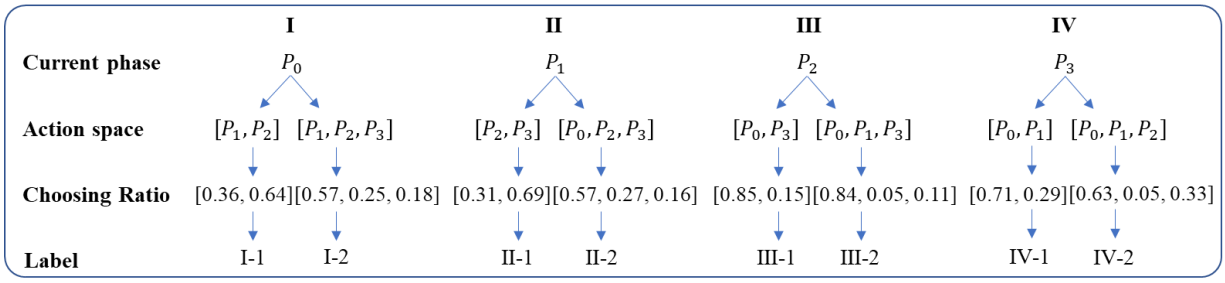


Figure 4.8. The action-choosing situations and ratios of choosing each phase.

Table 4.6. The total number and corresponding ratio of actions chosen in 10 episodes.

Intersection	Situation	Current phase	Number				Ratio			
			P_0	P_1	P_2	P_3	P_0	P_1	P_2	P_3
1	I	P_0	-	198	183	40	-	0.47	0.43	0.10
	II	P_1	173	-	151	195	0.33	-	0.29	0.38
	III	P_2	390	7	-	62	0.85	0.02	-	0.14
	IV	P_3	86	19	26	-	0.66	0.15	0.20	-
2	I	P_0	-	386	267	33	-	0.56	0.39	0.05
	II	P_1	129	-	97	181	0.32	-	0.24	0.44
	III	P_2	354	7	-	56	0.85	0.02	-	0.13
	IV	P_3	119	19	14	-	0.78	0.13	0.09	-
3	I	P_0	-	159	180	52	-	0.41	0.46	0.13
	II	P_1	188	-	154	168	0.37	-	0.30	0.33
	III	P_2	351	14	-	101	0.75	0.03	-	0.22
	IV	P_3	89	17	17	-	0.72	0.14	0.14	-

Only situation I does not have a clear pattern to choose action, so the decision tree model is used to explain the action-choosing behavior in this situation further. Here, the dependent variable is the chosen phase, and the independent variables are the average queue length $Queue_p$ and bus urgency $Prio_p$ of phase p ($p = 1,2,3$) movements. Because three intersections have the same tendency, we just show the results of Intersection 1 as the example in Figure 4.9. When there is no queue or the queue is less than 2.5 m in the movements of P_2, 80% and 65% will not skip the next phase P_1, respectively; when the queue of P_2's movements is larger than 2.5 m but smaller than 6.25 m, agents would like to skip P_1 and switch to queued P_2 with a ratio of 53%; when the queue of P_2's movements is larger than 6.25 m, a much higher ratio, up to 83%, will switch to P_2.

Therefore, we can summarize the action-choosing pattern briefly as follows.

Situation I. Agents tend to choose the next phase P_1; only when the minor through P_2 queues more than one vehicle, they tend to skip the next phase P_1 and switch to P_2.

Situation II. Agents tend to choose P_3 which is closer to major through P_0 when P_0 is not valid (Situation II-1), otherwise directly choose P_0 (Situation II-2).

Situation III. Agents tend to skip the minor left-turn P_3 with less traffic volume and choose the major through P_0 with the heaviest traffic.

Situation IV. Agents tend to choose the next phase P_0.

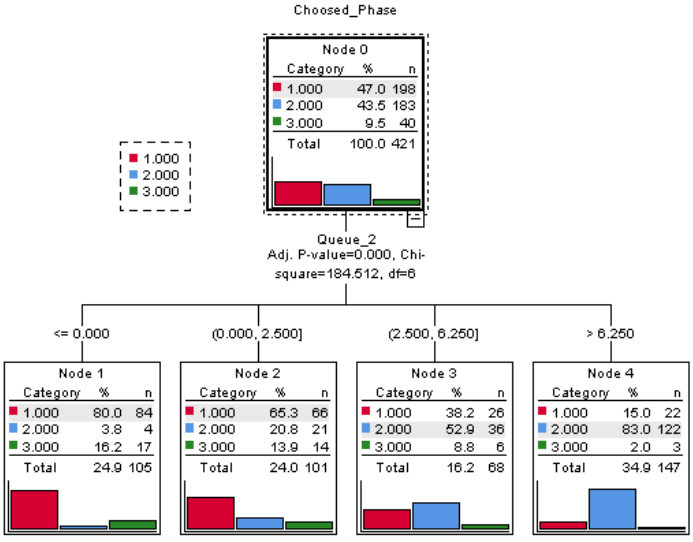


Figure 4.9. The decision tree results of the first intersection in situation I. (Category 1, 2, and 3 refer to P_1, P_2, and P_3. Queue_2 refers to the queue of P_2.)

4.4.2 Generalized rule-based methods

The above action-choosing pattern, summarized from the RL-based CTSPV method, can be utilized as one rule-based skipping strategy, which can decide phases and allow phase skipping. Integrating this rule-based skipping separately with the fixed phase duration as IFTPS by SIDRA, the average phase duration (in *Section 4.4.1(2) Phase duration*) of the CTSPV method, and variable phase duration by F value of Eq (4.10), three rule-based TSP strategies can be generalized, namely Fixed Time with Rule-Based Skipping (FTRBS), Average Duration with Rule-Based Skipping (ADRBS), and F-value with Rule-Based Skipping (FRBS). For the FRBS method, if the F value of the current phase is

the largest one, then keep the current phase; otherwise, switch to yellow. The pseudocode of FRBS is present in Appendix B. Simulations are conducted to evaluate these generalized rule-based methods by comparing them to the RL-based CTSPV method. Table 4.7 illustrates the difference of those comparing TSP strategies, and Figure 4.10 shows their performance. We just present the total performance here because different traffic demand periods show the same tendency.

Table 4.7. The summary of detailed methods for different TSP strategies.

Methods	Phase duration		Phase sequence	
	Fixed	Variable	Fixed	Variable
CFT	Coordinated Fixed SIDRA plan		Fixed SIDRA plan	
IFTPS	Fixed SIDRA plan			F-value
FTRBS	Fixed SIDRA plan			Action-choosing pattern
ADRBS	Average phase duration			Action-choosing pattern
FRBS		F-value		Action-choosing pattern
CTSPV		RL		RL

From Figure 4.10, FTRBS outperforms IFTPS and CFT so that the action-choosing pattern of the RL method is better than the F-value rule and Fixed SIDRA plan, though F-value uses the same input information of the RL method. However, the improvements of IFTPS and FTRBS are tiny compared to ADRBS with the average phase duration of the RL method and other variable phase duration methods. The average phase duration of RL achieves much better results than the fixed one by SIDRA. Hence, the learned knowledge of the RL method, including the average phase duration and action-choosing pattern, is superior to the traditional traffic signal-timing optimization model, like SIDRA. Even though ADRBS deploys the learned knowledge, the duration of phases is fixed for each period, so it improves less than the other two approaches with variable phase duration and variable phase sequences. Additionally, it is surprising that the FRBS approach performs the same well as the RL-based CTSPV.

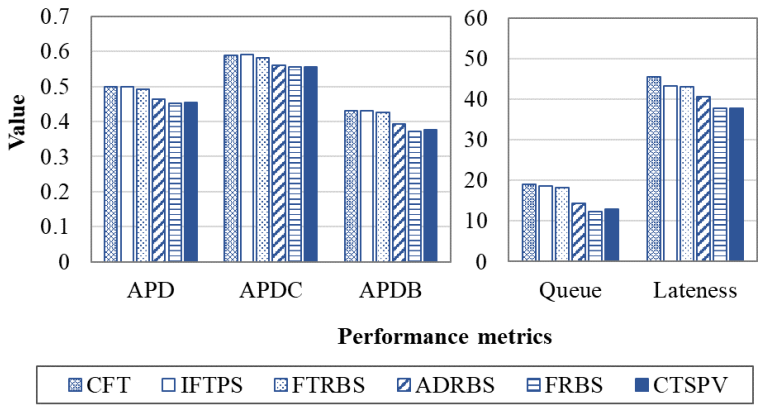


Figure 4.10. The performance comparison of different rule-based strategies. (The unit of APD, APDC, APDB: s/pax, the unit of Queue: m/lane, the unit of Lateness: s/pax.)

4.5 Summary

This chapter proposes a CTSPV algorithm by MARL for the arterial road to improve transit schedule adherence. It flexibly changes phase sequence and duration following real-time traffic conditions considering the trade-off between transits and non-prioritized vehicles, the multiple

conflicting bus requests, and the cooperation between agents. The proposed method utilizes invalid action masking to satisfy traffic constraints and solve the asynchronous decision-making of agents.

Simulation results show that the proposed method excels in algorithm learning and traffic performance owing to three superiorities, i.e., skippings of unurgent phases, agents' cooperation, and the proper credit assignment among agents. It is highly essential to set appropriate signal constraints for the RL method to avoid the vicious cycle of bad actions, poor experiences, and worse training. The best minimum green time is formulated by the normalized queue length due to efficient learning and improved traffic performance. This work records the signal timings of the proposed CTSPV and first analyzes the signal-timing pattern of the RL method. Results show that RL agents tend to have variable cycle length, short phase duration, and frequent phase skipping than traditional traffic control methods. Agents are more likely to skip phases with less traffic demand and without bus priority requests and always choose the phase with heavy traffic demand, like major through phase P_0. More specifically, when the intersection is at the yellow end of the major through phase, the agent would like to choose the subsequent major left-turn phase, or long-queued minor through phase; otherwise, agents tend to go to the major through phase. The results of rule-based strategies generalized by those patterns also prove the good performance of RL-learned knowledge.

Chapter 5

TSP at multiple intersections to improve headway adherence

This chapter extends the work from promoting bus schedule adherence to headway adherence. In the process of changing research objectives, new problems may arise, including how to formulate headway and design proper reward functions to realize goals. Different from Chapter 4, this chapter also considers phase transition loss.

This chapter is organized as follows. Section 5.1 introduces the methodology of this chapter and describes the proposed CTSPVH algorithm. The experiment framework is presented in Section 5.2. Experiment results are shown in Section 5.3. Section 5.4 discusses the limitations of the baseline and the benefits of the proposed approach. Section 5.5 summarizes this chapter.

5.1 Methodology

5.1.1 Problem statement

This work considers an arterial road with N signalized intersections and complex bus routes from different directions. Each intersection is equipped with a multi-phase signal controller with three signals per phase, containing green, yellow, and red. As presented in *Section 2.5 Transit reliability*, headway adherence significantly influence bus reliability and passenger waiting time. Therefore, this work aims to develop a TSP strategy to modify traffic signals in real-time to promote headway adherence based on the observed traffic states. The modification on the traffic signal is regarded as the action, and the improvement of headway adherence is the reward. Therefore, this problem can be modeled as MDP with the objective of decreasing headway deviation and be addressed by MARL. The following describes the four key components of this MARL framework, including agent, state, action, and reward. And the performance evaluation metrics are also presented in the last part.

(1) Agent

The signal controller of each intersection is a learning-based agent. Given the observed state s_t^i of the environment at time step t ($t \in T$), the agent of intersection i ($i \in N$) takes an action $a_t^i \in A_t^i$ (A_t^i is the valid action space) following a policy π and then obtains a reward r_t^i when the environment transits to the next state ${s'}_t^i$ (i.e., s_{t+1}^i). The objective of agents is to find an optimal policy π^* which maximizes the expected return.

