Electronic Research Archive

http://www.aimspress.com/journal/era

ERA, 32(6): 3843–3866.
DOI: 10.3934/era.2024174
Received: 29 February 2024
Revised: 23 April 2024
Accepted: 06 May 2024
Published: 12 June 2024

Research article

Uniformity of markov elements in deep reinforcement learning for traffic signal control

Bao-Lin Ye[1,2,*], Peng Wu[1,2], Lingxi Li[3] and Weimin Wu[4]

[1] School of Information Science and Engineering, Jiaxing University, Jiaxing 314001, China
[2] School of Information Science and Engineering, Zhejiang Sci-Tech University, Hangzhou 310018, China
[3] Elmore Family School of Electrical and Computer Engineering, Purdue University, Indianapolis 46202, USA
[4] State Key Laboratory of Industrial Control Technology, Zhejiang University, Hangzhou 310027, China

* **Correspondence:** Email: yebaolin@zjxu.edu.cn.

Abstract: Traffic signal control (TSC) plays a crucial role in enhancing traffic capacity. In recent years, researchers have demonstrated improved performance by utilizing deep reinforcement learning (DRL) for optimizing TSC. However, existing DRL frameworks predominantly rely on manually crafted states, actions, and reward designs, which limit direct information exchange between the DRL agent and the environment. To overcome this challenge, we propose a novel design method that maintains consistency among states, actions, and rewards, named uniformity state-action-reward (USAR) method for TSC. The USAR method relies on: 1) Updating the action selection for the next time step using a formula based on the state perceived by the agent at the current time step, thereby encouraging rapid convergence to the optimal strategy from state perception to action; and 2) integrating the state representation with the reward function design, allowing for precise assessment of the efficacy of past action strategies based on the received feedback rewards. The consistency-preserving design method jointly optimizes the TSC strategy through the updates and feedback among the Markov elements. Furthermore, the method proposed in this paper employs a residual block into the DRL model. It introduces an additional pathway between the input and output layers to transfer feature information, thus promoting the flow of information across different network layers. To assess the effectiveness of our approach, we conducted a series of simulation experiments using the simulation of urban mobility. The USAR method, incorporating a residual block, outperformed other methods and exhibited the best performance in several evaluation metrics.

Keywords: deep reinforcement learning; traffic signal control; Markov elements; residual block

1. Introduction

In recent years, the rapid expansion of urban areas has exacerbated the problem of traffic congestion [1]. This predicament has resulted in significant economic drawbacks, including heightened fuel consumption and prolonged travel times [2, 3]. Therefore, the development of advanced traffic signal control (TSC) methods has emerged as a crucial solution to not only reduce traffic congestion but also alleviate its costly consequences [4, 5]. Efficient TSC methods can potentially boost the capacity of intersection traffic considering they directly influence vehicle access [6, 7]. Hence, a multitude of scholars are engaged in enhancing the resolution to the TSC problem.

The fixed-time control centered on the Webster's formula has been widely applied at urban intersections [7–9]. This formula calculates the duration of green lights for each phase according to historical traffic flow and relevant data. However, this method encounters issues in regulating traffic effectively in instances of drastic traffic flow fluctuations at intersections. In response to this, Yu et al. [10] introduced a time-slot-based signal mechanism model encompassing a generalized cycle structure, allowing the omission of certain phases in partial sub-cycles. Further studies aim at refining the Webster method by developing novel formulas that factor in vehicle delay and queue length in determining the optimal cycle length [11]. Simulation results have affirmed that the improved Webster approach aptly resolves the predicament of overestimating the green cycle length.

Recently, the incorporation of reinforcement learning (RL) theory into the TSC problem has facilitated noticeable advancements [12, 13]. Additionally, deep learning (DL) techniques, renowned for their superior performance in problem-solving across various domains, have gained considerable recognition. Research suggests that, as the fusion of RL and DL techniques, DRL can greatly enhance traffic management in TSC [14]. Utilizing the Deep Q-Network (DQN) framework, the DRL-based TSC solution has significantly outperformed traditional TSC methodologies [15]. This approach affords a reduction in average vehicle delay and optimization of intersection capacity. In the domain of DRL, the agent discerns the environment's state and interacts with it through the execution of actions, receiving instant rewards in the process. The precise formulation of these pivotal elements is particularly fundamental, as they directly impact the agent's decision-making strategy and overall behavior within the DRL ecosystem. Particularly in TSC, a common hurdle is to appropriately craft the states, rewards, and actions to facilitate the agent's learning of an optimal action-selection policy. Ambiguous definitions of elements such as states and actions can disrupt the agent's optimal learning trajectory, leading to less than optimal policies. Moreover, the construction of states and actions needs to reflect practical traffic scenarios for relevant applicability. Unfortunately, many existing DRL frameworks are limited by manually designed state and action, compromising the agent's capacity for optimal decision-making.

This paper proposes a DRL algorithm framework that defines the state and action spaces, aiming at optimizing the process of action execution guided by states. This straightforward definition approach allows agents to receive direct feedback from the environment and adjust their behavior accordingly. This methodology is intended to facilitate a more efficient interaction between the agent and its environment, streamlining the learning process in DRL algorithms. Moreover, we stress the significance of maintaining simplicity when defining states and actions. Evading overly convoluted state representations and identifying the essential information to capture intersection states is

paramount in tackling the TSC issue. This facet of consistency fosters practicality and efficiency during the design process. This design methodology simplifies operations by directly linking current state to the formation of actions. By embracing a straightforward state representation during training, the agent can investigate a smaller space, leading to enhanced learning efficiency. Additionally, this style of action design bolsters training stability, triggering quicker convergence to peak performance. This fortifies the correlation between the environment state and the actions undertaken by the agent. Consequently, the method predicated on direct convergence outlined in this paper, is expected to be considered as the optimal solution strategy for the TSC challenge in intelligent transportation systems. This paper's major contributions can be summarized as follows:

1) We categorize the number of vehicles on a lane into inner states (queued vehicles) and outer states (moving vehicles) and assign different priority weights to these states. To enhance training efficiency, we have developed a effective reward function directly associated with the internal state. This enables the agent to receive direct feedback in response to changes in environmental states.

2) We propose a design approach that utilizes a formula with weighted coefficients to update actions related to state definitions in a unified manner. The design method aims to maintain consistency between states and actions, allowing the agent to dynamically update its action selection for the next step using the formula when perceiving changes in the environment's state at the current step. This method significantly enhances the direct interaction of information between the agent and the environment.

3) To further enhance the feature extraction capability of the proposed method, we employ a residual block into the Double Deep Q-Network (DDQN) framework. This residual connection technique strengthens the feature representation between input and output information in the convolutional neural network, increasing the convergence speed of the algorithm and the robustness during model training.

The remaining sections of this paper are organized as follows. Section 2 reviews the relevant literature and discusses related works. Section 3 provides a brief introduction to the background knowledge and necessary preliminary work for the proposed method. Section 4 presents the problem formulation for the TSC model, along with the optimization objectives. Section 5 details the design of the proposed DDQN framework and residual block structure. In Section 6, we conduct traffic simulation experiments and ablation studies. We analyze and discuss the experimental results. Section 7 concludes the paper and provides a discussion of potential future work.

2. Related work

The exceptional performance of RL methods in recent years has ignited a considerable surge of research interest across multiple disciplines. Notably, it has made impressive strides in areas like energy costs and autonomous driving [16], especially in the aftermath of the rise of DL. Recently, researchers have gravitated toward DRL as a solution to address the challenges inherent to the TSC domain. The aim is to transition from traditional control systems to intelligent TSC systems [17–19]. A widely adopted method, first implemented by Gregurić et al. [20], comprises a unique traffic state representation that utilizes a series of matrices. This representation divides each lane into homogeneously distributed cells that carry information on the vehicles' presence or absence, their velocity or acceleration, and the current phase or duration of the green light. These vectors are

arranged to construct an image-like matrix, which is subsequently processed using convolutional neural networks (CNN) [21]. Haddad et al. [22] proposed a method to efficiently use information from the intersection road network by selecting agent behavior based on local lane state information. In parallel, a few research groups have experimented with simplified state spaces to illustrate vehicle data using metrics like the number of vehicles and queue length [23, 24]. In the latter approach, the state is depicted by the number of vehicles facilitated, with the reward corresponding to the length of the vehicle queue [25].

