Transit Signal Priority Control Method Based on Deep Reinforcement Learning

Junnan Chen[1], Xufei Zhuang[2, *], Heng Li[3], Chenxi Yang[4]

{20211800083@imut.edu.cn[1], zxf@imut.edu.cn[2], henrylee.m@foxmail.com[3], 1260917842@qq.com[4]}

Inner Mongolia University of Technology, Hohhot, China

Abstract. Transit signal priority control plays a pivotal role in enhancing the efficiency of public transportation and mitigating traffic congestion. Despite advancements in current research on transit signal priority control using deep reinforcement learning, the field is still nascent, encountering challenges like limited model generalization and slow training convergence, particularly in intricate traffic environments. This paper proposes a transit signal priority control model based on deep reinforcement learning. The model utilizes a deep neural network framework, integrating crucial optimizations like Dueling DQN, Distributional DQN and the PER mechanism, alongside enhancements to the loss function, to accommodate diverse traffic scenarios. The model incorporates essential information at intersections, encompassing vehicle positions, velocities, and lane conditions as input states to comprehensively depict the traffic situation. Concurrently, by delineating a set of signal phase configurations at intersections, a reward function is devised that considers various factors, such as intersection passage efficiency, bus delays, and passenger experience. This guides the agent to optimize the entire system throughout the learning process, duly considering signal control strategies to ensure effective decision-making. The proposed method's feasibility and effectiveness are validated through simulation experiments on the SUMO traffic simulation platform and comparison with three typical deep reinforcement learning algorithms. The experimental results indicate that the enhanced deep reinforcement learning network model not only accelerates the model's convergence speed but also dynamically adjusts signal timing based on real-time traffic conditions, effectively enhancing bus passage efficiency and mitigating congestion. It demonstrates superior adaptability, robustness, and scalability, offering strong support for intelligent traffic signal control systems.

Keywords: Intelligent transportation; transit signal priority control; deep reinforcement learning; SUMO simulation

1 Introduction

Urban traffic congestion has consistently posed a challenging concern for both authorities and scholars. Promoting public transport stands as an effective measure to alleviate traffic congestion in global metropolitan areas, given its higher capacity and recognized efficiency in large cities. Transit signal priority (TSP) control not only minimizes the dwell time at intersections and shortens passenger travel time, enhancing the appeal of public transport, but also promotes increased public transit usage, leading to a reduction in private car travel and, consequently, alleviating traffic congestion. Traditional TSP control methods exhibit limitations, including passive and active priority. Conventional timing or inductive control methods

PMIS 2024, March 15-17, Changsha, People's Republic of China

DOI 10.4108/eai.15-3-2024.2346425

frequently struggle to adapt to intricate and dynamic traffic conditions, making real-time signal length adjustments unfeasible to meet actual demand. Consequently, during peak hours or specific circumstances, buses may encounter stoppages and delays, impacting overall operational efficiency.

In recent years, Artificial Intelligence (AI) has played a significant role in intelligent transport, encompassing various applications, including traffic flow prediction, intelligent signal control, traffic monitoring, autonomous driving, travel recommendation, path planning, and bus priority control. While Reinforcement Learning (RL) and Deep Reinforcement Learning (DRL) have shown notable advancements in traffic signal control [1-3], their application in bus priority control is still in the early exploration stage. In contrast to traditional methods, RL and DRL methods avoid the need for detailed analysis and optimization schemes for various traffic situations; instead, they establish an interaction framework between the environment and the Agent. The Agent selects and performs actions based on state information from buses, other vehicles, and signal timings in the environment. This involves sensing information, modifying traffic signals, and obtaining the Agent's next state and immediate reward. Thus, RL and DRL have the advantage of robust adaptability to dynamic traffic conditions without requiring a priori assumptions about the optimization model. Present RL and DRL models primarily employ actions like green extension and red truncation [4-6]. However, this approach often results in the current timing scheme being one of the best among a limited set of schemes and not necessarily the most adaptive to the current intersection. Moreover, control strategies like green extension and red truncation may lead the model to overly prioritize reducing bus delays, potentially causing delays and congestion in other non-priority phases.

Based on deep reinforcement learning, the TSP control method developed in this paper transcends the conventional paradigms of signal prolongation or early termination. It introduces a strategy for dynamically adjusting signal duration. This method prioritizes buses and concurrently mitigates delays and congestion in other priority phases. Deep neural networks are employed to model and control traffic signal behavior, ensuring a more flexible response of the signal control system to real-time traffic conditions. The DRL algorithm, incorporating the strengths of distributed DQN, dueling DQN and prioritized experience replay (PER), along with optimizing the loss function, empowers the agent to learn the optimal signal control strategy autonomously. This strategy aims to maximize bus operational efficiency, minimizing stopping time and queuing delays. Lastly, this paper will simulate the intersection traffic situation using the SUMO simulation platform to assess the performance and practicality of the proposed method.

2 Related works

The prevailing TSP control systems predominantly employ the conventional active priority control strategy. As active priority control excessively relies on the precision of the prediction model and necessitates prior bus priority requests, the system waits for the bus to apply, responds, and then decides whether to grant signal priority. This often results in avoidable delays and congestion. Moreover, it struggles to cope with intersections characterized by high traffic volume, complexity, and dynamic changes in active priority control. Real-time priority control employs advanced sensors and communication technologies to acquire real-time traffic