(2) State

For intersection i, the state represents four-part information at each time step t, i.e., $s_t^i = (S_{A_t}^{\ i}, L_{Q_t}^{\ i}, P_{C_t}^{\ i}, S_{Bus_t}^{\ i})$.

a. Average speed $S_{A_t}^{\ i}$

$$S_{A_t}^{\ i} = \{S_{Alm_t}^{\ i}\}, l \in \Psi, m \in M_l \tag{5.1}$$

where $S_{Alm_t}^{\ i}$ represents the average speed of movement m at the approach link l for intersection i at time t; Ψ is the approach set, including southbound (SB), northbound (NB), eastbound (EB), and westbound (WB) approaches; M_l is the movement set of approach link l, containing through and left-turn movements for right-hand traffic, in which right-turns are permitted to pass on red.

b. Queue length $L_{Q_t}^{\ i}$

$$L_{Q_t}^{\ i} = \left\{L_{Qlm_t}^{\ i}\right\}, l \in \Psi, m \in M_l \tag{5.2}$$

where $L_{Qlm_t}^{\ i}$ represents the queue length of movement m at the approach link l for intersection i at time t. If a traffic movement occupies more than one lane, e.g., two lanes for through traffic, then we take the maximum queue length of those lanes as the queue length of that movement.

c. Current phase $P_{C_t}^{\ i}$

It is a one-hot encoding vector with several values, each representing one traffic light indication: green, yellow, and all-red time of all phases. The vector consists of 0 in all cells except for a single 1 in a cell used uniquely to represent the current signal situation. For example, if the current signal shows the green time of the second phase in a common four-phase signal scheme, $P_{C_t}^{\ i} = (0,0,0,1,0,0,0,0,0,0,0,0)^T$.

d. Bus state $S_{Bus_t}^{\ i}$

The representation of bus information will determine the effectiveness of RL agents serving bus priority requests on headway adherence, especially in complex situations with multiple conflicting priority requests. In this work, we also collect the distance to the stop line D, schedule delay SD, and passenger occupancy O of buses approaching the intersection as former two chapters. Bus schedule delay is the delay of a bus compared to its schedule, which we set to 0 if the bus is not late than schedule. Passenger occupancy is the number of bus passengers available by the onboard APCs. Moreover, forward headway deviation FHD, backward headway deviation BHD, and whether at bus stops AtS are also included.

Headway deviation is the deviation of the current headway from the scheduled headway, which is an essential indicator to reflect the headway adherence. It is easy to calculate buses' headway deviation when they arrive at bus stops, as the arrival time of all buses at stops are collected, and the headway is the difference between the arrival time of the current bus and the last bus at this stop. However, it is rather complicated to obtain the headway deviation in real-time. In this study, we develop an approach to approximately calculate real-time headway deviation based on the relationships between headway and schedule delay.

Here are two headway deviations, forward headway deviation FHD and backward headway

deviation BHD. Figure 5.1 shows the headway and schedule delay diagram. The green vehicle is the current bus, and the bus in front (or behind) is called the forward (or backward) bus here. Forward (or backward) headway is the headway between the current bus and the corresponding forward (or backward) bus. Forward (or backward) headway deviation is the deviation of forward (or backward) headway from scheduled headway. In this work, it should be noted that each bus entering the network is given an initial schedule deviation ISD. ISD_k denotes the initial schedule deviation of bus k; FH_t^k and BH_t^k are the forward and backward headway of bus k at time t; SH denotes the scheduled headway; FHD_t^k and BHD_t^k are the forward and backward headway deviation of bus k at time t. For each bus line, the real-time headway deviations are computed as follows.

1) When the first bus of each line enters the network:

It should be noted that when the first bus of each line enters the network, there is no forward buses but backward buses have already been scheduled. Therefore, they can be given by

$$FHD_{t_0}^1 = ISD_1 \tag{5.3}$$

$$BHD_{t_0}^1 = ISD_2 - ISD_1 \tag{5.4}$$

2) When the following bus k enters the network:

$$FHD_t^k = BHD_t^{k-1} \tag{5.5}$$

$$BHD_t^k = ISD_{k+1} - ISD_k \tag{5.6}$$

3) When bus k runs in the network:

In Figure 5.1, we describe the distance of buses by time as all buses of one line set the same commercial speed, and distance can be approximated by multiplying the time with the scheduled speed when buses run between two bus stops. From Figure 5.1, we can derive:

$$FH_t^k = FH_{t-\Delta t}^k - \Delta t + \delta_c + \Delta t - \delta_f = FH_{t-\Delta t}^k + \delta_c - \delta_f \tag{5.7}$$

where $\delta_c = \Delta t - \frac{\Delta D_c}{V}$, $\delta_f = \Delta t - \frac{\Delta D_f}{V}$; δ_c and δ_f are the delay of the current and forward buses in Δt; ΔD_c and ΔD_f are the traveled distance of the current and forward buses in Δt; V is the pre-set commercial speed of buses, 20km/h. The green bus of Figure 5.1 actually runs ahead of the expected location after Δt, so δ_c is negative and has a positive sign in the above formula.

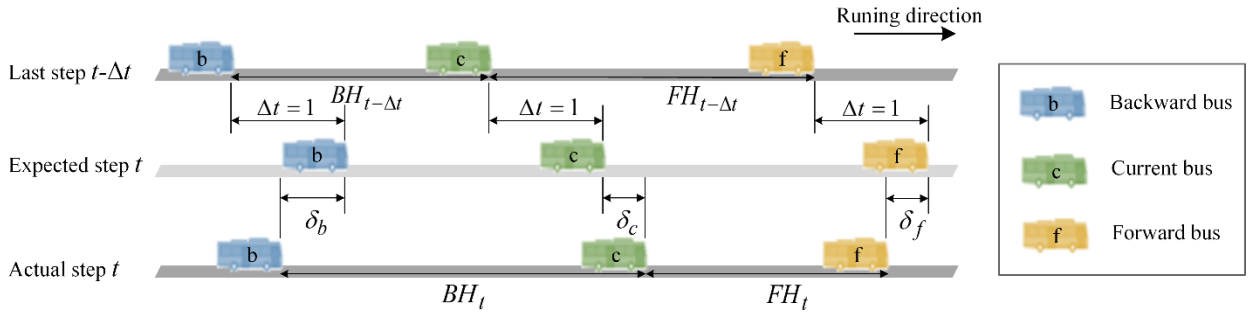


Figure 5.1 The headway and schedule delay diagram.

According to the definition, we know

$$FHD_t^k = FH_t^k - SH \tag{5.8}$$

Based on Eq (5.8), Eq (5.7) can be derived to

$$FHD_t^k = FHD_{t-\Delta t}^k + \delta_c - \delta_f \tag{5.9}$$

Similarly, the calculation of backward schedule deviation can be derived as follows:

$$BHD_t^k = BH_t^k - SH \tag{5.10}$$

$$BH_t^k = BH_{t-\Delta t}^k - \Delta t + \delta_b + \Delta t - \delta_c = BH_{t-\Delta t}^k + \delta_b - \delta_c \tag{5.11}$$

$$BHD_t^k = BHD_{t-\Delta t}^k + \delta_b - \delta_c \tag{5.12}$$

where $\delta_b = \Delta t - \frac{\Delta D_b}{V}$; δ_b is the delay of the backward buses in Δt; ΔD_b is the traveled distance of the backward buses in Δt.

In addition, the relationship between forward and backward schedule deviation can be described as

$$FHD_t^k = BHD_t^{k-1} \tag{5.13}$$

Hence, we can calculate real-time BHD by Eq (5.12) and use Eq (5.13) to obtain FHD when buses run in the network. For the metric AtS, many sensors are installed at bus stops that can detect when a bus has arrived and also the GPS can retrieve the real-time location of buses to verify whether buses are at bus stops. $AtS = 1$ if the bus is at stops; otherwise, $AtS = 0$.

Finally, the bus state is denoted by six-tuple information of buses arriving at intersections:

$$S_{Bus} = \{D_{pb}, SD_{pb}, O_{pb}, FHD_{pb}, BHD_{pb}, AtS_{pb}\}, p \in \mathrm{P}, b \in \Gamma \tag{5.14}$$

where D_{pb}, SD_{pb}, O_{pb}, FHD_{pb}, and BHD_{pb} represent the distance to the stop line, schedule delay, passenger occupancy, forward headway deviation, and backward headway deviation of bus b, which requests the green time of phase p. AtS_{pb} denotes whether bus b controlled by phase p is at bus stops. P is the phase set, such as $P = \{0,1,2,3\}$ in a four-phase signal scheme. It should be noted that the bus state only considers buses that have already arrived at intersections. We define the arrival of one bus as 1) its distance to the stop line is less than 40 m, or 2) it queues at the intersection. Γ is the bus set, and here we set $|\Gamma| = 3$. Thus, if there are more than three arrived buses detected at an intersection and requesting phase p, we only put the information of the three most urgent buses in S_{Bus} for phase p. The urgency is measured by $\frac{|FHD_{pb}|+|BHD_{pb}|}{D_{pb}+\delta}$ where $\delta = 1$ to avoid invalid calculations because we think buses with larger headway deviation and closer to the stop line are much more urgent to consider. Therefore, we can simultaneously consider up to 12 bus requests from different directions at each intersection. Finally, S_{Bus} is a 12x6 matrix.

(3) Action

Consider a series of traffic phases allowing phase skipping. The valid action space A_t^i is a set of valid signal phases for intersection i at each step t. The validity is determined by signal constraints (namely, the minimum and maximum green time) and skipping rules of Chapter 4. Therefore, agent i selects next step's phase from A_t^i as action a_t^i each step. In this case, the phase duration and sequence are totally determined by agents.

The signal constraints would cause asynchronous decision-making time of different agents as agents are restricted to experience yellow time and all-red time when the last action is to change to another phase. Such asynchrony would increase the difficulty of cooperation among agents as cooperation requires agents to consider each other's actions when they make decisions. To address the

asynchrony of agents, we also treat the yellow time and all-red time as an action so that each agent can select one action at every step. Then the universal action space is a set of the green, yellow, and all-red of all considered phases. The valid action space is a subset of them.

In this study, we set 3s yellow time and 1s all-red time before transitioning to another phase to guarantee the phase transition safety and also consider the phase transition loss.

(4) Reward function

This work seeks the optimal policy that can 1) prioritize transits to promote bus headway adherence while bringing less detrimental effects (e.g., queue length) on non-transit vehicles, and 2) improve the general traffic efficiency when there are few buses. Considering the cooperation among agents and their co-efforts on the network performance, all agents utilize a joint reward r_t. Hence, the reward r_t^i is defined as the weighted sum of queue density and headway gains, given by Eq (5.15).

$$r_t = w_1 \frac{\sum_l \sum_\varphi L_{Ql\varphi t}}{\sum_l \sum_\varphi Det_{l\varphi}} + w_2 \sum_k \phi_k \left(FHD_t^k - BHD_t^k \right) \tag{5.15}$$

where $L_{Ql\varphi t}$ and $Det_{l\varphi}$ represent the queue length at time t and detector length on the lane φ of the approach link l in the whole network, respectively; $\phi_k = \begin{cases} 1, & \text{action=the green phase for bus } k \\ -1, & \text{others} \end{cases}$ and bus k belongs to buses in S_{Bus}; w_1 and w_2 are a negative weight for queues of general traffic and a positive weight for headway gains of buses, respectively.

Here we take both forward and backward directions into consideration in headway gains. Note that, according to Eq (5.8) and Eq (5.10), the positive value of FHD and BHD denote the larger headway than scheduled one. It is positive to select the green phase for bus k when bus k has a larger forward headway than scheduled headway as prioritizing bus k benefits on shortening the distance with the forward bus. In contrast, it is harmful to select the green phase for bus k when bus k has a larger backward headway than scheduled headway as prioritizing bus k enlarges the distance with backward bus and deteriorates headway adherence; vice visa.

(5) Evaluation metrics

We evaluate the MARL-based TSP strategies regarding learning efficiency and traffic performance, as in Chapter 4. And we also introduce several new indexes of traffic performance:

- *Average headway deviation (AHD)*: It is the average headway deviation of each bus, computed by

$$AHD = \sum_t \sum_k \left| FHD_t^k \right| / \sum_t |K_t| \,, k \in K_t \tag{5.16}$$

  where K_t is the set of buses running in the network at time t and $|K_t|$ represents the total number of buses.

- *Average bi-headway difference (ABHD)*: This term shows how even the forward and backward headway is. It is defined as the average difference between forward and backward headway deviations:

$$ABHD = \sum_t \sum_k \left| FHD_t^k - BHD_t^k \right| / \sum_t |K_t| \,, k \in K_t \tag{5.17}$$

- *Average person waiting time (APWT)*: This belongs to person-based metrics and is calculated by:

$$APWT = \sum_t wn_t \,/\, \sum_t in_t \tag{5.18}$$

  where wn_t denotes the number of persons waiting for buses at time t, and its sum in time T equals the waiting time of all persons; in_t denotes the number of inserted people at time t and its sum in time T equals the total number of inserted persons in the network.