Indeed, future work should focus on developing the connections between state and action, as well as between state and reward, to facilitate the flow of information among the elements in these Markov processes. The TSC methods based on a hybrid action space generally implement each phase for a fixed length of time [26, 27]. The agent considers whether it should transition to the next stage or extend the existing stage by a certain amount of time. Although expanding the action space in this way can improve the response speed, it will significantly enlarge the action gap, which will affect the convergence accuracy of the model. To achieve a balance among the factors in these conflicting action spaces, the proposed method leverages queue length within state definition to design actions, thus ensuring a direct correlation between action execution and state changes. This is the issue addressed in this paper.

In a TSC system, reward as the feedback of the environment after the agent performs an action, is the key to the evaluation of action value. Many definitions of rewards have been deployed in the TSC system, including changes in total waiting time, queue length, and accumulation of vehicle delays [28]. Bouktif et al. [25] found that the queue length of the reward vehicle has higher performance. Building on this finding, our study further refines the state and reward design by targeting the unified method [29]. The selected reward combined with the number of vehicles in the queue is used to accelerate convergence. This design philosophy emphasizes the direct implementation of reduced queue vehicles as a reward, which helps to improve the convergence speed. This is another issue addressed in this paper. In summary, the current work faces two main issues. 1) In the reinforcement learning-based Markov processes, there is an over-reliance on manually crafted components, failing to establish a consistent flow of information during the agent's learning process. 2) In the current TSC methods, the design of the action space does not closely correlate with changes in the environmental states, preventing agents from efficiently converging to the optimal signaling strategy.

The proposed method is inspired by the work of Bouktif et al. [25], which aims to streamline the constitution of state, action, and reward by incorporating the state into the formulation of the action space and reward function. This integration allows the TSC model to enable the agent to execute actions pertinent to queue length while perceiving the environmental state during training. Concurrently, the feedback in the form of rewards assesses the value of the actions, thus facilitating direct information exchange between the agent and the environment.

3. Theoretical background and preliminaries

In this section, we will briefly introduce the concept and theoretical basis of DRL algorithm used in this paper.

3.1. Reinforcement learning

The RL is an algorithm that focuses on learning how to maximize rewards by interacting with the environment. In RL, an agent continuously receives feedback from the environment to learn an optimal policy. The RL algorithm is commonly described using Markov decision processes (MDP) and characterized by a set of terms (S, A, R, P). S is the set of all states, which consists of a finite set of Markov states that the agent can utilize to assess the relative quality of the current environment. The agent typically relies on the current state representation to make decisions for the subsequent state. A is the set of all actions. At each time step t, the agent selects the current optimal action a_t from the action space A following a policy π to maximize the long-term reward. The agent chooses actions a_t according to the policy π within a time step t and receives a feedback reward r_t from the environment. The agent can adjust the decision of the next time step through the reward feedback of the environment, so that the whole has a better decision-making trend. For each state s_t , the transition probability $P(s_{t+1}|s_t)$ gives the probability of moving to state s_{t+1} by taking an action a_t.

To maximize the cumulative reward value, the agent engages in ongoing information interaction with the environment by following policies which determine the actions to be taken based on the current state. In this study, we employ the Q-learning algorithm, a model-free, value-based, and off-policy form of reinforcement learning. The Q-learning algorithm utilizes the Q-function to estimate the expected discounted reward for an action in a given state. The expression for the Q-function is as follows:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_t + \gamma \max_{A_{t+1}} Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)] \tag{3.1}$$

where α is learning rate that denotes the magnitude of the Q value update. The γ is discount factor that is used to balance current reward and future reward.

3.2. Deep reinforcement learning

The traditional Q-learning algorithm uses tables to store state-action pairs of Q-values, which becomes inefficient and expensive for high-dimensional or continuous spaces. To address this, a deep neural network (DNN) method is proposed for nonlinear function approximation. It captures complex state features to better approximate Q-values. DQN is a common DRL algorithm, with an input layer representing the state (image or vector). Hidden layers learn features, and the output layer provides the Q-value for a state-action pair. The DQN uses a nonlinear neural network function approximator with weight $\hat{\theta}$ and an experience replay memory by storing quadruples (S, A, R, S') into the memory as historical experiences at each time step. In the process of learning and training, the DQN is updated by randomly sampling a small batch of tuples in the memory bank at intervals, and the target value y_t^{DQN} is shown in formula (3.2):

$$y_t^{DQN} = R_t + \gamma \hat{Q}(S_{t+1}, \arg\max_a \hat{Q}(S_{t+1}, a; \hat{\theta}); \hat{\theta}), \tag{3.2}$$

where $\hat{Q}$ represents the q-value of the state-action pair. The $\hat{\theta}$ is used both to estimate Q and to select the next action a. The goal of the training and learning process of the DQN algorithm is to build a Q-network that can accurately predict the target value in Eq (3.3). Therefore, the learning objective of

the agent is to minimize the loss function $L_t(\theta)$,

$$L_t(\theta) = \mathrm{E}[(y_t^{DDQN} - Q(s_t, a_t; \theta_t))^2], \tag{3.3}$$

where θ_t represents the network parameters at time step t taking action a_t and state s_t. The DDQN algorithm was introduced to mitigate the problem of overestimation in Q-value calculation with a single target network. It involves using a main network to determine the optimal action and estimating the action value using an additional pair of target networks. At the same time, it will be updated after some time steps by cloning the weight value θ of the main network. Formally, y_t^{DDQN} is defined as follows:

$$y_t^{DDQN} = R_t + \gamma \hat{Q}(s_{t+1}, \arg\max_a Q(S_{t+1}, a; \theta); \hat{\theta}), \tag{3.4}$$

where γ represents the discount factor for the Q value.

4. Problem definition

In this section, we provide an illustration of the TSC problem definition for our DRL formulation.

4.1. Environment

The environment is defined as four directions (i.e., North 'N', South 'S', West 'W', East 'E'). Among them, the north-south (NS) direction is three lanes, and the east-west (EW) direction is a single lane. In order to ensure the safety of road traffic, the traffic of vehicles does not interfere with each other. The vehicle traffic directions include: north-south direction straight (NS-S), north-south direction left turn (NS-L), north-south direction right turn and straight (NS-RS), east-west direction right turn and straight (WE-LSR). Figure 1 shows the environment structure, and the intersection entrance in the figure represents the traffic flow input point of each lane.

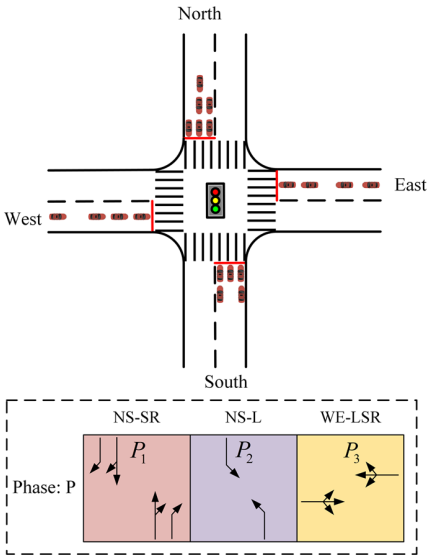

Figure 1. Schematic diagram of a single intersection.