information at intersections, encompassing the position, speed, and direction of buses and other vehicles. Utilizing this information and employing prediction models and intelligent algorithms, the signal control strategy dynamically adapts to the real-time traffic environment. It flexibly adjusts the signal control period, phase length, and sequence to optimize bus and vehicle traffic efficiency, minimizing delays and congestion. Lee [7] and colleagues identified the most suitable TSP control scheme using a bus travel time prediction model. They employed the priority plan redevelopment function to modify or switch the implemented priority plan. This involved a combination of traditional green extension, red truncation, and queue clearing strategies. Zhang [8] and collaborators used a queue prediction model to forecast the queue length at an intersection. They utilized bus vehicle data, queue length, bus stop location, dwell time, and bus vehicles in front of the bus stop to formulate a distance prediction model between buses and stop lines. Employing genetic algorithms, they resolved the signal timing scheme. They implemented a rolling optimization strategy to adjust the signal timing scheme continuously. Sun [9] introduced a two-layer optimization model to enhance the capacity of buses at intersections and decrease the delay in social vehicle operation resulting from the timing optimization scheme. The lower layer model aims to reduce bus delays at intersections, minimizing the average passenger delay. It is solved using the differential evolution algorithm. Hamid [10] applied motion wave theory to depict the queuing process of social vehicles and buses. They optimized the Mixed-Integer Non-Linear Programming (MINLP) problem, focusing on minimizing passenger delays through signal timing. This problem is solved using a genetic algorithm, with green time and phase sequence as two decision variables. Although traditional real-time priority control methods for bus signals have been effective in improving intersection capacity and reducing delays, they still have some limitations. These methods typically rely on prediction models and optimization algorithms to adjust the signal timing scheme, making it challenging to adapt to complex traffic scenarios and real-time changes in traffic flows. Additionally, traditional methods encounter difficulties in addressing non-linearity, non-determinism, and dynamics, thus limiting their capacity to enhance the overall efficiency of the public transport system.

In TSP control, DRL utilizes deep neural networks to model interactions with the environment, presenting novel opportunities for TSP control. Recent research in traffic signal control [11, 12] has employed DRL and 'pressure' control to discern more effective strategies for traffic signal optimization. Positive rewards are bestowed upon the agent when it undertakes actions to ameliorate traffic conditions, alleviate congestion, or enhance traffic efficiency, thereby aiding in pressure alleviation. Conversely, if the agent's actions lead to increased traffic congestion or reduced traffic efficiency, it receives negative rewards to encourage avoidance of unfavorable behaviors. Sun [13], leveraging DRL and integrating the benefits of Double DQN and PER, achieved notable advancements in traffic signal control. Significant progress has been made in traffic signal control in RL and deep reinforcement learning DRL. In bus priority control, there is less research related to these methods. Shang [14] formulated the traffic signal control challenge for intersections with buses as a Markov Decision Process. They considered the repercussions of signal adjustments on adjacent intersections. Their study aims to mitigate passenger delays and instances of stopping. They systematically analyzed and formulated a signal priority strategy for mainline buses through the iterative employment of DRL. Long [5] introduced an improved approach for TSP control termed Enhanced Dueling Double Deep Q Network with Invalid Action Masking (ED3QNI). The algorithm employs invalid action masking, considering traffic signal constraints and skip-phase rules. It incorporates a reward function evaluated by humans and accommodates multiple TSP requests. Simulation results

illustrate that the ED3QNI method proficiently investigates the interplay between the environment and the agent, thereby improving bus operational efficiency compared to conventional methods like fixed-time signals and active priority control strategies. In summary, current research has made significant strides in applying DRL to TSP control. In conclusion, contemporary research has made substantial progress in employing DRL for TSP control. Nevertheless, there is a need for advancement in the following areas: 1) Enhancing algorithmic stability and robustness to handle fluctuations in TSP request numbers and traffic volumes. 2) Investigating the intricate interplay between the traffic environment and the agent, formulating universally applicable and adaptive reward functions for a more precise evaluation of TSP and traffic conditions—a prospective avenue for research. 3) To boost performance, Previous studies have implemented enhancements using DRL, including Dueling DQN, Double DQN, and PER. However, the applicability and efficacy of these improvement methods may vary across different traffic scenarios, warranting additional empirical research to validate their effectiveness and generalizability.

3 Deep Reinforcement Learning

3.1 Deep Reinforcement Learning

DRL is an innovative algorithm integrating Deep Learning with Reinforcement Learning for end-to-end learning from perception to action. Traditional reinforcement learning faces limitations in handling high-dimensional and continuous traffic state information, constraining its further optimization for real-time decision-making. In response, DeepMind [15] introduced the Deep Q-Network (DQN) model, which integrates deep neural networks and Q-learning algorithms to facilitate learning and decision-making in complex environments by mapping the state space to action-value functions (Q-values). The core idea is to use a deep convolutional neural network (CNN) to approximate and learn Q-values between state-action pairs. The network is trained to minimize the Q-values' prediction error to approximate the target Q-values in Q-learning. In addition, DQN introduces the concepts of Experience Replay (ER) and Goal Network to improve the stability and effectiveness of training.ER helps to break the temporal correlation between the data and improve the efficiency and stability of training by preserving the Agent's previous experience and randomly sampling it for training. The target network is then used to stabilize the estimation of the target Q-value, which helps to mitigate the instability problem in training by slowing down the update frequency of the target network. DQN employs a convolutional neural network to approximate the action-value function of the current state to that of the target state, with and denoting the parameters of the current network and the target network, respectively. The approximate optimisation objective representation of the current value function is given in equation (1):

$$y(s,a) = r + \gamma \max_{a'} Q(s', a'; \theta^{-}) \tag{1}$$

The optimal value is obtained by minimizing the Temporal Difference (TD) between the target value and the current value, see equation (2):

$$\delta = r + \gamma \max_{a'} Q(s', a'; \theta^{-}) - Q(s, a, \theta) \tag{2}$$

The loss function of the neural network is given in equation (3):

$$Loss(\theta) = \frac{1}{2}\sum_{i=1}^{N} \delta^2 \tag{3}$$

In many DRL tasks based on visual perception, the corresponding value functions of different state-action pairs are different. However, in some states, the magnitude of the value function is independent of the actions. Based on the above idea, Wang [16] proposed a competitive network structure as a network model for DQN. The network structure is shown in Fig. 1.