- *Average person running delay on bus (APRDB)*: This is also a person-based metric to reflect the running performance of buses, which can be computed by

$$APRDB = \sum_t \sum_k (d_{kt} * O_{kt}) / (\sum_t in_t - wn_T) \quad (5.19)$$

where wn_T denotes the number of persons waiting for buses at the last time step T. The numerator is the total running delay of all passengers on buses, and the denominator is the total number of inserted passengers in time T.

- *Total person delay on bus (TPDB)*: The total person delay consists of waiting delays at bus stops and running delays on buses. Hence, we can calculate $APWT$ by roughly summing $APWT$ and $APRDB$ together.

5.1.2 CTSPVH algorithm

This work utilizes the same MARL framework of Chapter 4 to propose the *Cooperative TSP strategy of Variable phase for Headway adherence* (CTSPVH). It is also based on QMIX and D3QN, and its pseudocode is the same as the CTSPV algorithm. The only difference with the CTSPV approach is the size of convolutional (Conv) and fully connected (FC) layers for approximating Q value due to the different representations of state, action, and reward. Figure 5.2 shows the CTSPVH framework, using DNN to approximate the Q value of each agent and mixing network to obtain the global Q value.

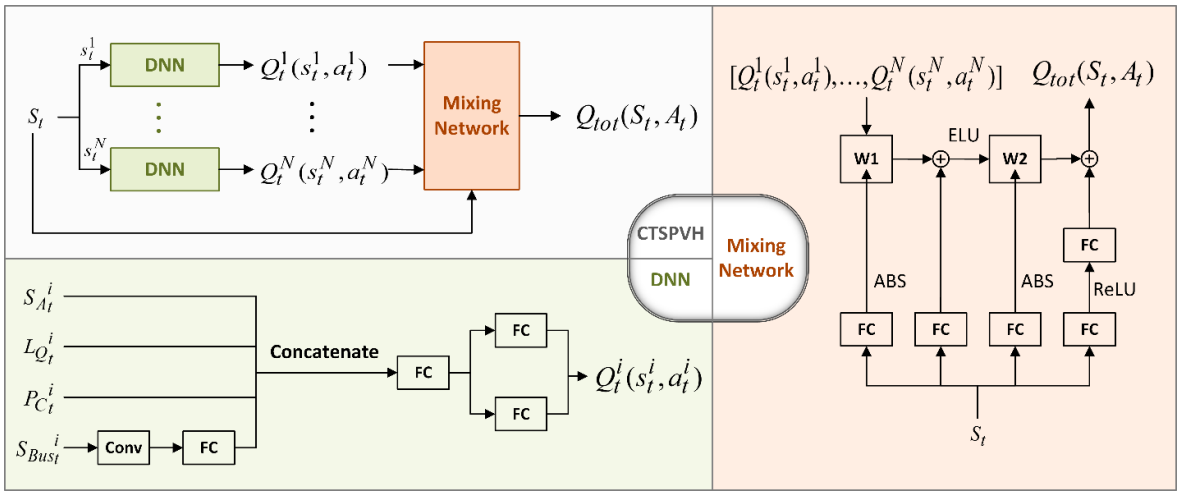


Figure 5.2. The CTSPVH framework.

5.2 Experiment

5.2.1 Simulated environment

To assess the performance of the proposed CTSPVH strategy, we simulated a three-intersection arterial by SUMO as the environment to interact with RL agents, as shown in Figure 5.3. Each intersection has four legs, of which EB and WB approaches on the major roads have three lanes, and SB and NB approaches on minor roads have only two lanes. The spacing between two adjacent intersections is 450m.

Traffic demands: The simulated network has eight inflows and eight outflows, so 56 OD pairs are set. We consider 5-period demands, of which the flow ratios are around 0.35, 0.5, 0.55, 0.45, and 0.3, respectively. The total simulation time is 5700 s, in which the warm-up is 5 mins, and the time of S1~S5

are 15, 15, 30, 15, and 15 mins, respectively.

The environment has six two-way bus lines, as shown in Figure 5.3 (the same as Figure 4.2). To determine bus schedules for better studying headway adherence, related literature is summarized in Table 5.1. Referring to those studies, we set a 5-min headway for S2~S4 with higher demand and a 10-min headway for S1 and S5 with lower demand.

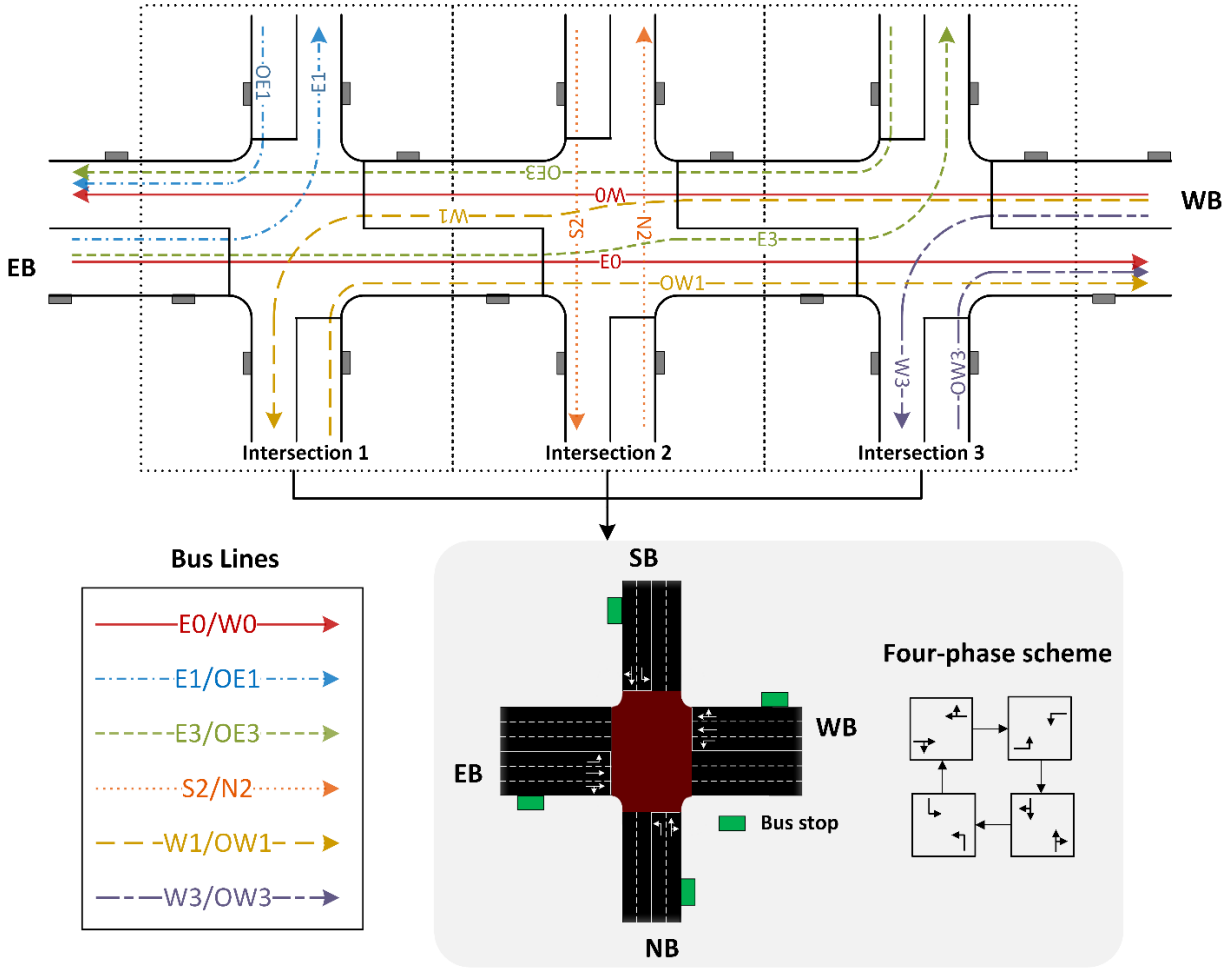


Figure 5.3. Simulated environment.

Table 5.1. Summarization of scheduled headway of related literature.

Paper	Scheduled headway
Van Oort et al. (2010)	5 mins
Anderson and Daganzo (2020)	10 mins
Long et al. (2020)	1.5~3 mins
Khan et al. (2023)	2~20 mins
Wang and Sun (2020)	8 mins
Zhou et al. (2022)	5 mins

Traffic signal phase: The traffic signal of each intersection adopts a standard four-phase scheme as shown in Figure 5.3, namely, major through phase P_0, major left-turn phase P_1, minor through phase P_2, and minor left-turn phase P_3. All right turns are allowed to pass on red. We emphasize there is a 1s all-red time after 3s yellow and before the green time of other phases.

Bus settings: All buses are given one random initial schedule deviation, following a normal distribution $N(0,180^2)$, so that the inserted time of each bus is determined by the sum of scheduled headway and that initial schedule deviation. The capacity of each bus is 70 pax, and the boarding time per person is 2 s. At each stop, the passenger arrival rate of each line is 0.0083 person/s for S1 and S5 and 0.0167 person/s for other periods.

5.2.2 Compared methods

We compared the proposed CTSPVH approach with the following baselines. Parameters settings are shown in Table 5.2.

(1) *Coordinated Fixed-Time signal* (CFT): We obtain the optimal fixed signal timing considering arterial coordination by SIDRA INTERSECTION, as shown in Table 5.3. As Figure 4.3 in Chapter 4 shows, CFT provides sufficient green bandwidth and coordinates well.

(2) *CTSPV*: This is a cooperative TSP strategy by MARL framework for arterial roads to improve schedule adherence, referred to *Section 4.1.2 CTSPV algorithm*.

(3) *Bus Holding strategy (BH)*: We refer to the work of Daganzo (2009) and use this strategy to hold buses with small headway at bus stops (see Appendix). Traffic signals at all intersections are controlled by CFT.

Table 5.2. The parameters settings.

Parameters	Value
Discounting factor	0.99
Learning rate	0.0001
Epsilon of ϵ-greedy	Exponentially decade from 0.6 to 0.1 with a rate 5e-5
Target networks replace iterations	2000
Size of the replay buffer	20000
Size of minibatch	64
Number of episodes	160
w_1 and w_2 for the reward	-1 and 1/5120

Table 5.3. Signal timings of CFT.

Intersection	Periods	Green time (s)				Yellow time (s)	All-red time (s)	Cycle time (s)	Offset (s)
		P_0	P_1	P_2	P_3				
1	S1	26	12	9	9	3	1	72	0
	S2	69	35	17	9			146	
	S3	71	36	18	9			150	
	S4	38	18	9	9			90	
	S5	22	10	9	9			66	
2	S1	32	6	9	9			72	35
	S2	89	12	20	9			146	
	S3	92	12	21	9			150	
	S4	48	6	11	9			90	
	S5	26	6	9	9			66	
3	S1	26	12	9	9			72	2
	S2	69	35	17	9			146	70
	S3	71	36	18	9			150	70
	S4	38	18	9	9			90	70
	S5	22	10	9	9			66	8

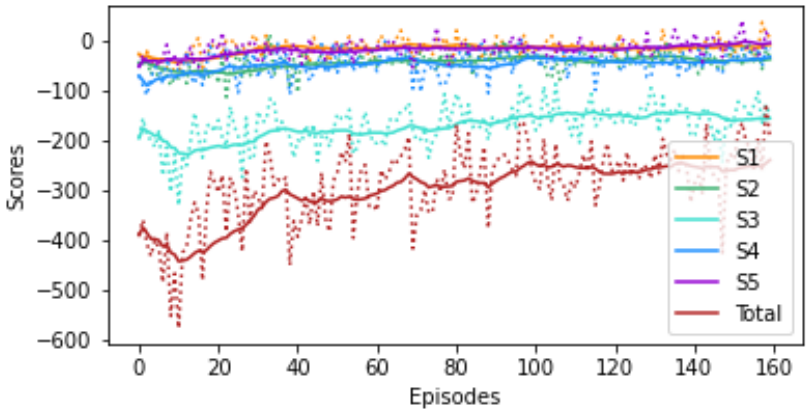
5.3 Results

This section assesses the proposed CTSPVH approach and compares it with other baselines. Traffic simulation of each episode sets different random seeds, facilitating testing varied traffic situations. Simulation results show the learning efficiency and network performance of the proposed method.