4.2. Phase sequence of the signal

The phase P_n of a signal is defined as a certain set of signals (G for green light, R for red light) at an intersection. For instance, P_1 indicates that the NS-SR and NS-R lanes as shown in Figure 1 are executed with green light action, P_2 indicates that the NS-L lanes are executed with green light action, and P_3 indicates that the WE-LSR lanes are executed with green light action. In the definition of phase sequence, each phase will be executed in turn when it is green and the other phase is red within the same signal cycle. The execution order of phases is shown in the below panel of Figure 1.

4.3. Agent

In the DRL framework, the agent plays a vital role. It senses the current state of the environment, and based on the policy, selects the optimal action during a phase P_n. At each time step, the agent receives immediate feedback in the form of a reward. Figure 2 illustrates the flow of interaction between the agent and the environment in the TSC problem.

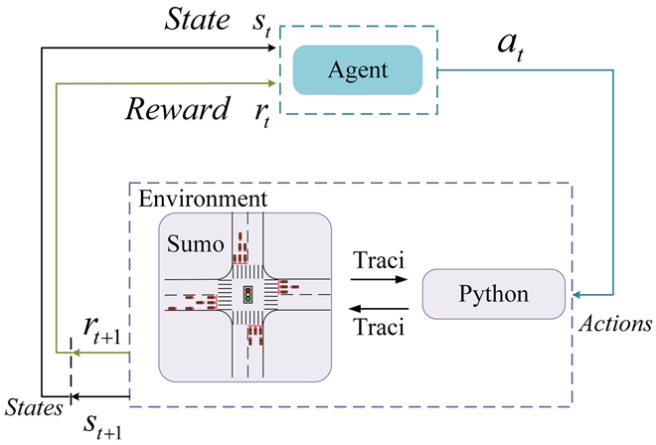

Figure 2. The process of the agent interacting with the environment.

4.4. Queue of vehicles

In this paper, we define "queuing vehicles" as a series of vehicles that have approached an intersection and are arranged in a specific order. Within our intersection scenario, vehicles fall into two distinct categories: the queue of vehicles stationed consecutively at the stop line, and the string of vehicles in active transit on the road. When the agent selects actions based on the given state, vehicles positioned before the stop line will be prioritized for passage. The transit vehicles, consequently, assume a lower priority in comparison to the immobile queuing vehicles. Within the framework of our TSC research, the queuing vehicles stationed before the stop line are allocated a higher weightage than their transit counterparts. To clarify the terminologies used throughout this paper, we have provided a summary in Table 1.

Table 1. Definition of intersection-related parameters.

Notation	Description
s	State
a	Action
r	Reward
P	Phase of signals
L	Number of lanes
V_l^{in}	The vehicle in the inner state area lane l
V_l^{out}	The vehicle in the outer state area lane l
t	Time step

5. Proposed method

In this section, our objective is to shed some light on the methods of applying the DRL framework to manage TSC at standalone intersections. In this study, based on the latest research methodologies in consistent state and reward design [25], we propose a novel design method that maintains consistency among states, actions, and rewards, named the uniformity state-action-reward (USAR) method for TSC. Compared to existing techniques, our method significantly accelerates convergence while maintaining cutting-edge accuracy. The main innovation of the USAR method lies in our pioneering fusion of state representation in the action definition formula (as given in Eq (5.3)) and the introduction of a weight coefficient w_{out} to enhance the coordination between actions and states. To optimize the convergence properties of the network model during training, we incorporate residual blocks into the traditional DDQN algorithm, which not only aids in network training but also further improves model performance.

Figure 3 delineates the employment of the DRL framework for proposed USAR method. For each time interval, the agent evaluates its environment and encapsulates the current state in the context of a vector. Owing to its underlying policy and the current state, the agent opts for the most appropriate action. The chosen action comprises of the phase sequence for the agent to observe and the duration of the green signal associated with it. To cater to the varying right-of-way circumstances of vehicles situated at different spots, the definition of our state relies on the queued vehicles waiting at the stop line, hence called inner state. As a contrast, any vehicle heading towards the stop line within the lane attributes it as an outer state. To ensure the TSC system efficiently restricts traffic congestion, vehicles within the inner state are given priority over others with a longer signal duration to speed up their passage through the intersection. Figure 4 elaborates the definition of the state more precisely. Thereafter, as a result of the action implantation, the agent attains the consequent state s_{t+1} and the subsequent reward r_t. To summarize, the tuple consisting of (s_t, a_t, r_t, s_{t+1}) is preserved in the memory, thus allowing the agent to practice mini-batch sampling at periodic intervals; this in turn enhances the agent's learning efficiency.

Bearing the larger picture in mind, the DRL framework involves several major components: the state representation, reward function, action space, and the agent architecture. The next subtopics will delve into these components in detail.

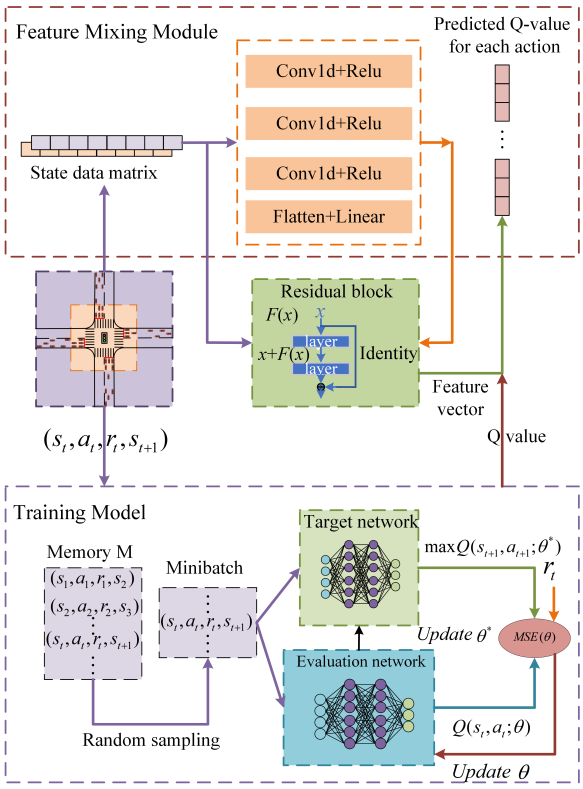

Figure 3. Double deep Q-network with residual block architecture.

5.1. State representation

The complexity of state representation directly influences the computational difficulty involved with DRL-based algorithms. Therefore, simplifying the state representation is crucial within a TSC system framed by DRL algorithms. In pursuit of a simple and consistent representation method, this paper will utilize the spatial distribution quantity of vehicles on the lanes within intersections as the components of the state representation, with specific configurations as follows:

1) At time step t, we define the sequence of vehicles queuing in front of the stop line as the inner state, while the vehicles currently driving on the road represent the outer state. The specific differentiation interval is illustrated in Figure 4.

2) The agent performs actions to correct the performance of the next state to achieve the purpose of channelling traffic. We divide the observed states of the intersection into inner states (vehicles that stop before the stop line) and outer states (vehicles on the road that are heading toward the stop line). Therefore, the state vector of the intersection point can be expressed as follows.

$$s_t = w_{in} \cdot \left[V_1^{in}(t), V_2^{in}(t), \ldots, V_l^{in}(t)\right] + w_{out} \cdot \left[V_1^{out}(t), V_2^{out}(t), \ldots, V_l^{out}(t)\right] \tag{5.1}$$

where w_{in} and w_{out} denote the weight coefficients of the inner state and outer state, respectively. This approach is designed to ensure that agents prioritize inner states due to their higher weight, relegating the lower-priority outer states to a secondary consideration. Taking the environment state in Figure 4 as an example, the state is represented as a set of vectors of the number of vehicles in each lane within

the intersection. The objective of this paper is to demonstrate that a simple representation of the state and uniform approach for information interaction among the elements can be effective.