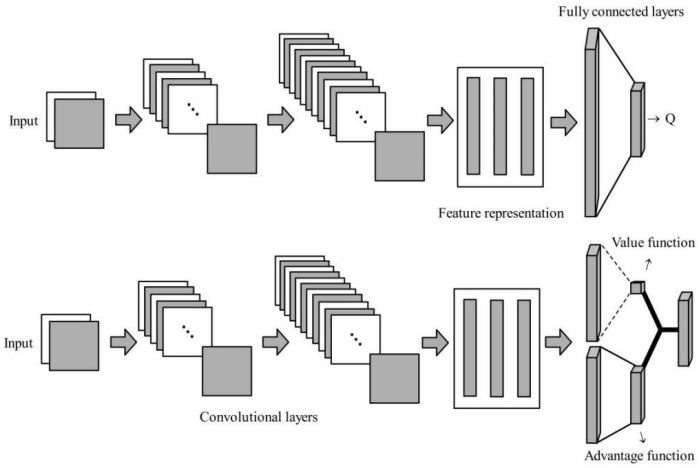


Fig. 1. Dueling DQN network structure

The first model uses the standard DQN structure, which connects the convolutional and fully connected layers through an input layer to output the value of each action. On the other hand, the second model, known as the competitive network, introduces a branching structure based on the abstract features extracted from the convolutional layers. The upper branch signifies the state value function, reflecting the static state value of the environment itself; the lower branch represents the state-dependent action advantage function, signifying the additional value derived from choosing a specific action. Ultimately, these two branches are combined to derive the value associated with each action. This competitive structure enables the model to learn the value of the environment's state independent of action effects.

Schaul [17] introduced a deep reinforcement learning algorithm called Prioritized Experience Replay based on priority replay sampling. The algorithm incorporates a prioritization strategy that assigns sample weights based on the significance of prior experiences. Samples typically deemed advantageous for learning will receive higher priority, being more likely to be selected for training, enhancing the learning of crucial experiences during the training process. This mechanism intensifies the algorithm's emphasis on crucial experiences, expediting the model's learning process. The priority formula is shown in equation (4):

$$P(i) = \frac{p_i^{\alpha}}{\sum_k p_k^{\alpha}} \tag{4}$$

Where α determines the degree of priority to be applied.

3.2 Distributional reinforcement learning

Distributional DQN represents an enhancement of the conventional DQN algorithm, with its

primary innovation involving the incorporation of distributional estimation for the value function associated with state-action pairs. Unlike traditional DQN, which solely estimates the expected value, Distributional DQN offers a more comprehensive depiction of the uncertainty in state values by portraying the value function as a probability distribution. The algorithm employs a collection of atomic primes to approximate the distribution of the value function, expressing diverse potential state values through the learned probabilities associated with each atomic prime. This enhancement aids in addressing variability and uncertainty among distinct state values, enhancing the algorithm's adaptability to intricate tasks. This capability allows Distributional DQN to exhibit enhanced flexibility across reinforcement learning scenarios, leading to superior learning outcomes.

Bellenmare [18] introduced a probability mass distribution relying on discrete supports, see equation (5), where z is a vector comprising atomic elements with $N_{atoms} \in N^+$.

$$z^i = v_{min} + (i-1)\frac{v_{max}-v_{min}}{N_{atoms}-1}, i \in \{1, \cdots, N_{atoms}\} \tag{5}$$

Employ $\mathrm{d_t}$ to represent the estimated distribution within this support at time t. The probability mass associated with each atom i is $\mathrm{p_\theta^i(S_t, A_t)}$, i.e., $\mathrm{d_t = \left(z, p_\theta(S_t, A_t)\right)}$. The learning process aims to achieve convergence between the estimated distribution of cumulative returns and the actual distribution through updates to parameter θ. In distributed reinforcement learning, the distribution of returns adheres to a modified version of Bellman's equation. In a given state $\mathrm{S_t}$ and for a specific action $\mathrm{A_t}$, the payoff distribution associated with the optimal policyπ^*should align with the target payoff distribution derived from the distribution corresponding to the subsequent state $\mathrm{S_{t+1}}$ and action $\mathrm{a_{t+1}^* = \pi^*(S_{t+1})}$. Distributional DQN initially constructs a support vector for the target distribution $\mathrm{d_t'}$ and subsequently minimizes the KL divergence between the distribution $\mathrm{d_t}$ and the target distribution $\mathrm{d_t'}$. See equations (6) and (7) for target distributions.

$$d_t' \equiv \left(R_{t+1} + \gamma_{t+1} z, p_{\bar{\theta}}(S_{t+1}, \bar{a}_{t+1}^*)\right) \tag{6}$$

$$D_{KL}(\phi_z d_t' \parallel d_t) \tag{7}$$

Where $\mathrm{\phi_z}$ represents the projection of the target distribution onto the fixed support z, and $\mathrm{\bar{a}_{t+1}^* = argmax q_{\bar{\theta}}(S_{t+1}, a)}$ is the greedy action corresponding to the mean action value function at state $\mathrm{S_{t+1}}$, see equation (8).

$$p_{\bar{\theta}}(S_{t+1}, a) = z^T p_\theta(S_{t+1}, a) \tag{8}$$

A deep neural network models the parameterized distributions, and the parameters θ and θ^- are employed to formulate the current value distribution and the target distribution. The model's network structure resembles the DQN, with a critical distinction: the output is transformed from an action-value function to FDASFUGZ\SGF. Additionally, softmax is individually applied to each output action to guarantee proper distribution normalization for each action.