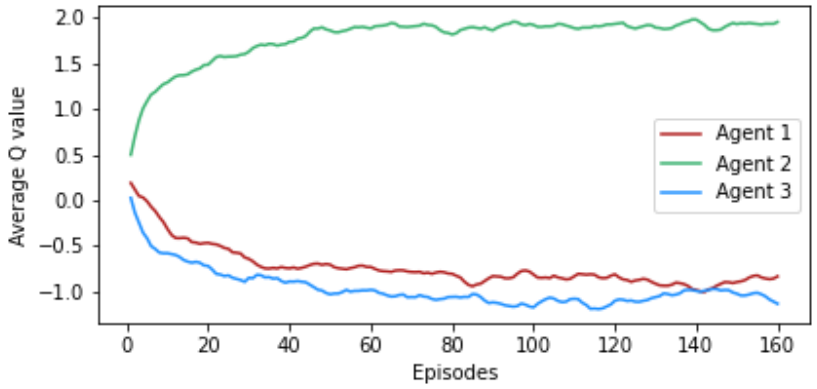
5.3.1 Algorithm learning

Learning efficiency can be clearly obtained from the learning curve. Here, two indicators are computed to plot the learning curve: scores and average Q value. Scores are the cumulative rewards in one episode. The average Q value is an average of 64 Q values, which is the Q value of the best action selected for a given 64 states under the current policy. As agents' policies are updated with learning, changes of scores and average Q value illustrate algorithm learning performance. The results are shown in Figure 5.4.

In Figure 5.4 (a), dotted lines are actual scores with episodes, and solid lines are average scores per 20 episodes, making them more stable than dotted ones. The scores of the peak demand (S3) are lower than off-peaks (S1,2,4 and 5), and score improvements of peak demand are much larger than off-peaks with the training process. After 50 episodes, total scores become much more stable, which illustrates the algorithm roughly converges. In addition, Figure 5.4 (b) shows varieties of the average Q value with the simulation time for different agents. Because the average Q value is obtained by conditioning on given 64 states all the time without any randomness, its curve does not oscillate much as scores. Hence, curves of average Q value are more intuitive to find that agents roughly converge after 50 episodes, and each agent has learned policies from centralized learning. It should be noted that the decreases in agents 1 and 3's average Q value with training do not mean agents 1 and 3 are learning worse. The given 64 states may be bad situations for agents 1 and 3, and those two agents should obtain negative values, but agents don't learn enough knowledge at the beginning of training so they obtain inaccurate positive Q values. Then agents learn with time and gradually obtain the accurate negative Q value. Hence, we should focus on the tendency of Figure 5.4 (b), which show the learning effectiveness.



(a) Scores. (S1~S5 refers to the five periods of traffic demands, and the Total is the sum of all five periods)



(b) Average Q value

Figure 5.4. The learning curve of the CTSPVH strategy.

5.3.2 Performance comparison

The learning curve shows that the proposed algorithm has converged after 50 episodes. Hence, we reported average performance metrics of the last 60 episodes for performance evaluations. Figure 5.5 summarizes the performance improvement rates of various approaches compared with the CFT method. BH reduces AHD, ABHD, and APWT by 17.5%, 26.1%, and 16.4%, respectively, which are the highest rates among all methods, but it also seriously increases APD, Queue, and APRDB by 10%, 9.5%, and 45%, respectively. BH holds buses with small headway to improve headway adherence and reduce person waiting time at bus stops while holding behaviors delay buses and passengers in buses heavily. Overall, the delay of transit users increases, namely TPDB, by 14.8%. Therefore, the implementation of BH improves transit headway adherence with severe adverse impacts on general traffic and running delay of buses. Detail reasons are discussed in *Section 5.4.1 Limitations of the BH method*. In contrast, CTSPVH significantly decreases AHD and ABHD by 6.5% and 5.3%, without any negative influence on general traffic. For example, it even slightly reduces APD, Queue, APRDB, and TPDB by 0.5%, 2.7%, 2.7%, and 2.3%, respectively. The CTSPV approach is designed to promote bus schedule adherence and decrease queue length; hence, it improves lateness and queue the most, i.e., by 6.6% and 25.8%, respectively.

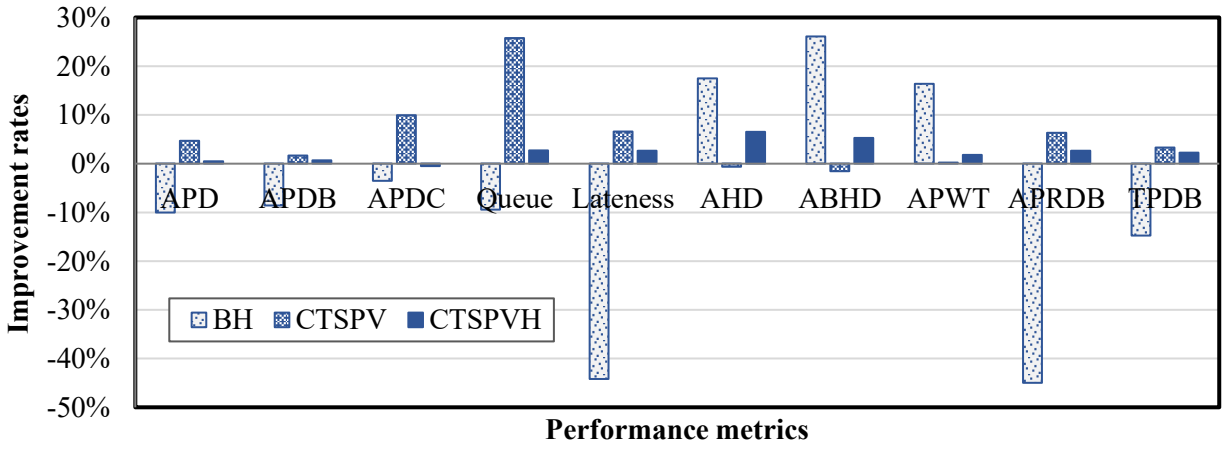


Figure 5.5. The comparison of performance improvement rates.

In summary, BH minimizes headway deviation, but delays and queues increase dramatically; CTSPV minimizes queues and delays significantly, but has no improvement on headway adherence; however, CTSPVH significantly improves headway adherence while ensuring nonnegative impacts on queue and delays.

5.3.3 Weight setting

To facilitate practical application, we need to find proper weights to formulate the reward function. In Eq (5.15), we can set $w_1 = -1$ and test various values of w_2. Figure 5.6 shows the performance comparison when w_2 varies from 1/12000 to 1/2400. As the formulation, the higher w_2 is, the more emphasis is on headway gains compared to queue length. Hence, results of larger w_2 show a smaller decrease in delay, queue, and lateness, but greater improvements in headway deviation and person waiting time. As the changes of APRDB are always larger than APWT with the changes of w_2, the TPDB improvement would change correspondingly with APRDB.

This work aims to minimize headway deviation without negative impacts on general traffic, including delays and queues. Therefore, we need to find a balance between those two parts. We obtain the CFT approach's queue and AHD (also equals the average FHD and BHD) are 22.88 m/lane and 195.23 s, respectively. Thus, the average queue density (the first part of reward function Eq (5.15)) in a 300-m length would be 22.88/300 = 0.07627, and the average bi-headway difference (the second part of reward function Eq (5.15)), i.e., average value of $|FHD - BHD|$, would fall into [0, 390.46] as $|FHD| - |BHD| \leq |FHD - BHD| \leq |FHD| + |BHD|$. When balancing queue density and headway gains and putting equal emphasis on them, we can set $w_1 = -1, w_2 = \frac{\text{average queue density}}{\text{maximum of average bi-headway diferrence}} = \frac{0.07626}{390.46} \approx \frac{1}{5120}$. As Figure 5.6 shows, in this case, CTSPVH can reduce AHD, ABHD, and APWT by 6.5%, 5.3%, and 1.8%, while APD, Queue, Lateness, APRDB, and TPDB also decrease by 0.5%, 2.7%, 2.6%, 2.7%, and 2.3%.

In practice, after collecting average queue length and headway deviation, balancing weights can be determined to perform well in headway adherence without negative impact on general traffic.

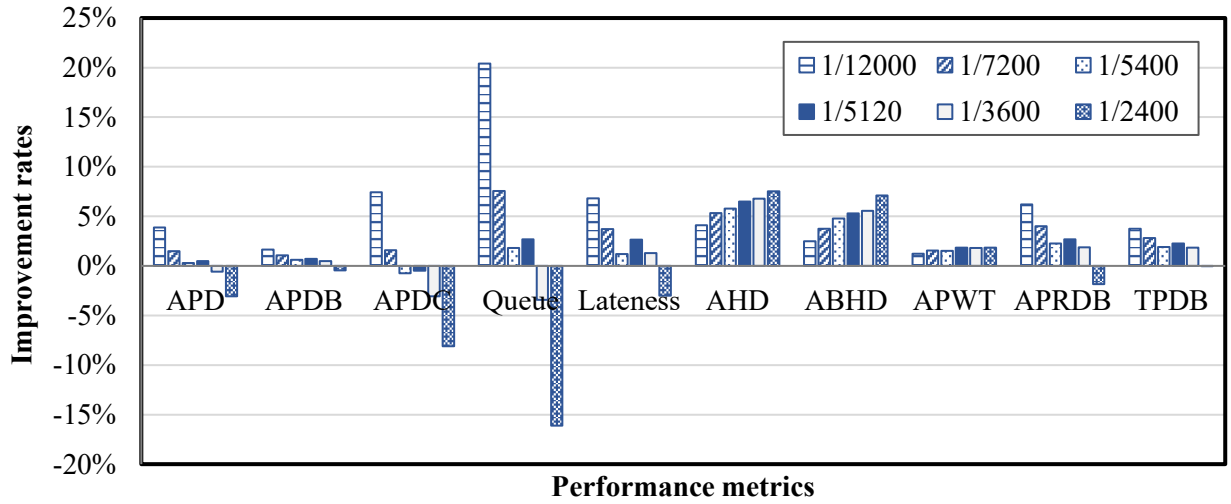


Figure 5.6. Performance comparison of different weight settings.

5.4 Discussions

This section focuses on the limitations of BH and the benefits of CTSPVH to answer the following

two questions: one is why does BH perform so poorly on delay in this study, and the other one is why does CTSPVH improve a little on APWT?

5.4.1 Limitations of the BH method

Bus holding strategy has been widely implemented to improve transit reliability; however, in this experiment, simulation results show that the BH method improves headway adherence and person waiting time significantly but imposes greater costs on bus delay, car delay, and queue. According to the principle of bus holding strategy, we can infer that the bus delay and person delay on buses would increase a little after holding buses at a stop, but the simulation results are much worse than we expected. By visualizing simulations, we found a severe problem, i.e., heavy queueing after bus stops due to the bus holding strategy, as shown in Figure 5.7.

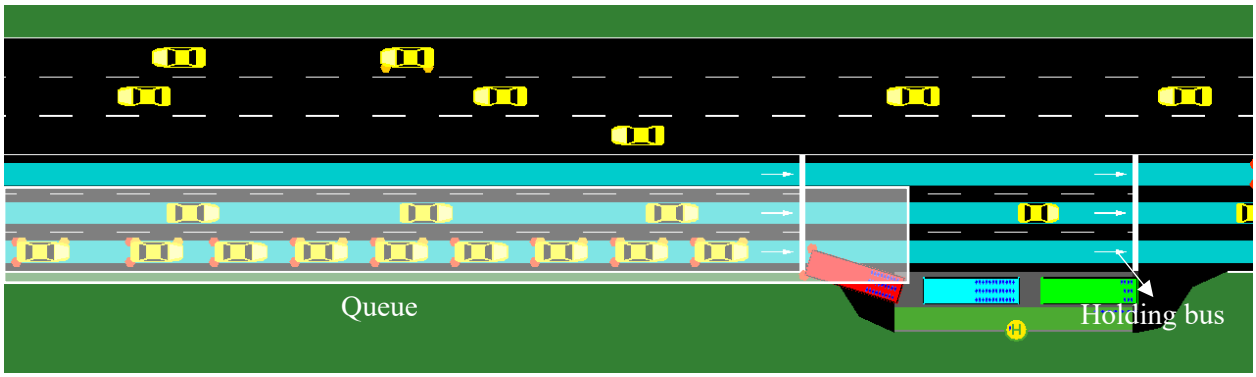


Figure 5.7. Queuing after bus stops due to bus holding strategy.