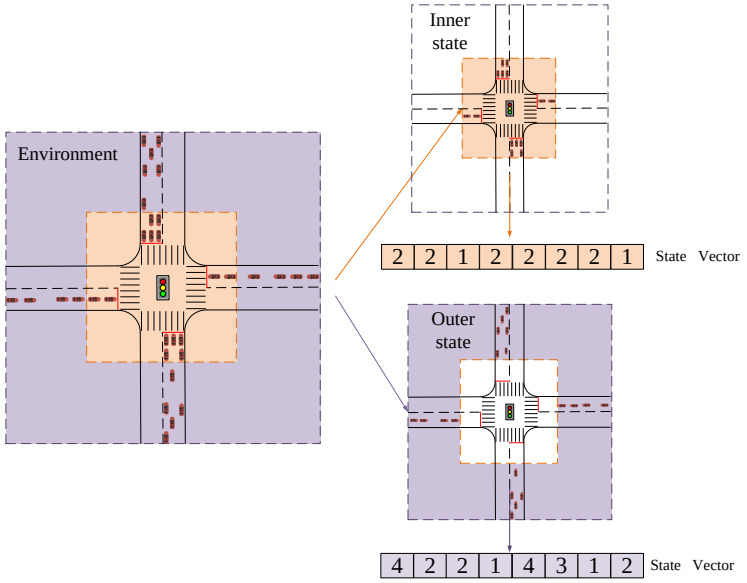

Figure 4. An example of the vector extracted from the intersection.

5.2. *Reward function*

To maintain uniformity between the state and the reward, ensuring that the agent can accurately assess the quality of its action choices through the feedback provided by the rewards, this paper constructs the reward function using the negative sum of queue lengths in the state representation. We propose reward functions obtained from the inner state at time step $t + 1$. The vehicle cumulative at time step reward function for lane l at time step t is defined as,

$$r(t) = -\sum_{l=1}^{L} V_l^{In}(t+1) \tag{5.2}$$

where V_l^{In} represents the number of vehicles queued in front of the stop line. In the above formula, the negative sign of the reward means that the objective of the agent is to minimize the vehicles queuing at intersections. This enables agents to better specify the optimal strategy using rewards.

5.3. *Action space*

In this paper, we have reported a design methodology that employs a weighted-formula approach for the cohesive update of states and actions. This method ensures consistency between states and actions, enabling an agent to dynamically adjust future action choices using the formula when perceiving real-time changes in environmental states. This approach significantly enhances the direct interaction between the agent and its environment. This means that the duration of signal phases will be determined according to the positions and movements of vehicles within the intersection. The action space A of the method proposed in this paper is defined as $A = \{a_{P_1}, a_{P_2} \cdots, a_{P_N}\}$, where P_N is the

number of phases belonging to the intersection. Meanwhile, for each $n \in \{1, 2, \cdots, N\}$, $a_{P_n} \in \{0, 0.25, 0.5, \cdots, 1.0\}$ represents the proportion of the optimal cycle time allocated to the green light duration of the different phases (P_1: provide the right of way for northbound and southbound straight, and right-turning vehicles; P_2: provide the right of way for eastbound and westbound vehicles; P_3: provide the right of way for northbound and southbound left-turning vehicles). Thus, the formula for calculating the specific content of the action space is defined as follows,

$$GT_{P_n} = (a_{p_n} \cdot GT_{\max}^{P_n}) \cdot (\frac{\sum\limits_{l \in P_n} V_l^{In} + w_{out} \cdot \sum\limits_{l \in P_n} V_l^{out}}{\sum\limits_{l \in P_n} V_l^{In} + \sum\limits_{l \in P_n} V_l^{out}}) \tag{5.3}$$

where $GT_{\max}^{P_n}$ represents the maximum green light duration of phase P_n. In order to efficiently alleviate traffic congestion at the intersection, agents will give higher attention to the inner state. Therefore, the $0 < w_{out} < 1$ will be a reasonable configuration.

5.4. Agent architecture

In our approach, we utilize the DDQN architecture as the network structure for our agent. The neural network architecture consists of three convolution layers with the activation function applied. Afterward, the output is flattened and passed through two multi-layer perceptrons to generate action neurons. Within the constructed double Q-network, we have two networks. The prediction network is responsible for predicting Q-values of actions, while the target network is utilized in updating the prediction network, as indicated in Eq (3.4) of the paper. Both networks include an input layer to receive the tensor of state of size L, where L is the number of lanes at the intersection, holding L information for L lanes (e.g., number of vehicles in queue). The output layer is the number of neurons for the action. In the learning phase, the agent receives the state s_t feedback from the environment at each time step t and action a_t is executed according to adaptive $\varepsilon - greedy$ action selection policy as follows,

$$a_t = \begin{cases} random \quad action, \quad otherwise \\ \arg\max\limits_{a_t \in A} Q(s_t, a_t; \theta), \quad if \ \zeta > \varepsilon^n \end{cases} \tag{5.4}$$

where ζ is a random number between (0,1), ε^n (ε to the n) is a value that is dynamically updated according to the number of iterations. In this paper, the agent uses tuple (s_t, a_t, r_t, s_{t+1}) to update the model weight θ of the main network. Target network is updated once each η of the predict network by copying the weights of the predict network.

In conventional DDQN, the number of layers in the network can hinder information transmission, leading to information attenuation. To address this issue, we propose a residual connections structure. This approach introduces cross-layer connections in the network, where the input is added to the output. By utilizing this method of information transfer, we can effectively address the over-estimation problem commonly encountered in traditional DDQN algorithms. Furthermore, this approach improves the convergence and stability of the algorithm.

Finally, the pseudo-code of our proposed DDQN-based USAR method is summarized in Algorithm 1.

Algorithm 1: DDQN-based USAR method

Input : Initialize θ and θ^{-}, ER Memory M, action space A, Greedy coefficient ε, network update rate χ and traffic flow data;

Output: The optimal plan for the duration of green traffic light phase;

Use residual block structure between input and output.;
for $n = 1$ **to** N **do**
    Observe initial state s_0 and take initial action a_0;
    for $t = 1$ **to** T **do**
        Observe state s_t, select action a_t according to ε-greedy policy;
        Feedback next state s_{t+1} and get r_t from s_{t+1};
        Store experience (s_t, a_t, r_t, s_{t+1}) in M;
        if *experience fills M* **then**
            Sample random experiences from M;
            Replace the oldest experience with the current one;
            Calculate y_t^{DDQN};
            Update the agent weights θ;
            if $step\%\chi = 0$ **then**
                Copy θ weights to θ^{-} weights;
            end
        end
        Agent chooses a_{t+1} based on s_{t+1};
        Start a new time step(step = step + 1);
    end
end
if *Model training finished* **then**
    Validating with the validation dataset;
end

6. Simulation experiment and discussion

In this segment, we commence by delving into the TSC challenge under investigation, and the conjectures that were assessed through experimental methodology. Details on the proposed USAR-based method will then be elucidated, along with the configuration of our simulation tests that were utilized to corroborate the postulated assumptions. The parameters included in these configurations span across the structure of the intersection environment, the traffic volume data, and the hyper-parameters manipulated during the training phase. Consequently, we will present a comprehensive analysis of the performance metrics derived from diverse methodologies, using data harvested from experimental trials. These metrics will be juxtaposed with other reference methods which encompass both conventional techniques and DRL strategies. The intention of this comparative analysis is to showcase the robustness and proficiency of our proposed USAR-based method.

6.1. Simulation settings

Within this section, our first course of action involves an introduction to the TSC issue and the hypotheses that have been put to the test through experimental means. Following this, we will furnish details concerning our proposed USAR-based method, as well as the blueprint of our simulation experiments that have been deployed for hypothesis validation purposes. Aspects encompassed by these configurations range from the infrastructural design of the intersection environment to the data regarding traffic influx, as well as the variety of hyper-parameters that underwent fine-tuning throughout the training stage. Subsequently, we will expound upon and scrutinize the results derived from a range of methodologies using empirical data. We will place these findings in comparison with other benchmark techniques, encompassing both traditional methods and those leveraging DRL. The objective of this juxtaposition is to affirm the superior efficacy of our proposed USAR-based method.