4 Distributed Deep Reinforcement Learning Based Approach for Transit Signal Priority Control

4.1 State

The design of the state space must consider the intersection's traffic information for a comprehensive and effective depiction of traffic conditions. This traffic information

encompasses various aspects: each inlet and outlet lane of the intersection, dedicated lanes for buses and social vehicles, the respective waiting times for buses and social vehicles, the length of the vehicle queue at the current intersection, and changes in signal phases. To precisely capture the bus's positional information, lanes are partitioned into cells of uniform size, describing the bus's position within the lane through cell P. The current state of the intersection is acquired by collecting and encoding the state into a state vector. This state vector, represented by an array of length j, assigns each element to a state within the state space. The state vector S_p is shown in equations (9) and (10).

$$S_p = \left(S_1, S_2, \cdots\cdots, S_i, \cdots\cdots, S_j\right) \tag{9}$$

$$S_i = \left(P_b, P_c, L, C_r, C_g\right) \tag{10}$$

Where S_i represents the state space of the current intersection in phase i. $\mathrm{P_b}$ denotes the bus's position, P_c signifies the location of the social vehicle, L represents the queue length at the current intersection, C_r indicates the duration of the red in the current phase, and C_g signifies the duration of the green in the current phase.

The elements in the array are encoded to represent various state information. For each vehicle, its state is mapped to the corresponding position in the state array based on its lane position, the lane ID, and other relevant information. The specific state coding includes: 1) Mapping the vehicle position on the lane to different lane cells by inverting and mapping the vehicle position. Each lane is 750m long, and to more accurately describe the current traffic conditions at the intersection, the first four cells are all 7m in length, while cells 5-10 are 12m, 20m, 40m, 60m, 240m, and 350m in length, respectively; 2) Combining the lane group and cell where the vehicle is located into a string and converting it to an integer to create a unique identifier for the vehicle location. For instance, if the vehicle is in the third lane cell of the second lane, the bus's location is represented as 23, enabling it to be pinpointed in the state array; 3) Calculating the position of the bus in the state array based on the lane group and cell where it is located, and marking it as 1 to indicate that the cell is occupied by the bus, otherwise, it is marked as 0. A schematic diagram illustrating the bus location information is presented in Figure 2.

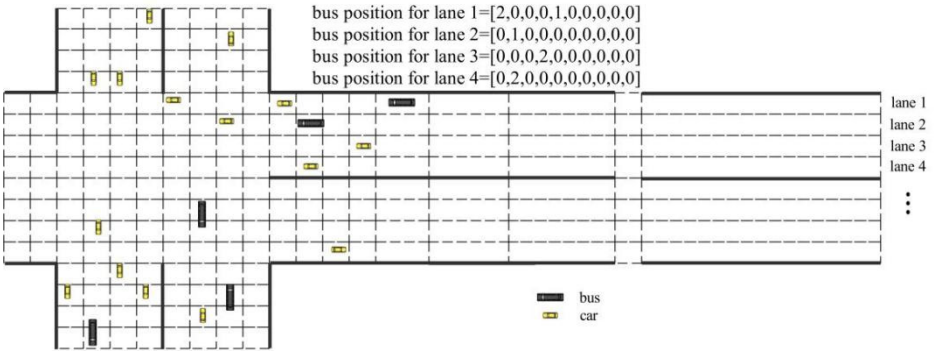


Fig. 2. Schematic of bus location information

4.2 Action

The definition of the action space must comprehensively consider the signal control strategy at the intersection, ensuring that the Agent can make effective decisions in diverse scenarios. The

fundamental structure of the action space comprises a set of signal phase configurations at intersections, denoted as F = {NSG, NSLG, EWG, EWLG}. Each action signifies a distinct phase setting, and each configuration corresponds to a green allocated to a different travel direction. NSG represents a green for the north-south direction, NSLG indicates a green for a north-south left turn, EWG signifies a green for the east-west direction, and EWLG represents a green for an east-west left turn. The action space can consist of integer-numbered phase configurations, where each integer corresponds to a distinct signal phase setting. The initial traffic light duration for each phase is set to 100 seconds. As the Agent learns iteratively, the Agent determines the duration of each phase based on the reward function and the prevailing traffic conditions at the intersection. Recognizing the time needed for pedestrians to reach the central safety island, the minimum green duration is established at 12 seconds, while the maximum green duration is set at 35 seconds. After each green interval, a yellow light phase of 3 seconds is implemented, and the yellow light transitions to red at its conclusion. If the duration is below the minimum green duration, the minimum duration is enforced; if it surpasses the maximum green duration, an immediate transition to the next phase occurs. The epsilon-greedy strategy is employed to strike a balance between exploration and exploitation when the Agent chooses an action. This facilitates the gradual development of knowledge regarding the optimal phase configuration throughout the learning process.

4.3 Reward

The formulation of the reward function must consider various factors, such as traffic flow efficiency, bus operating delays, and passenger travel experience. This ensures that the Agent, during the learning process, makes decisions aimed at the overall optimization of the system.

Firstly, the reward function should prioritize overall traffic flow efficiency at the intersection. In this context, penalty terms based on vehicle waiting time and queue length at the intersection can be considered to incentivize the Agent to implement measures that reduce congestion. The Agent will receive positive reinforcement by minimizing waiting time and queue length, reflecting its contribution to overall traffic smoothness. Secondly, in achieving TSP control, the delay of a particular bus at the current intersection should not be the sole consideration. Instead, the overall delay of multi-bus priority requests at the current intersection should be considered. Therefore, the reward term for the total waiting time of buses should be included. Ensuring that buses receive more green time at intersections or reducing their waiting time will earn positive rewards for Agents, motivating them to optimize signal control to meet bus priority requests. Finally, in considering the travel experience of all passengers at the intersection, the reward function should also incorporate delay considerations for social vehicles. This ensures the reduction of bus delays without adversely affecting the normal movement of social vehicles. In summary, the reward function is defined as a weighted average of the following rewards:

(1) The delay time d_b for a multi-bus request at the current intersection, denoted as d_{bt}, see equation (11), is calculated as the total waiting time for all buses at the intersection at the previous time t minus the waiting time for all buses at the intersection at the current time step t'.

$$d_b = \sum_{x=1}^{m}(d_{btx} - d_{bt\prime x})/m \tag{11}$$

(2) The average delay time d_c of all social vehicles at the current intersection, denoted as d_{ct}, see equation (12), is calculated as the total waiting time for all social vehicles at the intersection

at the previous time t minus the waiting time $d_{ct'}$ for all social vehicles at the intersection at the current time step t'.

$$d_c = \sum_{y=1}^{n}\left(d_{cty} - d_{ct'y}\right)/n \tag{12}$$

(3) The vehicle queue length at the current intersection denoted as L, see equation (13), is calculated as the difference between the vehicle queue length L_t at the intersection at the previous time t and the vehicle queue length $L_{t'}$ at the current time t'.

$$L = L_t - L_{t'} \tag{13}$$

Therefore, the final reward function R is shown in equation (14).

$$R = k_1 d_b + k_2 d_c + k_3 L \tag{14}$$

where k_1, k_2, k_3 are the weighting factors. $k_1 + k_2 + k_3 = 1$.