This problem happens when the length of the bus stop bay is limited, and one ahead holding bus occupies this bay area; other following buses scheduled to use the same stop would queue and even influence car movements. The queue of following buses and cars results in increased delays for both bus and car travelers and decreased bus service reliability. Moreover, after that holding bus finished its holding time, the following buses of other lines may also need to hold at this stop, so that the queue is not easy to clear due to the holding strategy of a large amount of bus lines. In reality, we usually focus on one the most key bus line or route with high passenger demand or serious unregular headway to consider implementing bus holding. Then, those essential buses can benefit a lot from the holding strategy with fewer adverse effects on other vehicles. Overall, there are two reasons for this queueing problem: the limited space of bus stop bays and the large amount of holding bus lines.

Consequently, the above two reasons for the BH strategy's poor performance are its limitations in real traffic; namely, the BH strategy can only be applied to networks with long bus stop bays and for a limited number of bus lines.

5.4.2 Benefits of the CTSPVH method

Simulation results do not show a significant improvement of the CTSPVH approach on APWT compared to the CFT approach, only 1.8%, even though it greatly decreases headway deviation by 6.5%, which seems inconsistent with the advantages of headway adherence on reducing person waiting time. To answer this question, we plot three bus trajectories of one line to analyze possible reasons.

As Figure 5.8 shows, X-axis denotes the time, and Y-axis represents the space. One bus line has many bus stops with an intersection between two stops. The studied network with three intersections

only contains around four stops from i to $i+3$ for most lines like Line E0/W0, other stops after $i+3$ is the downstream of the studied network. For simplification, we do not plot the queue time of buses before stop lines at signalized intersections and just change the slopes of bus trajectories between two stops. For example, if a bus queue a long time before the stop line to wait for the green time, the slop of corresponding trajectories between two stops, including this intersection, becomes much smaller; otherwise, the slop becomes larger when a bus is given a signal priority and pass this intersection quickly. There are three bus trajectories: red dotted lines are scheduled bus trajectories, black solid lines are bus trajectories controlled by CFT without TSP, and green dash lines are bus trajectories with CTSPVH strategy. We assume that the first bus travels on schedule, and the initial schedule deviation set for the second (third) bus is positive (negative), so the second (third) bus enters the studied network with large (small) headway.

For the second bus without TSP control, the large headway may cause more person arrivals and enlarge its dwell time. Then headway gradually increases, and the situation becomes increasingly worse. However, in an ideal situation, the CTSPVH strategy can always give signal priority to this bus so that it travels faster at every intersection and arrives at the next bus stop earlier than without TSP control. A smaller headway deviation than without control would shorten the dwell time and further reduce headway. After TSP of several intersections, headway will gradually decrease and eventually reach the scheduled one, thus can bringing the bus trajectory back to the scheduled one.

In contrast, for the third bus without TSP control, the small headway at the beginning will make the headway shorter and shorter. This third bus even catches on the second bus at stop $i+3$, and those two buses bunch together. However, the CTSPVH can delay this bus a bit at intersections by giving the signal to other movements. After a few intersections, such headway deviation can also be corrected.

As we mentioned in *Section 2.5*, the headway irregularities and bus bunching problems can lead to a large person waiting time, but these conditions are gradually exacerbated downstream (the gray area of Figure 5.8) with time. Because our studied network contains only one short arterial road with three intersections (the white area of Figure 5.8), the negative impacts of headway irregularities do not affect passenger waiting time too much, and the effect of early and late arriving buses on passenger waiting time is somewhat neutralized, so the improvements of CTSPVH strategy would not be so significant. The improvement of APWT by the CTSPVH approach would be larger if the studied environment covered the whole bus line because it can significantly improve the headway deviation, as the results in the studied network show.

Additionally, the TSP strategy enables to process buses whose headways are small and large, while the BH strategy can only hold small-headway buses. This is another benefit of the CTSPV method. Compared to the two limitations of the BH strategy, CTSPV also has no limits on bus stop bays and has the ability to serve various bus lines.

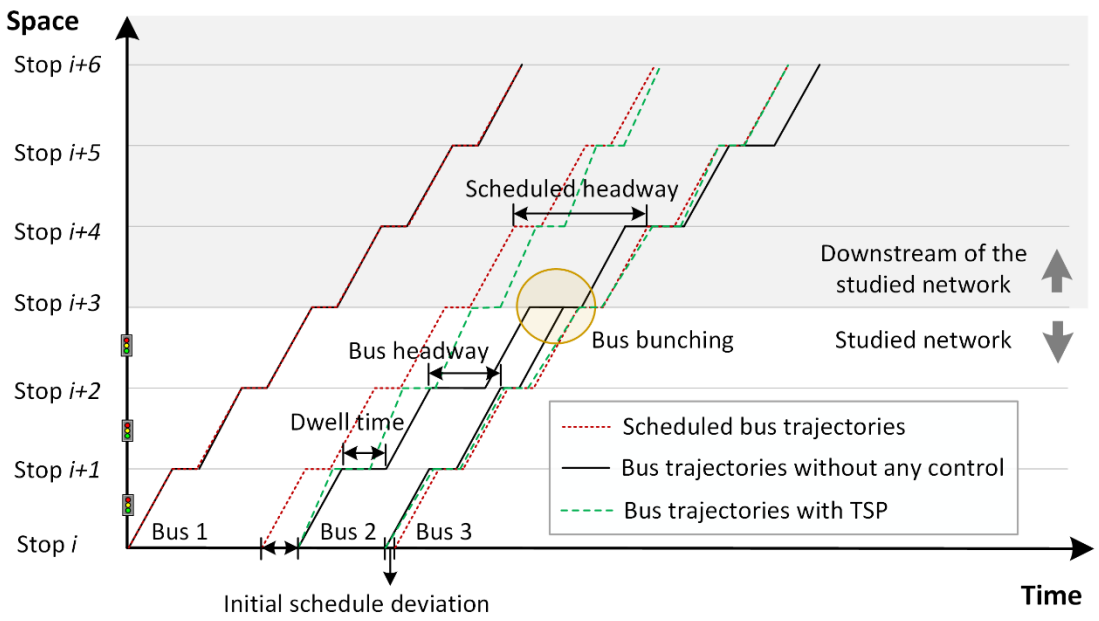


Figure 5.8. Bus trajectories of one line.

5.5 Summary

This chapter proposes a CTSPVH algorithm by Multi-agent RL (MARL) for the arterial road to improve transit headway adherence. It determines traffic signals of the next step based on real-time conditions of transits and non-transits, considering multiple conflicting bus requests, rational action constraints, a balance between buses and cars, and collaboration among intersections.

Simulation results show that the proposed method outperforms fixed-time signal and bus holding strategy when phase transition losses are considered due to greater improvements in transits and less harm to general traffic. The reward function's weight determines the focus of the proposed algorithm: the higher the weight of headway gains is, the more emphasis is on the transit headway, and the more significant improvements are achieved in headway adherence but with more negative effects on queue and delays. When the weight of the headway part sets to be the quotient of average queue density and average headway deviation, the CTSPVH algorithm decreases headway deviation significantly (i.e., by 6.5%) without negative effects on general traffic (i.e., APD decreases by 0.5% and Queue decreases by 2.7%) comparing to CFT method. Then we discussed the harm of the BH method due to limited bus bay length and many holding lines. Holding buses would block following buses that are expected to use the same stop and even cause queues of cars behind. In contrast, the benefits of CTSPVH were also discussed. Although the APWT at the studied intersections decreased slightly, the improvement would be more significant downstream. Due to these existing and potential advantages, the proposed method has a promising application in practice.

Chapter 6

Conclusions

6.1 Summary

This thesis presented three RL-based TSP strategies, ranging from isolated intersections to arterial roads with multiple intersections to improve transit reliability in terms of schedule and headway adherence, respectively.

Chapter 3 proposed an eD3QNI algorithm by single-agent RL for the isolated intersections. This algorithm considers multiple conflicting bus priority requests, adopts a person-based reward function, and utilizes invalid action masking to incorporate traffic domain knowledge (i.e., traffic signal constraints and phase skipping rule) with RL to guarantee action rationality. Simulation results showed the benefits of the **eD3QNI method** over fixed-time signal, active TSP strategy, and other common RL methods, invalid action maksing strategy over variable decision point method, and variable phase scheme over fixed sequece scheme in terms of the training process, convergence speed, and improvement in traffic-related metrics. The impacts of CV penetration rates were discussed, and found it would not influence the proposed algorithm's convergence efficiency but affect traffic performance. At the end of the chapter also discussed the performance of different reward functions and verified that the proposed eD3QNI algorithm could accommodate other specific reward functions to realize different operation goals.

Based on studies of single-agent RL on isolated intersections, Chapter 4 proposed a **CTSPV algorithm** by MARL for the arterial road to improve transit schedule adherence. It adopted the QMIX framework in which every single agent also used the eD3QNI algorithm, invalid action masking strategy to satisfy traffic constraints and solve the asynchronous decision-making of agents, and variable phase scheme to flexibly change phase sequence and duration following real-time traffic conditions. The reward function was designed to consider the trade-off between transits and non-prioritized vehicles. Simulation results proved the superiorities of the CTSPV algorithm and the essentials of appropriate signal constraints. Then signal timing patterns of the CTSPV algorithm were discussed, and found the tendency of variable cycle length, short phase duration, and frequent phase skipping to higher demand phases for RL agents compared to traditional traffic control methods. Evaluation results of rule-based methods generalized from those patterns prove the good performance of RL-learned knowledge.

Extending the above studies, Chapter 5 proposed a **CTSPVH algorithm** by MARL for the arterial road to improve transit headway adherence. It utilized the same framework as the CTSPV algorithm but designed specific state representation, action space, and reward functions to consider transit headway and phase transition loss. Simulation results show that the proposed algorithm reduces headway deviation significantly without adverse influence on non-transit vehicles, which is superior to the coordinated fixed-time signal and bus holding strategy. Also, we analyzed the impacts of the reward function's weight on performance. It found that since the CTSPVH algorithm performs optimally

considering the trade-off between transits and cars, the weight of the headway part is better to set as the quotient of average queue density and average headway deviation. Finally, the limitations of bus holding strategy and the benefits of CTSPVH were discussed to further emphasize the promising application of the proposed method in promoting bus reliability in practice.

6.2 Contributions

The contribution of this thesis can be summarized according to the three main works above.

The first work (in Chapter 3) about the eD3QNI algorithm at isolated intersections contributes threefold.

- The eD3QNI algorithm can incorporate traffic engineers' knowledge (domain knowledge) into the RL to guarantee action rationality and improve the agent's learning efficiency and traffic performance. Specifically, the employment of the so-called IAM strategy provides a way to integrate human engineering knowledge into the development of an RL scheme. By doing so, the proposed method not only guarantees the rationality for the application of DRL in terms of considering the minimum and maximum green times and phase skipping rule, but also aims to improve person-based performance by prioritizing buses with large total person lateness. Moreover, the IAM strategy provides the TSP with flexible cycle length, as the phase duration is determined by the agent without fixed cycle length constraints.
- The eD3QNI algorithm enables addressing complex multiple priority requests problem. In existing studies of DRL for TSP, it is assumed that priority requests from only one route (including up and down directions) can occur at one time. However, our proposed method represents proper information of multiple buses from conflicting routes in an appropriate form of states, and its agent learns to choose the action and prioritizes the approaching buses in multiple routes by modifying the signal timing.
- The eD3QNI algorithm is evaluated in scenarios with various CVs' penetration rates and different reward functions. This method can be applied to realize different operational goals by modifying the reward structure.

The second work (in Chapter 4) about the CTSPV algorithm at arterials has the following four main contributions.