6.1.1. Intersection environment structure

Our study delves into the analysis of a quadrilateral intersection layout comprising North, South, East, and West directions. Each road at this crossroad has three lanes, with the EW lane occurring uniformly in all cardinal directions. Each lane stretches 320 meters in length and possesses identical priority and width criteria. The leftmost lane is strictly reserved for left-turn traffic, whereas the middle lane accommodates both straight and right turns. The rightmost lane is exclusively dedicated to right turns. For a graphic elucidation of the intersection's intricate structure, kindly refer to Figure 1.

6.1.2. Traffic flow input

In the quest to simulate intersection traffic conditions with accuracy, it is vital to emulate the realistic traffic environment. The traffic flow in this paper begins with minimal vehicular density, which gradually escalates to reach the crest, prior to its decline marking the completion of the traffic simulation. Commonly used probabilistic distributions for traffic flow encompass the burr distribution, poisson distribution, and normal distribution [30, 31]. Choosing the normal distribution in this experiment effectively depicts the cyclical behavior of the traffic flow. In order to simulate the inflow of vehicles under different traffic demands, three scenarios were designed to represent low, medium, and high traffic conditions, respectively. In each phase, the traffic flow of each lane follows a normal distribution with varying means and variances, as detailed in Table 2.

Table 2. Three different input traffic flows obeying normal distributions.

	Low traffic flow		Medium traffic flow		High traffic flow	
	Mean value	Variance	Mean value	Variance	Mean value	Variance
Phase 1	550	150	850	150	1150	150
Phase 2	450	100	750	100	1050	100
Phase 3	350	50	650	50	950	50

6.1.3. Parametric and training settings

After conducting multiple runs of the experiment and fine-tuning the algorithm parameters, we have established the following parameters for the experiment: the number of training episodes N is

set to 500. When the memory bank is full, the primary network parameters are updated every 65 iterations, where each experience is represented as a tuple (s_t, a_t, r_t, s_{t+1}). For the two networks in the DDQN algorithm, the learning rate is set to 0.0001. In $\varepsilon-$ greedy action selection policy, we set the action selection probability ε=0.99. Table 3 summarizes various parameters used with their associated values.

Table 3. Values used for training parameters.

Parameter	Description	Value
N	Number of training iterations	500
t	Simulation time step	2500
max $\_size$	ER Memory maximum size	50
χ	Neural network weight update frequency	65
LR	Learning rate	0.0001
γ	Discount factor	0.9
ε	Action selection probability	0.99
w_{out}	Outer state weight value	0.9
w_{in}	Inner state weight value	1.0
$GT_{\max}^{P_1}$	Maximum green duration of the phase P_1	60
$GT_{\max}^{P_2}$	Maximum green duration of the phase P_2	50
$GT_{\max}^{P_3}$	Maximum green duration of the phase P_3	40
$GT_{\min}$	Minimum green duration of all the phases	10

6.2. Comparison approach

The setting of the signal period length plays a crucial role in TSC methods. We compared our proposed method with the following four methods to verify its effectiveness.

FTSC (Signal control based on fixed-timing): [8] The fixed-time approach is one of the most commonly used traditional TSC methods. This method involves setting a fixed duration for each phase, fixed cycle length, and a fixed order of phases. In this paper, the phase sequence of FTSC is arranged from phase 1 to 3, with durations of 40, 30, and 20 seconds for each phase.

HAS (Hybrid action space): [26] In this document, the first subspace is defined as the collection of three phases, while the second subspace is defined by the duration of the green-light phase. Consequently, the action space is described as $\{\{1\ 2\ 3\} \cup \{[\ GT_{\min}, GT_{\max}]\}\}$.

ATSC (Adaptive traffic signal control): [32] This method utilizes vehicle perception of the environmental state to dynamically adjust or ensure reasonable signal coordination control parameters. Specifically, it involves setting the duration of the signals for each phase based on the inflow of traffic volume.

CSR (Consistant states and rewards): [25] This approach defines states and rewards in a consistent and straightforward manner. Meanwhile, the agent executes the action by following the sequence of phases P_1, P_2, and P_3, adhering to the green light duration for each phase.

To validate the effectiveness of the USAR algorithm on evacuation traffic flow, this paper evaluates the performance of the proposed method by comparing it with several typical and state-of-the-art experimental results. The previously mentioned HAS algorithm is a design method with the highest convergence speed during training, CSR is a method that unifies state and reward consistently in a

formulaic approach, while FTSC and ATSC represent the most classic design methods in the field of TSC. By selecting these four algorithms as benchmarks for comparison, the fast convergence and training stability of the proposed USAR are verified from multiple perspectives.

6.3. Results and discussion

To ensure the comparability of the training outcomes, all intelligent TSC algorithms utilized a consistent reward function design. Noteworthy is the fact that the USAR algorithm employs an action space design based on Markov elements, whereas the HAS and CSR algorithms use different action space design schemes. The objective of this comparative experimental setup is to validate the effectiveness of the innovative action space design method proposed in this paper. The total training consisted of 500 iterations for all algorithms, with each iteration spanning a simulation duration of 2500 seconds. The value of the reward function is the negative sum of queue vehicles across lanes.

6.3.1. Comparison of training results

Figure 5 displays the training curves for the three intelligent TSC algorithms. The solid lines represent the average reward value from multiple experiments, while the shaded areas indicate the standard deviation of the results. It is observed that once the USAR algorithm stabilizes, its performance is significantly superior to the other algorithms, achieving higher reward values. The USAR algorithm's maximum reward value is approximately -40 with a standard deviation of about 10, while the peak values for other algorithms all fall below -70 with a comparable standard deviation. Despite some fluctuation in reward values, there is a steady growth trend that eventually stabilizes within a small standard deviation range, indicating that USAR has better convergence speed and stability compared to the alternative algorithms. Additionally, while the reward values for all algorithms are similar in the early stages of training, the USAR algorithm demonstrates better performance in the later stages. Both CSR and HAS exhibit larger fluctuations throughout the training period, sometimes exceeding USAR in initial rewards, but ultimately underperforming relative to USAR. This suggests that the action space designs of HAS and CSR are prone to instability and perform less effectively in managing congestion at intersections.

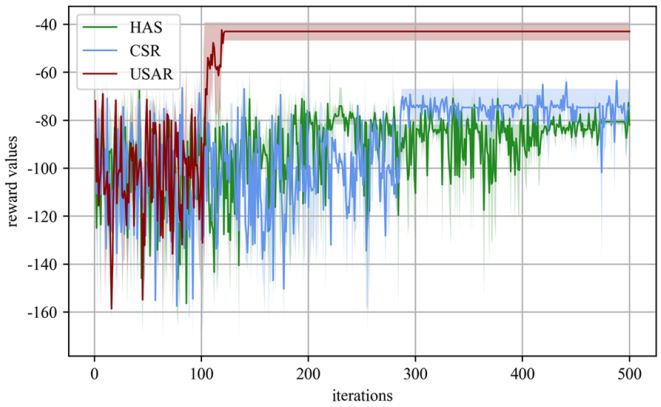

Figure 5. The reward convergence curves of different methods under medium traffic flow.

6.3.2. Comparison of evaluation results

The simulation experiment results displayed in Figure 6 through 9, encompassing four different metrics, indicate that the USAR algorithm is more effective at adapting to dynamically changing traffic flow conditions that follow a normal distribution compared to other TSC algorithms. What follows is a detailed analysis of the test outcomes:

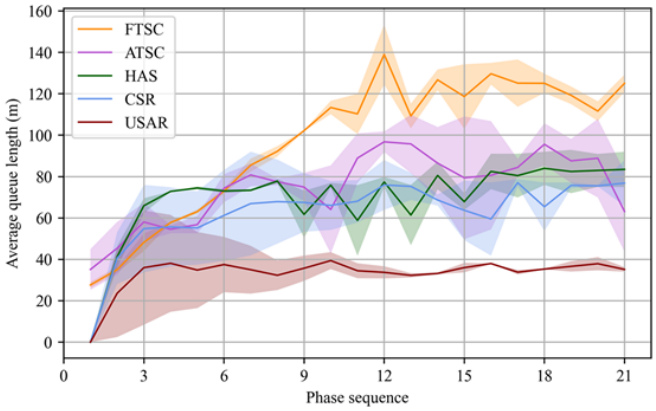

Figure 6. Experimental results of average queue length under medium traffic flow.