4.4 Improved Distributed Deep Reinforcement Learning Network Model

The enhanced distributed deep reinforcement learning network model is illustrated in Fig. 3. The position matrix and speed matrix of the buses, acquired by discretizing real-time traffic information, serve as inputs. These inputs undergo feature extraction and transformation into one-dimensional vectors through convolutional neural networks. These vectors capture spatial information, including lane occupancy, vehicle waiting time, queue length, and more. The unfolded vectors are utilized as input data, and following the forward propagation process of the neural network, the distribution of valuations for each action is ultimately obtained. During each interaction with the environment, the Agent stores the state, action, reward, next state$\langle s_t, a_t, r_t, s_{t+1}\rangle$, and other information of each time step in the experience pool. These experiences are randomly sampled for learning during training. Simultaneously, by introducing the concept of priority, we prioritize selecting more beneficial experiences for learning, thereby enhancing training efficiency and stability. Furthermore, unlike the traditional mean squared error loss function, the enhanced model is trained using the Huber loss function to enhance robustness to outliers. The Huber loss function, designed for regression problems, distinguishes between mean squared error (MSE) and mean absolute error (MAE), exhibiting reduced sensitivity to outliers. In handling more significant errors, the Huber loss function leverages the advantages of mean square error, whereas, for smaller errors, it aligns more closely with mean absolute error. This adaptability allows the model to handle diverse training scenarios better, resulting in improved training performance and experimental results. The Huber loss function is show in equation (15).

$$L_\delta\left(y, f(x)\right) = \begin{cases} \frac{1}{2}\left(y - f(x)\right)^2, & if\,|y - f(x)| \le \delta \\ \delta\left(|y - f(x)| - \frac{1}{2}\delta\right), & otherwise \end{cases} \tag{15}$$

In this equation, y represents the actual observed target value, $f(a)$ denotes the predicted value of the model, and δ is a hyperparameter that determines the point at which the Huber loss function transitions into a compromise between squared error and absolute error. If the difference between the predicted and actual values is less than or equal to 5, the Huber loss function incorporates squared error. However, when the difference exceeds 8, the Huber loss function adopts a linear error with a slope 5. This choice reduces sensitivity to outliers, enhancing robustness compared to MSE.

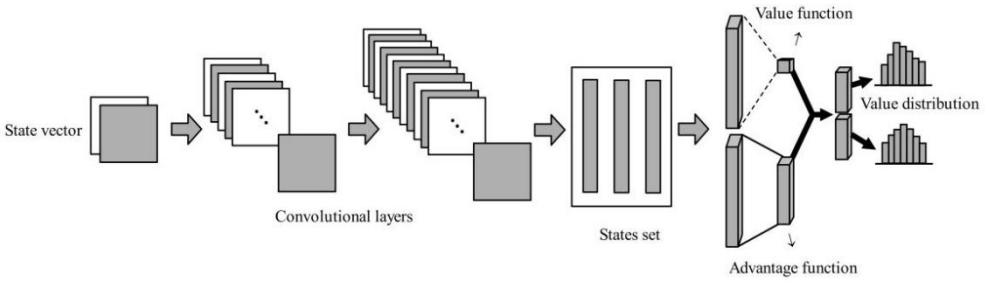


Fig. 3. Improved distributed deep reinforcement learning network model

5 Experimental results and analyses

In this study, we construct a simulation environment for TSP control using the SUMO (Simulation of Urban MObility) traffic simulation software. SUMO offers a practical platform for simulating dynamic changes in urban traffic flow, enabling users to test various traffic signal control strategies in the simulation. The simulation environment incorporates authentic road networks, vehicle flows, and intersections, ensuring the experiment's authenticity and reliability. This chapter initially introduces the simulation environment and parameter configurations for the experiment. Subsequently, it assesses the efficacy of the distributed deep reinforcement learning algorithm, Distributional Dueling Deep Q-Network based on Prioritized Experience Reply (DPDDQN), in TSP control within the SUMO traffic simulation software. Furthermore, it compares and analyzes this algorithm against other deep reinforcement learning algorithms, namely, DQN, Dueling DQN, and DQN with PER algorithm.

5.1 Simulation environment and parameter settings

The experiment is conducted using the microscopic traffic simulation platform SUMO for simulation experiments, and the hardware and software configuration information is presented in Table 1. The Traci (Traffic Control Interface) interface module provided in SUMO facilitates online interaction with the simulation platform. This module enables users to manipulate the traffic simulation online and retrieve various values of the objects in the simulation. The algorithmic model is implemented using the Tensorflow-gpu deep learning framework, and the detailed settings for the traffic road network simulation are as follows.