- First, we develop a MARL framework to realize a cooperative TSP strategy with variable phases. It is a totally data-driven approach and flexibly changes traffic signals (including phase sequence and duration) in accordance with real-time traffic conditions at each time step, differing from other existing TSP methods that change signal timings by cycles or make adjustments to one base signal.
- Second, the proposed method prioritizes buses and improves the efficiency of both transits and non-transits by formulating the reward function as the combination of observable metrics, i.e., queue length and bus lateness. It also solves multiple priority requests well by providing agents with the state of urgent buses from different directions. Moreover, it achieves multi-intersection cooperation by deploying the mixing network of the QMIX framework.
- Third, the proposed method with invalid action masking approach can satisfy the constraints of minimum/maximum green time and skipping rule and solve the

asynchronous decision-making of different agents by setting all agents choosing actions at every step. In addition, we test several minimum/maximum green time constraints to analyze their effects on the proposed approach and find the proper constraints for the RL method to learn efficiently.

- Last but not least, this is the first paper that analyzes the signal timing outcomes of the RL method to explain why RL performs better than conventional methods. We have analyzed the signal timing pattern of the proposed method in terms of cycle length, phase durations, phase-skipping pattern, and action-choosing pattern. We also generalize some rule-based strategies by those patterns and evaluate their performance.

Contributions of the third work (in Chapter 5) about the CTSPVH algorithm at arterials can be concluded to the following four contributions.

- The relationship between headway deviation and bus delay was discussed, on the basis of which we propose a calculation approach for forward and backward headway deviations in real-time.
- The CTSPVH algorithm, like the CTSPV algorithm in Chapter 4, considers four crucial components, i.e., complex state representation with multiple conflicting buses, rational actions constrained by domain knowledge, comprehensive rewards balancing transits and non-transits, and a centralized training scheme for cooperation among agents.
- Based on the above considerations, this work designs specific state representation, action space, and reward functions for arterial traffic control, which can improve transit headway adherence with less cost on non-transit vehicles. In addition, how to set weights of reward functions for the CTSPVH algorithm's optimal performance has also been discussed for practical applications.
- Compared with the second work in Chapter 4, we also consider phase transition loss by setting 3s yellow time and 1s all-red time between phase transitions. The results proved the superiority of RL-based signal control over other traditional methods in situations with phase transitions.

6.3 Future work

The following directions can be further explored in the future.

- Firstly, the low simulation speed of SUMO affects the algorithm training speed. We can integrate one analytical model in the RL algorithm, which can predict state transition and approximate reward fast. It helps to reduce the necessity of numerous environmental interactions by SUMO, and can plan and speed up the training.
- Secondly, the information of connected cars can enhance the estimates of state variables conventionally from fixed-point detectors. In the future, we can measure state variables by information all from CVs and evaluate the CV penetration rates' influence of both cars and buses on the efficiency of the proposed algorithm.
- More systematic sensitivity analysis on the internal parameters for RL methods can be conducted to provide guidance of parameter settings.
- Another possible future direction is to specify the objective agents for communications and design direct communications between agents. It does not make much sense for one agent to share information with distant agents, so there is a need to specify which agents

need mutual communication. In this way, we can decompose large networks into many small communication groups where agents share information mutually. In addition, the increase in the number of intersections would enlarge the neural networks and slow the training process for the proposed method, so we need first to determine which agents are necessary to communicate with which and then think of more efficient communication ways, such as directly sharing some local information with neighbor agents or sharing learned parameters.

- Last but not least, combinations with BH or transit speed advisory can improve TSP strategies' performance. The implementation of TSP strategies has limited adjustment for headway compared to BH approaches; therefore, collaboration with BH can better adjust small headways. In addition, the TSP strategy would be more effective if it could serve the priority requests of multiple buses at the same time. Transit speed advisory is able to advise bus drivers to modify speed and pass intersections with other buses, which gained signal priority. Similarly, speed advisory can further adjust transit headway by suggesting deceleration for small headway and acceleration for large headway. However, those combinations increase the complexity of research problems, as the mutual effects of bus holding time, real-time transit speed, and traffic signals must be considered. Whether using traditional model-based methods or RL algorithms, we need to figure out an appropriate problem formulation. In this case, it would be better to select an entire bus line for the case study rather than a few arterial intersections along the bus route for a thorough performance evaluation.

Appendix

A Framework of comparing methods in Chapter 3

A-I FT method

Given phase p with flow volume per lane of movement m $v_m(m \in J_p)$ where J_p is the set of movements for phase p.

1) Calculate the critical flow rations f_p

Calculate the saturation flow rate per lane of movement j: $s_j = 1/free\ flow\ headway$

$$f_p = \max(v_m/s_m), m \in J_p$$

$$F = \sum_p f_p$$

2) Calculate the cycle length C

$$C = \frac{1.5L + 5}{1 - F}$$

where L is the loss time in the cycle, and it simply equals the sum of all red time and yellow time. In this research, $L = 12$ s with no all-red time and 3s yellow time for 4 phases. We should round up the value of C. If C is larger than 120s, it will be set as 120s as a large cycle will bring the waiting anxiety of drivers.

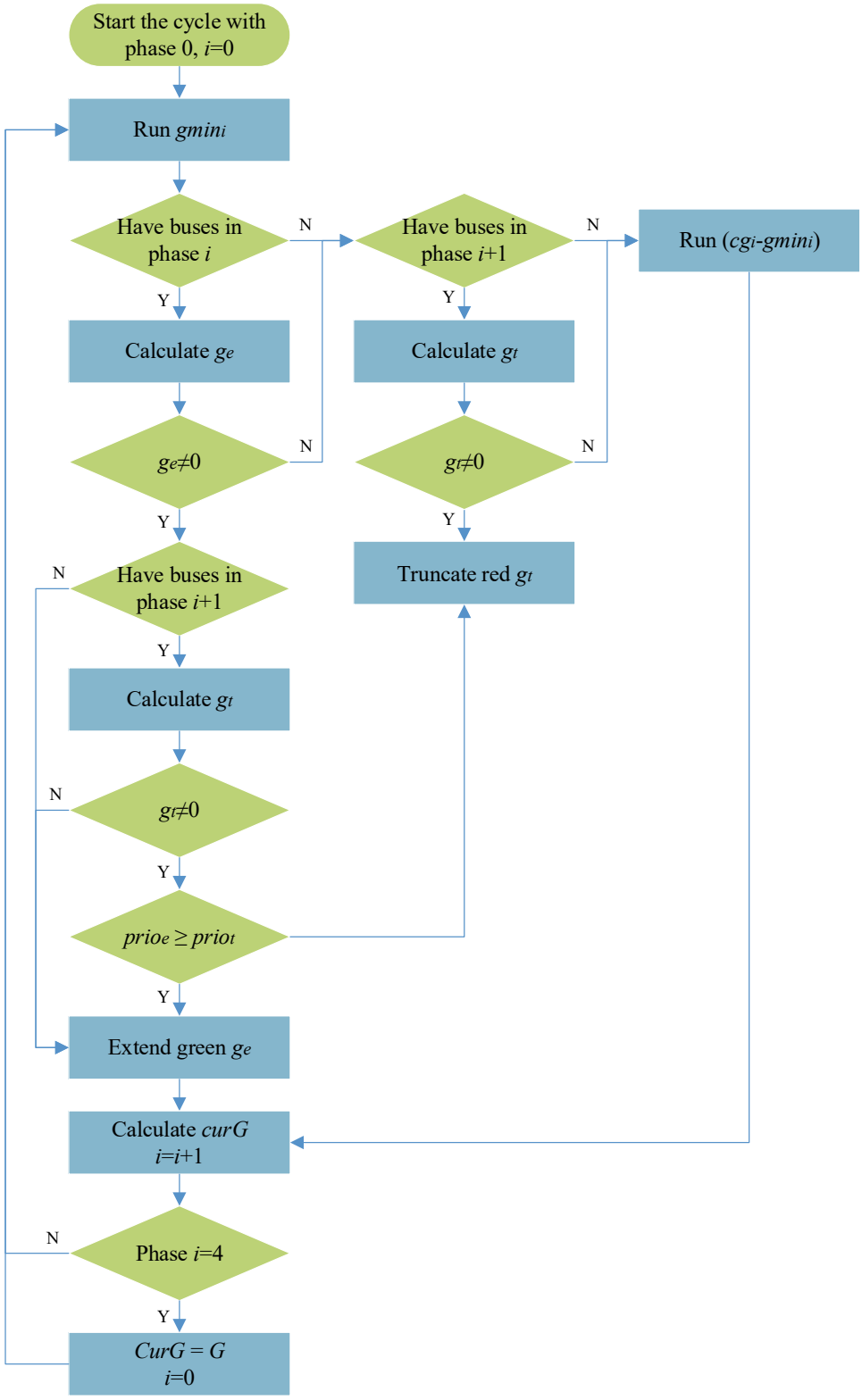
3) Calculate the green time g_p

$$g_p = \frac{f_p}{F}(C - L)$$

Note: We should round up or round down the value g_p and maintain the value of C.

A-II ATSP method

(I) ATSPF method



However, the initial queue should be assumed to be 0 for the ATSP method, and thus there is another strict minimum green time $\widehat{Gmin}$, given by max $(D_{Cross}/V_{Ped}, C * v/s)$, where C is the cycle length, equalling 60 s and 120 s for two demands individually; v and s is the flow volume and saturation

flow rate. we can get the $\widehat{Gmin}$ of P_0, P_1, P_2, P_3 for two demands are 9 s, 9 s, 11 s, 10 s and 19 s, 20 s, 29 s, 29 s, respectively.

Given $G = \{g_i\}, G_{min} = \{gmin_i\}, G_{max} = \{gmax_i\}, CurG = \{cg_i\}, i \in P$.

- **Green extension time g_e**

For bus k in S_{bus} of phase i:

$emax_i = \min\left(\sum_{j>p}(cg_j - gmin_j), 10\right)$

If $cg_i - gmin_i < ETA_k < cg_i - gmin_i + emax_i$:

$ge_k = ETA_k - (cg_i - gmin_i)$

Else:

$ge_k = 0$

$g_e = \max(ge_k)$

- **Red truncation time g_t**

For bus k in S_{bus} of phase i:

If $ETA_k < cg_i - gmin_i$:

$gt_k = cg_i - gmin_i - \max(0, ETA_k - 3)$

Else:

$gt_k = 0$

$g_t = \max(gt_k)$

- **Current green time of all phases $CurG = \{cg_i\}$**

At the beginning of the cycle, $CurG = G$

Then, when GE is implemented,

$$cg_i = \begin{cases} cg_i + g_e & , i \text{ is current phase } p \\ cg_i - g_e\,(cg_i - gmin_i)/\sum_{j>p}(cg_j - gmin_j), & i \text{ is another phase except for } p \end{cases}$$

Then, when RT is implemented,

$$cg_i = \begin{cases} cg_i - g_t & , i \text{ is current phase } p \\ cg_i + g_t\,gmin_i/\sum_{j>p} gmin_j, & i \text{ is another phase except for } p \end{cases}$$

- **Priority for GE and RT $prio_e, prio_t$**

$prio = \sum_k SD_k * O_k/(ETA_k + \varepsilon)$ where $\varepsilon = 10^{-5}$. $prio_e$ and $prio_t$ are the $prio$ term for GE and RT buses.