Average queue: It is clear from Figure 6 that the USAR algorithm outperforms the other methods. At the fourth phase sequence, its maximum value is 60 m. In contrast, the HAS algorithm has a maximum value of about 90 m at 16 phase sequences and the CSR algorithm has a maximum value of about 90 m at the seventh phase sequence. It can be observed that the result curves of each algorithm gradually increase in the early stage (the first 12 phase sequences), and then reach the peak and gradually decrease in different time periods. These convergence trends indicate that the signal control strategy of each algorithm is different; the USAR algorithm can effectively reduce the average queue length, while other algorithms can reduce the effective queue length to different degrees in the later stage. This shows that the USAR algorithm converges faster and has higher stability than other algorithms. This is because the USAR algorithm adopts a design method based on unity, which combines the state, action, and reward as Markov elements, so as to achieve fast convergence of the model. In addition, in the process of convolutional neural network capturing input features and output Q values, USAR algorithm uses the design of residual connection, which effectively improves the stability of the algorithm.

Maximum queue: Figure 7 clearly demonstrates that the USAR algorithm outperforms other algorithms in terms of average maximum queue length, achieving an average maximum of just 80 m at the 10th phase. In contrast, the HAS algorithm reaches a maximum of approximately 120 m at the seventh phase, the CSR algorithm peaks at about 125 m at the fourth phase, while the ATSC and FTSC algorithms reach maximum values near 200 and 225 m, respectively. The better performance of the ATSC and FTSC algorithms can be attributed to the lower mean input traffic flow used in the experiment, which allowed more room for optimization with traditional methods. During the initial stages of the experiment, we can observe a gradual increase in average queue lengths for all algorithms, followed by peaks and a steady decline at different points in time. These trends reflect the

diverse signal control strategies adopted by the different algorithms. The USAR algorithm proves effective in reducing average queue lengths, while other algorithms reduce queuing to varying extents later in the experiment. This indicates that the USAR algorithm achieves convergence more rapidly and with greater stability compared to others. The superiority of the USAR algorithm is attributed to its unified design approach that combines state, action, and rewards as elements of a Markov decision process, resulting in swift convergence of the model. Furthermore, the USAR algorithm employs a convolutional neural network with residual connections in capturing input features and outputting predicted values, significantly enhancing the algorithm's stability.

Average speed: As illustrated in Figure 8, the average driving speed of all vehicles on the lane achieved by the USAR algorithm surpasses that of the comparative algorithms, with its results' convergence curve clearly indicating superior performance. In the early stages of the experiment, the USAR algorithm performs similarly to others, reaching the highest driving speed before significant traffic congestion sets in. However, in later stages, as congested vehicles become more difficult to clear, the average driving speeds for all algorithms decline. The graph shows a continuous drop in speed for the HAS algorithm post-peak, struggling to recover. The FTSC and ATSC algorithms face challenges in promptly restoring smooth traffic flow during severe congestion on certain lanes, resulting in an overall reduction of driving speed. While both the USAR and CSR algorithms manage to enhance the overall driving speed in the latter part of the experiment, USAR achieves a quicker return to high-speed driving.

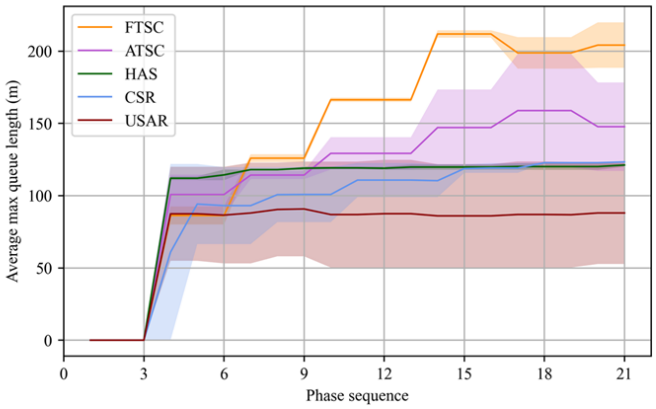

Figure 7. Evaluation results of average maximum queue length under medium traffic flow.

Average occupancy: As depicted in Figure 9, the simulation experiment showcases the capability of various algorithms in managing lane occupancy. Among the algorithms compared, the FTSC algorithm exhibited the highest lane occupancy rate, peaking at approximately 42%, and later decreased to around 37%. In contrast, the USAR algorithm proposed in this study demonstrated a significant improvement; starting from the seventh traffic signal cycle, the peak occupancy rate consistently declined to 22%. This suggests the USAR algorithm effectively ensures that no more than a quarter of the lanes within an intersection are occupied, thereby improving vehicular throughput. The area represented by the shaded region indicates that the USAR algorithm has the best stability, especially in the latter stages of the experiment, where fluctuations are minimal. Although

the HAS and CSR algorithms adopted the same reward function design as the USAR algorithm, their reliance on a traditional action space unrelated to states might have prevented optimal outcomes. Meanwhile, despite sustaining a relatively stable lane occupancy rate provided by the HAS, CSR, and FTSC algorithms, the overall level of congestion persisted at a high degree without any noticeable hints of abatement.

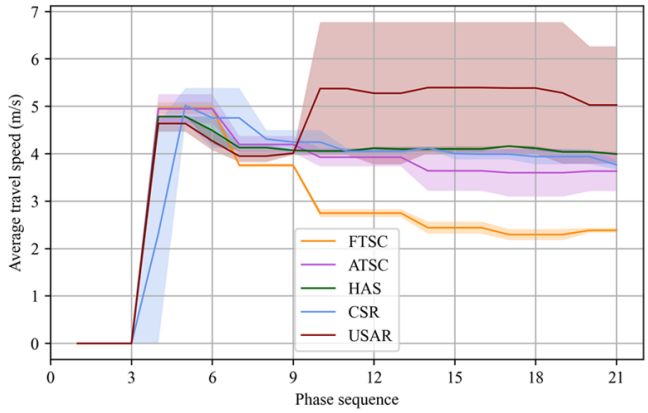

Figure 8. Evaluation results of average travel speed under medium traffic flow.

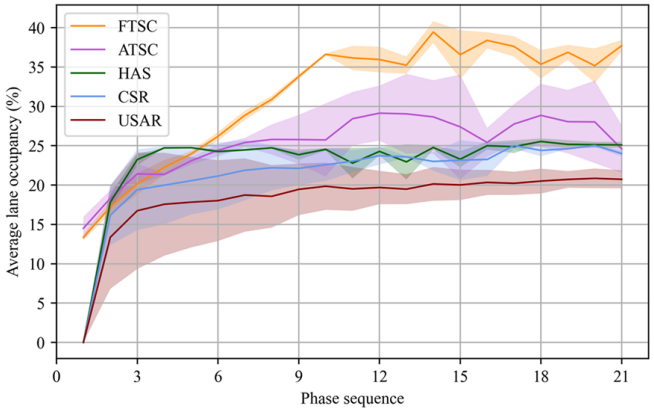

Figure 9. Evaluation results of average lane occupancy under medium traffic flow.

6.3.3. Traffic demand study

To evaluate the adaptability of the proposed method under various traffic demand scenarios, this study designed experiments using multiple types of traffic flows that conform to a normal distribution. Among low, medium, and high traffic flow conditions, three optimization algorithms were compared. As illustrated in Table 4, the experimental results show that the proposed method is superior to the other two methods in each evaluation index. Particularly under high traffic flow conditions, the proposed method USAR significantly improves the adaptability and effectively reduces key indicators such as queue length and lane occupancy rates.