Table 1. Hardware and software configurations

Configuration	Parameters
CPU	Intel(R) Xeno(R) Gold 6136 24-core @3GHz
Memory	64GB
Graphics Card	NVIDIA GeForce RTX 3090
Graphics Memory	24GB
Operating System	Windows 10
SUMO	1.17.0
Python	3.7.16
Tensorflow	2.0

Figure 4 depicts a simulated intersection of four roads measuring 750 × 750 m^2. It includes four inlet and four exit lanes. The individual lanes have a width of 3.2m, and there is a bus stop, 10m long, located in the rightmost lane of each route, positioned 100m from the stop line. Each inlet lane defines the possible directions that a car can follow: the leftmost lane is used only for left turns; the rightmost lane is dedicated to right turns and straight ahead; the two middle lanes are dedicated to straight ahead. The layout of the traffic signal system is as follows: the left-turn lane is provided with a dedicated traffic signal, while the other three lanes share the same traffic signal. The inlet lanes of the intersection are discretized and divided into cells. These cells are used to mark the presence or absence of bus vehicles. There are 20 cells in each direction, 10 of which are along the leftmost lane, while the other 10 are spread across the other three lanes. There are a total of 80 cells for the entire intersection.

In each experimental round, 1000 vehicles are generated, and the details of the vehicle information are presented in Table 2. To better emulate real-world traffic scenarios, vehicle arrival times are determined based on the Weibull distribution with a shape parameter of 2, exhibiting a rapid increase in the middle moment followed by a gradual decrease. The generated vehicles consist of 20% buses and 80% social vehicles, primarily cars. Both buses and social vehicles have a 75% probability of traveling straight and a 25% probability of turning left or right. Every vehicle shares the same probability of being generated at the starting position of each exit. Due to the random generation of vehicles in each experimental round, the likelihood of vehicles arriving in precisely the same layout situation is low.

Table 2. Vehicle information sheets

Vehicle type	Length (m)	Initial speed (m/s)	Acceleration (m/s^2)	Maximum speed(m/s)
bus	8.5	10	0.5	25
car	5	10	0.2	20

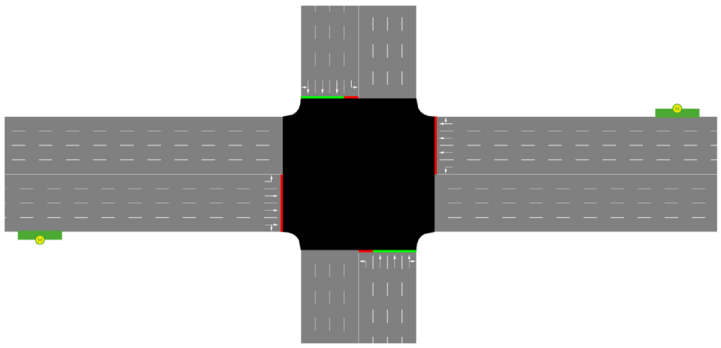


Fig. 4. Schematic diagram of the intersection simulation area

As this experiment requires comparison with three other algorithms, identical network structures are employed for each model to ensure experiment validity. The entire experiment comprises 800 iterations, with a maximum of 5400 steps(the duration of each episode, with 1 step = 1 second) per iteration, conducted over 100 rounds. Other fundamental parameter settings are detailed in Table 3.

Table 3. Basic parameter settings

Parameters	Parameter settings
Epsilon ε	Initial value is 1, final value is 0.01
The number of hidden layers in the neural network N_l	4
The number of neurons per layer in the neural network W_l	400
The number of episodes E_p	100
learning rate α	0.001
The number of iterations to replace Parameters T_e	800
the min number of samples needed into the memory M_{min}	600
the max number of samples needed into the memory M_{max}	50000
Size of state space S	80
Size of action space A	4
the gamma parameter of the Bellman equation γ	0.75

Apart from the aforementioned fundamental parameters, the PER algorithm incorporates a stochastic scaling factor μ = 0.6, adjusted importance sampling weight σ = 0.4, and specifies the value distribution range of DPDDQN as $v_{min} = -50$, $v_{max} = 0$, the number of atoms $N_{atoms} = 50$. Following continuous experimental verification, the weights for the reward function were ultimately established as k_1=0.4, k_2=0.3, and k_3=0.3.

5.2 Experimental evaluation and analysis of results

To assess the effectiveness of the DPDDQN algorithm, this section primarily analyzes its performance in TSP control, comparing it with DQN, DQN with PER, and Dueling DQN. Graphs illustrating the performance of each algorithm across various metrics are provided to visualize their effectiveness in managing TSP. The metrics encompass cumulative rewards, average delays for buses and social vehicles, average queue lengths, and per capita delays. When calculating per capita delay, it is assumed that the average number of passengers per social vehicle is 2. In practical scenarios, buses often have 15-20 times the passenger capacity of cars. For this study, we assume buses have 20 times the passenger capacity of social vehicles. Thus, the per capita delay, d_o, is defined in equation (16):

$$d_o = (\sum_{x=1}^{m} 40 * d_b + \sum_{x=1}^{n} 2 * d_c)/(40m + 2n) \quad (16)$$

The loss curves in deep reinforcement learning are typically non-smooth during training due to the non-convex nature and highly complex non-linear properties of optimization problems in this domain. Despite the lack of smoothness in the loss curve, the primary concern is usually the overall trend of the loss values. During training, if the model performance improves, the overall trend of the loss values should decrease. Figure 5 displays a graph of the prediction error of the enhanced neural network, exhibiting an overall decreasing trend and a gradual leveling off of the prediction error. Additionally, since scores in each set constantly fluctuate, a more intuitive observation of the Agent's reward acquisition through testing is facilitated by plotting the reward value curve, resulting in a smoother curve. From Fig. 6, it is evident that the DPDDQN demonstrates higher reward values and less fluctuation compared to the other three algorithms. This indicates that the algorithm is more stable during testing, less vulnerable to random factors, and able to optimise its control strategy faster during the learning process.

Moreover, Figure 7 illustrates that the improved algorithm yields favorable experimental results regarding both the average delay and queue length of buses and social vehicles and the per capita delay at the current intersection. This indicates that the algorithm reduces bus delays and

simultaneously decreases the delays and queue lengths of social vehicles, with a substantial reduction in per capita delay. Table 4 compares each index of DPDDQN with the other three algorithms. From Table 4, it is evident that the improved algorithm reduces the average delay of buses by 35.07%, the average delay of social vehicles by 22.42%, the average queue length by 11.52%, and the per capita delay by 26.61% compared to the other three typical algorithms with the best results. The experimental results indicate that the improved algorithm outperforms the other three algorithms.