(2) ATSPV method

Pseudocode of ATSPV method of choosing action

Obtain valid action space A_t by IAM method to consider constraints of green time and phase skipping rule
Calculate $prio_i = \sum_k SD_k * O_k/(ETA_k + \varepsilon)$ where $\varepsilon = 10^{-5}$, k is the index of three buses with the largest $prio$ of phase i, $i \in A_t$. $PRIO = \{prio_i\}$, $i \in A_t$.
if len(argmax $PRIO$) =1:
    choose $action = \underset{i}{\text{argmax}}\, PRIO$
elif current_phase in $\underset{i}{\text{argmax}}\, PRIO$:
    choose *action* = current_phase
else choose *action* = the phase in $\underset{i}{\text{argmax}}\, PRIO$ which is nearest to current_phase in the signal cycle (This signal cycle means from the current phase to the next 3 phases)
end if

A-III REINFORCE algorithm

Algorithm REINFORCE

Input: Discount factor γ, learning rate α, the number of episodes *N*, simulation time *T*, simulation warm-up time *Twarm*

Initialize parameter θ of policy $\pi_\theta(a|s)$

for *episode* = 1 **to** N_e **do**
    Initialize environment
    for *t* = 1 **to** *T* **do**
        if *t* < *Twarm* **then**
            run simulation one step, $count$ = 0, λ = 0
            return
        end if
        obtain state s_1 and action a_1, λ += 1
        Choose action $a_\lambda \backsim \pi_\theta(a_\lambda|s_\lambda)$
        Simulate one step, calculate reward R_k by $r_\lambda = \sum_{t=1}^{\Delta t_\lambda} \gamma^{t-1} r_t$ and get next state $s_{\lambda+1}$
        Store tuple $(s_\lambda, a_\lambda, r_\lambda, \Delta t_\lambda)$
        t = *t* + Δt_λ
    end for
    $\mathbb{N} = \lambda$
    for λ = 1 **to** $\mathbb{N}$ **do**
        Calculate discounted return $G_\lambda = \sum_{\lambda'=\lambda}^{T} \gamma^{\lambda'-\lambda} r_{\lambda'}$
        Accumulate weight changes $\theta \leftarrow \theta + \alpha\gamma^\lambda G_\lambda \nabla_\theta \ln \pi_\theta(a_\lambda|s_\lambda)$
    end for
    Update weights θ
end for

A-IV A2C algorithm

Algorithm A2C

Input: Discount factor γ, learning rate α, the number of episodes N_e, simulation time T, simulation warm-up time *Twarm*

Initialize parameter θ, ω of policy $\pi_\theta(a|s)$ and $V_\omega(s)$

for *episode* = 1 **to** N_e **do**
  Initialize environment
  for t = 1 **to** T **do**
    if t < *Twarm* **then**
      run simulation one step, $count = 0$, $\lambda = 0$
      return
    end if
    obtain state s_1 and action a_1, λ += 1
    Choose action $a_\lambda \backsim \pi_\theta(a_\lambda|s_\lambda)$
    Simulate one step, calculate reward R_k by $r_\lambda = \sum_{t=1}^{\Delta t_\lambda} \gamma^{t-1} r_t$ and get next state $s_{\lambda+1}$
    Calculate TD-error $\delta_\lambda = \hat{V} - V(s_\lambda)$ by $\hat{V} = \begin{cases} r_\lambda & , \sum_t \Delta t_\lambda = T \\ r_\lambda + \gamma^{\Delta t_\lambda} \hat{V}(s_{\lambda+1}; \omega), otherwise \end{cases}$
    Update weight $\omega \leftarrow \omega + \alpha \delta_\lambda \nabla_\omega V_\omega(s_\lambda)$
    Update weight $\theta \leftarrow \theta + \alpha \gamma^\lambda \delta_k \nabla_\theta \ln \pi_\theta(a_\lambda|s_\lambda)$
    $t = t + \Delta t_\lambda$
  end for
end for

A-V DDPG algorithm

Algorithm DDPG

Input: Discount factor γ, actor learning rate α, critic learning rate β, target network update rate τ, replay memory size S_{mem}, batch size S_{bat}, the number of episodes N, simulation time T, simulation warm-up time $Twarm$

Initialize parameter θ, ω of actor $\pi_\theta(a|s)$ and critic $V_\omega(s)$

Initialize parameter $\theta^- = \theta, \omega^- = \omega$ of target actor $\pi_{\theta^-}(a|s)$ and target critic $V_{\omega^-}(s)$

for *episode* = 1 **to** N_e **do**
  Initialize environment
  for t = 1 **to** T **do**
    if t < $Twarm$ **then**
      run simulation one step, $count$ = 0, λ = 0
      return
    end if
    obtain state s_1 and action a_1, $count$ += 1, λ += 1
    Choose action $a_\lambda \backsim \pi_\theta(a_\lambda|s_\lambda)$
    Simulate one step, calculate reward r_λ by $r_\lambda = \sum_{t=1}^{\Delta t_\lambda} \gamma^{t-1} r_t$ and get next state s_λ
    if replay memory reaches the size S_{rep} **then**
      Delete the oldest memory tuples
    end if
    Store tuple $(s_\lambda, a_\lambda, r_\lambda, s_{\lambda+1}, \Delta t_\lambda)$ to replay memory
    $t = t + \Delta t_\lambda$
    if the number of stored memories > S_{bat} **then**
      for i=1 **to** S_{bat} **do**
        Calculate critic TD-error $\delta_i = r_i + \gamma^{\Delta t_i} V_{\omega^-}(s_{i+1}) - V_\omega(s_i)$
        Accumulate weight changes $\omega \leftarrow \omega + \beta \delta_i \nabla_\omega V_\omega(s_i)$
        Accumulate weight changes $\theta \leftarrow \theta + \alpha \delta_i \nabla_\theta \ln \pi_\theta(a_i|s_i)$
      end for
      Update actor and critic networks' weights ω, θ
      Update target critic network $\omega^- \leftarrow \tau\omega + (1-\tau)\omega$
      Update target actor network $\theta^- \leftarrow \tau\theta + (1-\tau)\theta$
    end if
  end for
end for

B Framework of FRBS method in Chapter 4

Pseudocode: FRBS method

Input: traffic simulation time T, simulation warm-up time T_w, number of agents N, the number of episodes N_e, full action space Å $= \{P_0, P_1, P_2, P_3\}$

for $episode = 1$ **to** N_e **do**
  for $t = 1$ **to** T **do**
    Run one simulation step
    if $t \geq T_w$ **then**
      for agent $i = 1$ **to** N **do**
        Obtain current phase φ_t^i and valid action space A_t^i
        if len(A_t^i) == 1 **then**
          Choose the only valid action $a_t^i = A_t^i$
        else
          // F-value method (related to phase duration)
          if yellow in A_t^i **then**
            Calculate F-value F_p (Eq (4.10)) of all phases $p \in$ Å
            Choose the action $a_t^i = \begin{cases} \varphi_t^i, & \text{if } F_{\varphi_t^i} = \max\limits_{p \in Å} F_p \\ \text{yellow}, & \text{otherwise} \end{cases}$
          // Action-choosing pattern (related to phase sequence)
          else
            if φ_t^i==0 **then**
              Choose the action $a_t^i = \begin{cases} P_2, & \text{if } Queue_{P_2} > 2.5 \\ P_1, & \text{otherwise} \end{cases}$
            else
              Choose the action $a_t^i = \begin{cases} P_3, & \text{if } P_0 \text{ not in } A_t^i \\ P_0, & \text{otherwise} \end{cases}$
            end if
          end if
        end if
      end for
      Execute action $\Lambda_t = \{a_t^i\}$ for all $i \in \{1, \ldots, N\}$
    end if
  end for
end for

C Framework of BH method in Chapter 5

Referred to the work of Daganzo (2009), holding time is given by

$$D_{n,s} = d_s + (\alpha + \beta_s)(SH - h_{n,s})$$

- α is a term related to sensitivity to control, $\alpha \in (0,1)$. Here we set it to be 0.5.
- β_s is a dimensionless parameter that expresses the marginal increase in bus delays from a one-unit increase in bus headway. For most bus routes, the increased bus delay depends mainly on boarding as alighting is quick. Intuitively, a one-unit increase in bus headway will increase λ of people arriving at the bus stop and increase $(BT * \lambda)$ boarding time, which is the primary source of bus delay. Therefore, here we set $\beta_s = BT * \lambda$.
- d_s is the average bus delay at equilibrium at stop s, satisfying $d_s \geq 3(\alpha + \beta_s)\sigma_{ns}$ where σ_{ns} is the headway variance of bus n at stop s. Here we finally use $d_s = 3(\alpha + \beta_s)\sigma_{ns}$. Because the headway deviation we set follows a normal distribution with an expectation of 0, $\sigma_{ns} = 0$ and then $d_s = 0$.
- SH is scheduled headway.
- $h_{n,s}$ is the actual headway of bus n at stop s, which can be easily obtained by the difference in arrival time between bus n and the last bus $n - 1$.

Pseudocode of BH method

Given $\alpha = 0.5$, $\sigma_{ns} = 0$, person arrival rate $\lambda = \begin{cases} 0.0083, & \text{for S1 and S5} \\ 0.0167, & \text{for other periods} \end{cases}$, boarding time per person $BT = 2$, bus scheduled headway $SH = \begin{cases} 10, & \text{for S1 and S5} \\ 5, & \text{for other periods} \end{cases}$.

if bus n arrives bus stop s:

record arrival time $t_{n,s}$

calculate $h_{n,s} = t_{n,s} - t_{n-1,s}$

calculate $D_{n,s} = d_s + (\alpha + \beta_s)(SH - h_{n,s})$

hold bus for $D_{n,s}$ by syntax setStop in SUMO

Reference

Abdelhalim, A., Abbas, M., 2021. A Value Proposition of Cooperative Bus-Holding Transit Signal Priority Strategy in Connected and Automated Vehicles Environment. *IEEE Transactions on Intelligent Transportation Systems*, 1-6.

Ahmed, M.A.A., Khoo, H.L., Ng, O.-E., 2023. Discharge control policy based on density and speed for deep Q-learning adaptive traffic signal. *Transportmetrica B: Transport Dynamics* 11(1), 1707-1726.

Alegre, L.N., Ziemke, T., Bazzan, A.L.C., 2022. Using Reinforcement Learning to Control Traffic Signals in a Real-World Scenario: An Approach Based on Linear Function Approximation. *IEEE Transactions on Intelligent Transportation Systems* 23(7), 9126-9135.

Alizadeh Shabestray, S.M., Abdulhai, B., 2019. Multimodal iNtelligent Deep (MiND) Traffic Signal Controller, *IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE.

Anderson, P., Daganzo, C.F., 2020. Effect of transit signal priority on bus service reliability. *Transportation Research Part B: Methodological* 132, 2-14.

Bagherian, M., Mesbah, M., Ferreira, L., 2015. Using delay functions to evaluate transit priority at signals. *Public Transport* 7(1), 61-75.

Bie, Y.M., Liu, Z.Y., Wang, H.Q., 2020. Integrating Bus Priority and Presignal Method at Signalized Intersection: Algorithm Development and Evaluation. *Journal of Transportation Engineering Part A: Systems* 146(6), 11.

Chen, X., Yu, L., Zhang, Y., Guo, J., 2009. Analyzing urban bus service reliability at the stop, route, and network levels. *Transportation Research Part A: Policy and Practice* 43(8), 722-734.

Chow, A.H.F., Li, S., Zhong, R., 2017. Multi-objective optimal control formulations for bus service reliability with traffic signals. *Transportation Research Part B: Methodological* 103, 248-268.

Christofa, E., Ampountolas, K., Skabardonis, A., 2016. Arterial traffic signal optimization: A person-based approach. *Transportation Research Part C: Emerging Technologies* 66, 27-47.

Christofa, E., Papamichail, I., Skabardonis, A., 2013. Person-Based Traffic Responsive Signal Control Optimization. *IEEE Transactions on Intelligent Transportation Systems* 14(3), 1278-1289.

Chu, T., Wang, J., Codeca, L., Li, Z., 2020. Multi-Agent Deep Reinforcement Learning for Large-Scale Traffic Signal Control. *IEEE Transactions on Intelligent Transportation Systems* 21(3), 1086-1095.

Cvijovic, Z., Zlatkovic, M., Stevanovic, A., Song, Y., 2022. Conditional Transit Signal Priority for Connected Transit Vehicles. *Transportation Research Record: Journal of the Transportation Research Board* 2676(2), 490-503.

Daganzo, C.F., 2009. A headway-based approach to eliminate bus bunching: Systematic analysis and comparisons. *Transportation Research Part B: Methodological* 43(10), 913-921.

Daganzo, C.F., Pilachowski, J., 2011. Reducing bunching with bus-to-bus cooperation. *Transportation Research Part B: Methodological* 45(1), 267-277.

El-Tantawy, S., Abdulhai, B., Abdelgawad, H., 2014. Design of Reinforcement Learning Parameters for Seamless Application of Adaptive Traffic Signal Control. *Journal of Intelligent Transportation Systems* 18(3), 227-245.

Genders, W., Razavi, S., 2019. Asynchronous n-step Q-learning adaptive traffic signal control. *Journal of Intelligent Transportation Systems* 23(4), 319-331.

Ghaffari, A., Mesbah, M., Khodaii, A., 2020. Designing a transit priority network under variable demand. *Transportation Letters* 12(6), 427-440.

Gronauer, S., Diepold, K., 2021. Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review*.

Guler, S.I., Cassidy, M.J., 2012. Strategies for sharing bottleneck capacity among buses and cars. *Transportation Research Part B: Methodological* 46(10), 1334-1345.