Table 4. Performance comparison of different methods under different input traffic flows.

Evaluation index	HAS			CSR			USAR		
	Low	Medium	High	Low	Medium	High	Low	Medium	High
Ave. Queue	54.30	68.14	79.44	41.33	55.10	63.47	35.30	39.53	51.20
Max. Queue	72.83	100.16	100.19	69.15	92.14	97.42	50.71	77.93	90.33
Ave. Occupancy	20.72	21.53	22.27	17.49	19.98	21.42	15.10	18.27	19.31
Ave. Speed	4.35	3.58	3.19	4.98	3.54	3.05	5.96	4.27	4.10

6.3.4. State representation study

Our work employs a 1-D array model to enhance the efficiency of training the TSC model. In addition, we compared the effects of using 1-D array versus 2-D matrix for state representation. As shown in Table 5, compared with the 1-D array approach, the performance of the 2-D matrix method was improved by 5.93–11.3% in terms of the four metrics (e.g., average queue length, maximum queue length, average occupancy, and speed), but the average training time of the 2-D matrix method was also increased by 10.96%.

Table 5. Comparison of experimental results of different state representation methods.

	Ave. Queue	Max. Queue	Ave. Speed	Ave. Occupancy	Ave. Training time(s)
1-D array	58.86	102.65	3.54	22.44	26321
2-D matrix	52.21	92.93	3.75	20.77	29206
Improvement(%)	11.30	9.47	5.93	7.44	−10.96

6.3.5. State composition study

In the proposed method, queuing vehicles (inner state) and moving vehicles (outer state) are integrated into a comprehensive state representation framework, with a clear distinction made between the priority weights of internal and external states on agent behavior. Table 6 demonstrates the significant advantages of considering both queuing vehicles and moving vehicles as a unified state over the traditional approach of only using queuing vehicles as the state representation. The results indicate that, compared to conventional state design methods, our approach exhibits exceptional performance across a wide array of evaluation metrics.

Table 6. Comparison of experimental results of different methods with different states composition.

Evaluation index	Queue length-based method			Proposed method		
	Low	Medium	High	Low	Medium	High
Ave. Queue length	42.11	53.45	56.35	35.30	39.53	51.20
Max. Queue length	58.09	87.36	92.88	50.71	77.93	90.33
Ave. Lane occupancy	15.20	18.32	19.83	15.10	18.27	19.31
Ave. Speed	5.55	4.18	4.05	5.96	4.27	4.10

6.3.6. Ablation study

As shown in Table 7, ablation experiments were conducted on the USAR design approach and compared with the traditional DDQN algorithm. The results from the table indicate that incorporating the USAR design into the DDQN algorithm can effectively reduce queue lengths at intersections, while also having a positive impact on driving speed and lane occupancy rates. Notably, when using the negative sum of queue lengths as the reward function, the average queue length across multiple experiments is consistent with the negative value of the reward. This demonstrates that the USAR-based design methodology significantly enhances algorithm performance. As depicted in Figure 10, incorporating residual connections into the DDQN algorithm greatly accelerates the convergence speed and stability, whereas the traditional DDQN algorithm shows much greater volatility and only exhibits signs of full convergence in the later stages.

Table 7. Average evaluation results across different algorithms.

Evaluation index	DDQN	DDQN+USAR	Improvement(%)	Proposed	Improvement(%)
Ave. Queue	79.21	65.39	17.44	57.43	27.50
Max. Queue	116.88	109.76	6.09	105.45	9.78
Ave. Speed	3.67	3.77	2.72	4.01	9.26
Max. Speed	4.10	4.30	4.88	4.48	9.27
Ave. Occupancy	25.38	24.29	4.29	23.68	6.70
Ave. Reward	−81.97	−66.08	19.39	−55.63	32.13
Max. Reward	−69.5	−54.89	21.02	−46.78	32.69

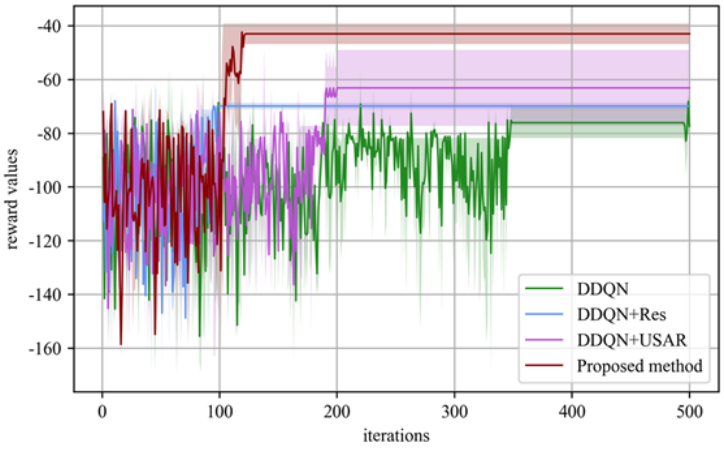

Figure 10. The ablation experiment results for the two structures of the proposed method.

7. Conclusions and future work

For the intelligent TSC problem, a USAR design method based on the DDQN algorithm framework is proposed. The proposed USAR algorithm defines the spatial distribution of vehicles at

intersections as the state and uses a formula with weighted coefficients to unify the state and action, enabling the agent to synchronize action selection with the perceived state from the environment. Meanwhile, the vehicle spatial distributions used in the state representation are used as the reward function, allowing the agent to adjust action selection strategies based on the reward values provided by the environment. The goal is to maintain the consistency between the action, state, and reward elements in the Markov process, significantly enhancing the interaction between the agent and the environment. Within the framework of DDQN, a residual connection is used to strengthen the feature correlation between input and output data, and the USAR design method mentioned earlier is jointly trained with the entire algorithm. The simulation experiments conducted on single intersections under different traffic flow show that the USAR algorithm outperforms existing single-intersection algorithms in multiple indicators.

In future work, we will focus on the following improvements: 1) Introducing additional elements (such as travel speed, lane occupancy, etc.) to define the representation of state and reward functions, further enhancing the performance of the TSC algorithm in feature representation and information interaction for signal control strategies. 2) Subsequent research will delve into multi-agent signal control in multi-intersection networks, which will examine the cooperative capabilities between multiple agents within multiple intersections, making the research more practically relevant.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This work was supported in part by Zhejiang Provincial Natural Science Foundation of China under Grant No. LTGS23F030002; by the Jiaxing Public Welfare Research Program No.2023AY11034; by the National Natural Science Foundation of China under Grant No. 61603154; and by the Open Research Project of the State Key Laboratory of Industrial Control Technology, Zhejiang University, China (No. ICT2022B52).

Conflict of interest

The authors declare there is no conflicts of interest.