Table 4. Comparison of each metric for each algorithm

Algorithm	Average reward	Average bus delay(s)	Average social vehicle delay(s)	Queue length	Average per capita delay(s)
DQN	-12.77	73.25	930.28	6.43	185.03
DQN with PER	-7.54	59.93	771.72	6.15	152.77
Dueling DQN	-8.23	53.43	719.61	5.99	140.32
DPDDQN	-6.21	34.69	558.26	5.30	102.98

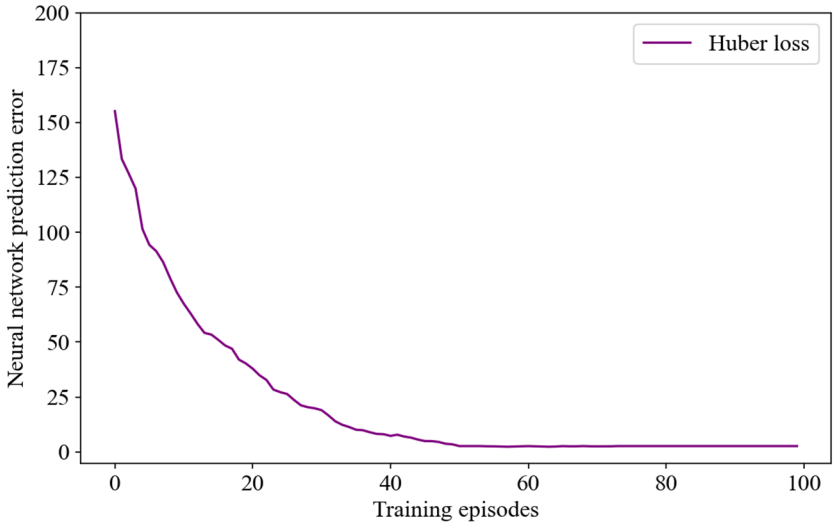


Fig. 5. Neural network prediction error curve

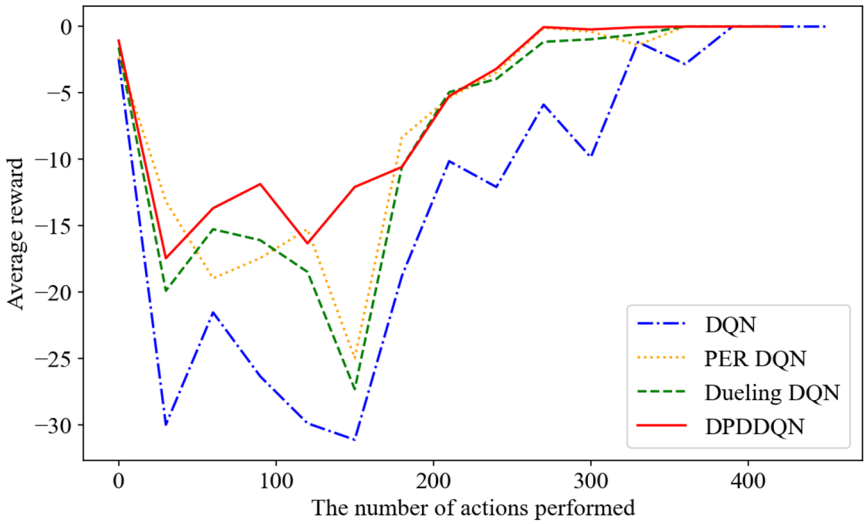
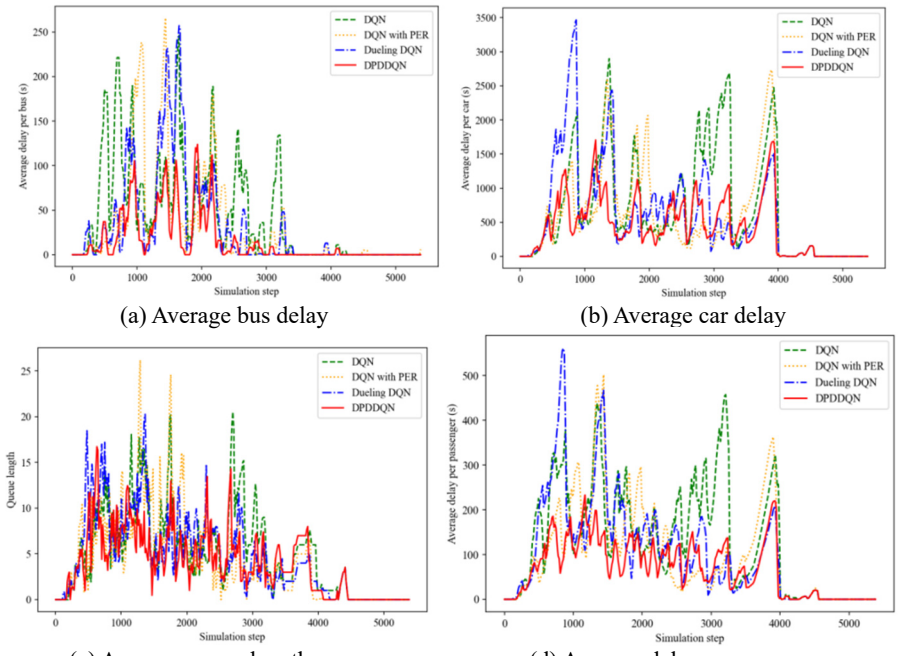


Fig. 6. Comparison of average reward values of the algorithms



(a) Average bus delay

(b) Average car delay

(c) Average queue length

(d) Average delay per passenger

Fig. 7. Pair of traffic scenarios for each algorithm

6 Conclusion

This paper explores the TSP control problem and introduces a TSP method grounded in deep reinforcement learning. The enhanced algorithm amalgamates the strengths of Dueling DQN, Distributional DQN and PER, substantially improving control effectiveness.