Guo, G., Wang, Y.P., 2021. An integrated MPC and deep reinforcement learning approach to trams-priority active signal control. *Control Eng Pract* 110.

Gupta, J.K., Egorov, M., Kochenderfer, M., 2017. Cooperative Multi-agent Control Using Deep Reinforcement Learning. Springer International Publishing, pp. 66-83.

Haitao, H., Menendez, M., Ilgin Guler, S., 2018. Analytical evaluation of flexible-sharing strategies on multimodal arterials. *Transportation Research Part A: Policy and Practice* 114, 364-379.

Han, T., Ito, M., Shirahata, K., Oguchi, T., 2021. A Study on Possibility of Predictive Deep Reinforcement Learners for Isolated Intersection Signal Control. *SEISAN KENKYU* 73(2), 107-112.

Head, K., Gettman, D., Wei, Z., 2006. Decision Model for Priority Control of Traffic Signals. *Transportation Research Record: Journal of the Transportation Research Board* 1978(1), 169-177.

Hu, J., Park, B.B., Lee, Y.J., 2016. Transit signal priority accommodating conflicting requests under Connected Vehicles technology. *Transportation Research Part C: Emerging Technologies* 69, 173-192.

Huang, S., Ontañón, S., 2020. A Closer Look at Invalid Action Masking in Policy Gradient Algorithms. *arXiv pre-print server*.

Kai, Poddar, S., Sharma, A., Sarkar, S., 2019. Deep Reinforcement Learning for Adaptive Traffic Signal Control. *arXiv pre-print server*.

Khan, Z.S., He, W., Menéndez, M., 2023. Application of modular vehicle technology to mitigate bus bunching. *Transportation Research Part C: Emerging Technologies* 146.

Krajzewicz, D., Hertkorn, G., Rössel, C., Wagner, P., 2002. SUMO (Simulation of Urban MObility) - an open-source traffic simulation, *4th Middle East Symposium on Simulation and Modelling*, Sharjah (United Arab Emirates), pp. 183-187.

La, P., Bhatnagar, S., 2011. Reinforcement Learning With Function Approximation for Traffic Signal Control. *IEEE Transactions on Intelligent Transportation Systems* 12(2), 412-421.

Lee, W.H., Wang, H.C., 2022. A Person-Based Adaptive Traffic Signal Control Method with Cooperative Transit Signal Priority. *Journal of Advanced Transportation* 2022, 1-17.

Li, J., Liu, Y., Zheng, N., Tang, L., Yi, H., 2022. Regional Coordinated Bus Priority Signal Control Considering Pedestrian and Vehicle Delays at Urban Intersections. *IEEE Transactions on Intelligent Transportation Systems* 23(9), 16690-16700.

Liang, S., Leng, R., Zhang, H., Zhao, J., 2022. Two-Stage Transit Signal Priority Control Method to Improve Reliability of Bus Operation Considering Stochastic Process. *Transportation Research Record: Journal of the Transportation Research Board* 2677(2), 1713-1727.

Liang, X., Du, X., Wang, G., Han, Z., 2019. A Deep Reinforcement Learning Network for Traffic Light Cycle Control. *IEEE Transactions on Vehicular Technology* 68(2), 1243-1253.

Ling, K., Shalaby, A., 2003. Automated transit headway control via adaptive signal priority. *Journal of Advanced Transportation* 38(1), 45-67.

Liu, H., Teng, K., Rai, L., Xing, J., 2021. Transit Signal Priority Controlling Method Considering Non-Transit Traffic Benefits and Coordinated Phase States for Multi-Rings Timing Plan at Isolated Intersections. *IEEE Transactions on Intelligent Transportation Systems* 22(2), 913-936.

Long, K., Wei, J., Gu, J., Yang, X., 2020. Headway-Based Multi-Route Transit Signal Priority at Isolated Intersection. *IEEE Access* 8, 187824-187831.

Ma, J., Wu, F., 2022. Feudal Multi-Agent Reinforcement Learning with Adaptive Network Partition for Traffic Signal Control. *arXiv*.

Ma, W., Head, K.L., Feng, Y., 2014. Integrated optimization of transit priority operation at isolated intersections: A person-capacity-based approach. *Transportation Research Part C: Emerging Technologies* 40, 49-62.

Ma, W., Liu, Y., Yang, X., 2013. A Dynamic Programming Approach for Optimal Signal Priority Control Upon Multiple High-Frequency Bus Requests. *Journal of Intelligent Transportation Systems* 17(4), 282-293.

Mao, F., Li, Z., Li, L., 2022. A Comparison of Deep Reinforcement Learning Models for Isolated Traffic Signal Control. *IEEE Intelligent Transportation Systems Magazine*, 2-22.

Mirchandani, P., Knyazyan, A., Head, L., Wu, W., 2001. An Approach Towards the Integration of Bus Priority, Traffic Adaptive Signal Control, and Bus Information/Scheduling Systems. *Lecture Notes in Economics and Mathematical Systems*, 319-334.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M., 2013. Playing Atari with Deep Reinforcement Learning. *arXiv pre-print server*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D., 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540), 529-533.

Rasheed, F., Yau, K.L.A., Low, Y.C., 2020. Deep reinforcement learning for traffic signal control under disturbances: A case study on Sunway city, Malaysia. *Future Generation Computer Systems* 109, 431-445.

Rashid, T., Samvelyan, M., Witt, C.S.d., Farquhar, G., Foerster, J., Whiteson, S., 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. *arXiv pre-print server*.

Rummery, G.A., Niranjan, M., 1994. On-line Q-Learning using connectionist systems.

Schaul, T., Quan, J., Antonoglou, I., Silver, D., 2016. Prioritized Experience Replay. *arXiv pre-print server*.

Shalaby, A., Hu, W.X., Corby, M., Wong, A., Zhou, D., 2021. Transit signal priority: research and practice review and future needs. Edward Elgar Publishing, pp. 340-372.

Sheffield, M.H., Schultz, G.G., Bassett, D., Eggett, D.L., 2021. Sensitivity Analysis of the Transit Signal Priority Requesting Threshold and the Impact on Bus Performance and General Traffic. *Transportation Research Record: Journal of the Transportation Research Board* 2675(5), 036119812098585.

Shi, J., Sun, Y., Schonfeld, P., Qi, J., 2017. Joint optimization of tram timetables and signal timing adjustments at intersections. *Transportation Research Part C: Emerging Technologies* 83, 104-119.

Smith, H.R., Hemily, B., Ivanovic, M., 2005. Transit Signal Priority (TSP): A Planning and Implementation Handbook.

Stevanovic, J., Stevanovic, A., Martin, P.T., Bauer, T., 2008. Stochastic optimization of traffic control and transit priority settings in VISSIM. *Transportation Research Part C: Emerging Technologies* 16(3), 332-349.

Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W.M., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J.Z., Tuyls, K., Graepel, T., 2018. Value-Decomposition Networks For Cooperative Multi-AgentLearning Based On Team Reward, *Proceedings of the 17th international conference on autonomous agents and multiagent systems (AAMAS 2018).* Stockholm, Sweden, pp. 2085-2087.

Sutton, R.S., Barto, A.G., 2018. *Reinforcement learning: An introduction*. MIT press.

Sutton, R.S., McAllester, D., Singh, S., Mansour, Y., 1999. Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems* 12, 1057–1063.

Tan, M., 1993. Multi-agent reinforcement learning: Independent vs. cooperative agent, *Proceedings of the tenth international conference on machine learning*, pp. 330-337.

Tan, T., Bao, F., Deng, Y., Jin, A., Dai, Q., Wang, J., 2020. Cooperative Deep Reinforcement Learning for Large-Scale Traffic Grid Signal Control. *IEEE Transactions on Cybernetics* 50(6), 2687-2700.

Thodi, B.T., Chilukuri, B.R., Vanajakshi, L., 2021. An analytical approach to real-time bus signal priority system for isolated intersections. *Journal of Intelligent Transportation Systems*, 1-23.

Timothy, Jonathan, Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D., 2019. Continuous control with deep reinforcement learning. *arXiv pre-print server*.

Truong, L.T., Currie, G., Sarvi, M., 2017. Analytical and simulation approaches to understand combined effects of transit signal priority and road-space priority measures. *Transportation Research Part C: Emerging Technologies* 74, 275-294.

Turnquist, M.A., 1981. Strategies for Improving Reliability of Bus Transit Service. *Transportation Research Record*(818).

Van Oort, N., Wilson, N.H.M., Van Nes, R., 2010. Reliability Improvement in Short Headway Transit Services. *Transportation Research Record: Journal of the Transportation Research Board* 2143(1), 67-76.

Wan, C.H., Hwang, M.C., 2018. Value-based deep reinforcement learning for adaptive isolated intersection signal control. *IET Intelligent Transport Systems* 12(9), 1005-1010.

Wang, J., Sun, L., 2020. Dynamic holding control to avoid bus bunching: A multi-agent deep reinforcement learning framework. *Transportation Research Part C: Emerging Technologies* 116.

Wang, T., Cao, J., Hussain, A., 2021. Adaptive Traffic Signal Control for large-scale scenario with Cooperative Group-based Multi-agent reinforcement learning. *Transportation Research Part C: Emerging Technologies* 125.

Wang, Z., Schaul, T., Hessel, M., Hado, Lanctot, M., Nando, 2016. Dueling Network Architectures for Deep Reinforcement Learning. *arXiv pre-print server*.

Williams, R.J., 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8(3-4), 229-256.

Xiao, Y., Ali, M.S., Kaisar, E.I., Hadi, M., 2022. Development of Planning-Level Guidelines for Deploying Transit Signal Priority. *IEEE Transactions on Intelligent Transportation Systems* 23(9), 14731-14740.

Xu, M., Chai, J., Yan, Y., Qu, X., 2022a. Multi-Agent Fuzzy-Based Transit Signal Priority Control for Traffic Network Considering Conflicting Priority Requests. *IEEE Transactions on Intelligent Transportation Systems* 23(2), 1554-1564.

Xu, T., Barman, S., Levin, M.W., Chen, R., Li, T., 2022b. Integrating public transit signal priority into max-pressure signal control: Methodology and simulation study on a downtown network. *Transportation Research Part C: Emerging Technologies* 138, 103614.

Yang, K., Menendez, M., Guler, S.I., 2019. Implementing transit signal priority in a connected vehicle environment with and without bus stops. *Transportmetrica B: Transport Dynamics* 7(1), 423-445.

Yu, Z., Gayah, V.V., Christofa, E., 2017. Person-Based Optimization of Signal Timing. *Transportation Research Record: Journal of the Transportation Research Board* 2620(1), 31-42.

Yu, Z., Gayah, V.V., Christofa, E., 2018. Implementing phase rotation in a person-based signal timing optimization framework, *21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE.

Yu, Z., Xu, G., Gayah, V.V., Christofa, E., 2022. Incorporating Phase Rotation Into a Person-Based Signal Timing Optimization Algorithm. *IEEE Transactions on Intelligent Transportation Systems* 23(1), 513-521.

Zeng, X., Zhang, Y., Jiao, J., Yin, K., 2021. Route-Based Transit Signal Priority Using Connected Vehicle Technology to Promote Bus Schedule Adherence. *IEEE Transactions on Intelligent Transportation Systems* 22(2), 1174-1184.

Zhang, H., Liang, S., Han, Y., Ma, M., Leng, R., 2020. A Prediction Model for Bus Arrival Time at Bus Stop Considering Signal Control and Surrounding Traffic Flow. *IEEE Access* 8, 127672-127681.

Zhang, Y., Zhou, Y., Lu, H., Fujita, H., 2021. Cooperative multi-agent actor–critic control of traffic network flow based on edge computing. *Future Generation Computer Systems* 123, 128-141.

Zhou, C., Tian, Q., Wang, D.Z.W., 2022. A novel control strategy in mitigating bus bunching: Utilizing real-time information. *Transport Policy* 123, 1-13.

Zhou, Y., Ozbay, K., Kachroo, P., Zuo, F., 2020. Ramp Metering for a Distant Downstream Bottleneck Using Reinforcement Learning with Value Function Approximation. *Journal of Advanced Transportation* 2020, 1-13.

Zlatkovic, M., Stevanovic, A., Martin, P.T., 2012. Development and Evaluation of Algorithm for Resolution of Conflicting Transit Signal Priority Requests. *Transportation Research Record: Journal of the Transportation Research Board* 2311(1), 167-175.