References

1. B. Ye, S. Zhu, L. Li, W. Wu, Short-term traffic flow prediction at isolated intersections based on parallel multi-task learning, *Syst. Sci. Control Eng.*, **12** (2024), 1–17. https://doi.org/10.1080/21642583.2024.2316160
2. M. J. Smith, T. Iryo, R. Mounce, K. Satsukawa, D. Watling, Zero-queue traffic control, using green-times and prices together, *Transp. Res. Part C: Emerging Technol.*, **138** (2022), 103630. https://doi.org/10.1016/j.trc.2022.103630

3. B. Ye, W. Wu, L. Li, W. Mao, A hierarchical model predictive control approach for signal splits optimization in large-scale urban road networks, *IEEE Trans. Intell. Transp. Syst.*, **17** (2016), 2182–2192. https://doi.org/10.1109/TITS.2016.2517079
4. H. Wang, J. Zhu, B. Gu, Model-based deep reinforcement learning with traffic inference for traffic signal control, *Appl. Sci.*, **13** (2023), 4010. https://doi.org/10.3390/app13064010
5. B. Ye, W. Wu, K. Ruan, L. Li, T. Chen, H. Gao, et al., A survey of model predictive control methods for traffic signal control, *IEEE/CAA J. Autom. Sin.*, **6** (2019), 623–640. https://doi.org/10.1109/JAS.2019.1911471
6. B. B. Elallid, N. Benamar, A. S. Hafid, T. Rachidi, N. Mrani, A comprehensive survey on the application of deep and reinforcement learning approaches in autonomous driving, *J. King Saud Univ.-Comput. Inf. Sci.*, **34** (2022), 7366–7390. https://doi.org/10.1016/j.jksuci.2022.03.013
7. B. Ye, W. Wu, W. Mao, A two-way arterial signal coordination method with queueing process considered, *IEEE Trans. Intell. Transp. Syst.*, **16** (2015), 3440–3452. https://doi.org/10.1109/TITS.2015.2461493
8. X. Li, Webster sequences, apportionment problems, and just-in-time sequencing, *Discrete Appl. Math.*, **306** (2022), 52–69. https://doi.org/10.1016/j.dam.2021.09.020
9. T. Thunig, R. Scheffler, M. Strehler, K. Nagel, Optimization and simulation of fixed-time traffic signal control in real-world applications, *Proc. Comput. Sci.*, **151** (2019), 826–833. https://doi.org/10.1016/j.procs.2019.04.113
10. C. Yu, W. Ma, X. Yang, A time-slot based signal scheme model for fixed-time control at isolated intersections, *Transp. Res. Part B: Methodol.*, **140** (2020), 176–192. https://doi.org/10.1016/j.trb.2020.08.004
11. A. J. Calle-Laguna, J. Du, H. A. Rakha, Computing optimum traffic signal cycle length considering vehicle delay and fuel consumption, *Transp. Res. Interdiscip. Perspect.*, **3** (2019), 100021. http://doi.org/10.1016/j.trip.2019.100021
12. M. Noaeen, A. Naik, L. Goodman, J. Crebo, T. Abrar, Z. S. H. Abad, et al., Reinforcement learning in urban network traffic signal control: A systematic literature review, *Expert Syst. Appl.*, **199** (2022), 116830. https://doi.org/10.1016/j.eswa.2022.116830
13. R. Bokade, X. Jin, C. Amato, Multi-agent reinforcement learning based on representational communication for large-scale traffic signal control, *IEEE Access*, **11** (2023), 47646–47658. https://doi.org/10.1109/ACCESS.2023.3275883
14. A. A. A. Alkhatib, K. A. Maria, S. AlZu'bi, E. A. Maria, Smart traffic scheduling for crowded cities road networks, *Egypt. Inf. J.*, **23** (2022), 163–176. https://doi.org/10.1016/j.eij.2022.10.002
15. M. R. T. Fuad, E. O. Fernandez, F. Mukhlish, A. Putri, H. Y. Sutarto, Y. A. Hidayat, et al., Adaptive deep Q-network algorithm with exponential reward mechanism for traffic control in urban intersection networks, *Sustainability*, **14** (2022), 14590. https://doi.org/10.3390/su142114590
16. S. Choi, D. Lee, S. Kim, S. Tak, Framework for connected and automated bus rapid transit with sectionalized speed guidance based on deep reinforcement learning: Field test in sejong city, *Transp. Res. Part C: Emerging Technol.*, **148** (2023), 104049. https://doi.org/10.1016/j.trc.2023.104049

17. D. He, J. Kim, H. Shi, B. Ruan, Autonomous anomaly detection on traffic flow time series with reinforcement learning, *Transp. Res. Part C: Emerging Technol.*, **150** (2023), 104089. https://doi.org/10.1016/j.trc.2023.104089

18. D. Li, F. Zhu, T. Chen, Y. D. Wong, C. Zhu, J. Wu, COOR-PLT: A hierarchical control model for coordinating adaptive platoons of connected and autonomous vehicles at signal-free intersections based on deep reinforcement learning, *Transp. Res. Part C: Emerging Technol.*, **146** (2023), 103933, https://doi.org/10.1016/j.trc.2022.103933

19. I. Tunc, M. T. Soylemez, Fuzzy logic and deep Q learning based control for traffic lights, *Alexandria Eng. J.*, **67** (2023), 343–359. https://doi.org/10.1016/j.aej.2022.12.028

20. M. Gregurić, K. Kušić, E. Ivanjko, Impact of Deep Reinforcement Learning on Variable Speed Limit strategies in connected vehicles environments, *Eng. Appl. Artif. Intell.*, **112** (2022), 104850. https://doi.org/10.1016/j.engappai.2022.104850

21. B. Liu, Z. Ding, A distributed deep reinforcement learning method for traffic light control, *Neurocomputing*, **490** (2022), 390–399. https://doi.org/10.1016/j.neucom.2021.11.106

22. T. A. Haddad, D. Hedjazi, S. Aouag, A deep reinforcement learning-based cooperative approach for multi-intersection traffic signal control, *Eng. Appl. Artif. Intell.*, **114** (2022), 105019. https://doi.org/10.1016/j.engappai.2022.105019

23. S. M. A. B. A. Islam, A. Hajbabaie, H. A. A. Aziz, A real-time network-level traffic signal control methodology with partial connected vehicle information, *Transp. Res. Part C: Emerging Technol.*, **121** (2020), 102830. https://doi.org/10.1016/j.trc.2020.102830

24. A. Jaleel, M. A. Hassan, T. Mahmood, M. U. Ghani, A. U. Rehman, Reducing congestion in an intelligent traffic system with collaborative and adaptive signaling on the edge, *IEEE Access*, **8** (2020), 205396–205410. https://doi.org/10.1109/ACCESS.2020.3037348

25. S. Bouktif, A. Cheniki, A. Ouni, H. El-Sayed, Deep reinforcement learning for traffic signal control with consistent state and reward design approach, *Knowl.-Based Syst.*, **267** (2023), 110440, https://doi.org/10.1016/j.knosys.2023.110440

26. S. Bouktif, A. Cheniki, A. Ouni, Traffic signal control using hybrid action space deep reinforcement learning, *Sensors*, **21** (2021), 2302. https://doi.org/10.3390/s21072302

27. B. Ye, P. Wu, W. Wu, L. Li, Y. Zhu, B. Chen, Q-learning based traffic signal control method for an isolated intersection, in *2022 China Automation Congress (CAC)*, (2022), 6063–6068, https://doi.org/10.1109/CAC57257.2022.10054839

28. Y. Gong, M. Abdel-Aty, Q. Cai, M. S. Rahman, Decentralized network level adaptive signal control by multi-agent deep reinforcement learning, *Transp. Res. Interdiscip. Perspect.*, **1** (2019), 100020. https://doi.org/10.1016/j.trip.2019.100020

29. J. Gu, Y. Fang, Z. Sheng, P. Wen, Double deep Q-network with a dual-agent for traffic signal control, *Appl. Sci.*, **10** (2020), 1622. https://doi.org/10.3390/app10051622

30. W. Ma, L. Wan, C. Yu, L. Zou, J. Zheng, Multi-objective optimization of traffic signals based on vehicle trajectory data at isolated intersections, *Transp. Res. Part C: Emerging Technol.*, **120** (2020), 102821. https://doi.org/10.1016/j.trc.2020.102821

31. A. Lopez, W. Jin, M. A. Al Faruque, Security analysis for fixed-time traffic control systems, *Transp. Res. Part B: Methodol.*, **139** (2020), 473–495. https://doi.org/10.1016/j.trb.2020.07.002

32. W. Lin, H. Wei, Cyber-physical models for distributed CAV data intelligence in support of self-organized adaptive traffic signal coordination control, *Expert Syst. Appl.*, **224** (2023), 120035. https://doi.org/10.1016/j.eswa.2023.120035

AIMS Press