Firstly, unlike previous TSP control based on deep reinforcement learning, this paper abandons conventional actions like green extension or red truncation. Instead, it mitigates bus delays and traffic congestion through the Agent's flexible adjustment of signal duration. Secondly, the structure of Dueling DQN is incorporated, enhancing the modeling capability for complex environments by decomposing the value function into state value and advantage function. Distributional DQN is introduced to offer a more comprehensive understanding of the potential benefits of different behaviors by modeling the value distribution. This enriches information for the model, enabling better adaptation to complex and dynamic traffic environments. Additionally, the paper employs the Huber loss function for training, which handles outliers in the reward signal more effectively than the traditional mean square error, enhancing training stability and convergence speed. Experimental results demonstrate a significant improvement in model performance with this enhancement. The improved algorithm handles multiple bus requests in the same control cycle. It effectively reduces the delay of bus and social vehicles, and the queue length at intersections adjusts signal duration flexibly and adapts better to complex traffic scenarios. This provides a more flexible and efficient solution for applications in real traffic management.

In summary, the TSP control method based on deep reinforcement learning proposed in this paper has achieved significant innovations and advantages in algorithm structure, training methods, and application scenarios, offering novel ideas and approaches for developing the bus priority field. Future research can focus on the following aspects: 1) Deepening the understanding and improvement of the deep reinforcement learning model. By optimizing the algorithm structure, introducing more environmental features, and considering diverse reward mechanisms, the model's adaptability to complex urban traffic environments can be enhanced, making it more versatile and robust. 2) This study only considered expediting buses through the current intersection without affecting the traffic of social vehicles, neglecting overall scheduling delays of buses and queuing issues such as overflow at downstream bus stops. Future research could expand to a more comprehensive bus system scheduling optimization, considering cooperative scheduling between buses to enhance overall bus transport efficiency. 3) Future research should enhance validation and application in real-world scenarios. Conducting field experiments in real traffic networks can verify the feasibility and effectiveness of the algorithms in real traffic management, providing stronger support for practical applications.

Acknowledgments. This work was supported by the Science and technology plan projects of Inner Mongolia Autonomous Region(Grant No. 2020GG0104).

References

[1] Bouktif S, Cheniki A, Ouni A. Traffic Signal Control Using Hybrid Action Space Deep Reinforcement Learning[J]. Sensors, 2021,21(7), 2302.

[2] Ducrocq R, Farhi N. Deep Reinforcement Q-Learning for Intelligent Traffic Signal Control with

Partial Detection[J]. International Journal of Intelligent Transportation Systems Research, 2023,21(1), 192-206.
[3] Wu C, Kim I, Ma Z. Deep Reinforcement Learning Based Traffic Signal Control: A Comparative Analysis[J]. Procedia Computer Science,2023, 220, 275-282.
[4] Cheng H K, Kou K P, Wong K I. Transit Signal Priority Control with Deep Reinforcement Learning[C]//2022 10th International Conference on Traffic and Logistic Engineering (ICTLE). IEEE, 2022: 78-82.
[5] Long M, Zou X, Zhou Y, et al. Deep reinforcement learning for transit signal priority in a connected environment[J]. Transportation Research Part C: Emerging Technologies, 2022, 142: 103814.
[6] Shen W C, Zou L, Deng R S, et al. A TSP Control Method Based on Deep Reinforcement Learning[J].2023.Applied Sciences-Basel, 13(11), 6772.
[7] Lee J, Shalaby A. Rule-based transit signal priority control method using a real-time transit travel time prediction model[J]. Canadian Journal of Civil Engineering, 2013, 40(1): 68-75.
[8] Zhang C L, Yang X D, Wei J M, et al. Cooperative Transit Signal Priority Considering Bus Stops Under Adaptive Signal Control[J]. IEEE Access,2023,11:66808 - 66817.
[9] Sun X, Lin K, Jiao P, et al. Signal timing optimization model based on bus priority[J]. Information, 2020, 11(6): 325.
[10] Hamid B, Mohammad H. Proposing a kinematic wave-based adaptive transit signal priority control using genetic algorithm[J].IET Intelligent Transport Systems,2023,17(5):912-928.
[11] Wei H, Chen C, Zheng G, et al. Presslight: Learning max pressure control to coordinate traffic signals in arterial network[C]//Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2019: 1290-1298.
[12] Wei H, Xu N, Zhang H, et al. Colight: Learning network-level cooperation for traffic signal control[C]//Proceedings of the 28th ACM International Conference on Information and Knowledge Management. 2019: 1913-1922.
[13] Sun H, Chen C L, Liu Q, et al. Traffic Signal Control Method Based on Deep Reinforcement Learning [J]. Computer Science,2020,47(02):169-174.
[14] Shang C L, LIU X M, Tian Y L, et al. Priority of Dedicated Bus Arterial Control Based on Deep Reinforcement Learning [J]. Journal of Transportation Systems Engineering and Information Technology,2021,21(03):64-70.
[15] Mnih V, Kavukcuoglu K, Silver D, et al. Human-levelcontrol through deep reinforcement learning[J]. Nature, 2015 .518(7540):529-533.
[16] Wang Z, Freitas N D, Lanctot M. Dueling network architecturesfor deep reinforcement learning[C]//Proceedings of the International Conference on Machine Learning. New York, USA,2016: 1995-2003.
[17] Schaul T, Quan J, Antonoglou, et al. Prioritized experience replay[C]// Proceedings of the 4th International Conference on Learning Representations. San Juan, Puerto Rico.2016:322-355.
[18] Bellemare M G, Dabney W， Munos R. A distribution-alperspective on reinforcement learning[C]//Proceedings of the 34th International Conference on Machine Learning.JMLR.org,2017:449-458.