A Deep Adaptive Traffic Signal Controller With Long-Term Planning Horizon and Spatial-Temporal State Definition Under Dynamic Traffic Fluctuations

SHURONG LI, CHONG WEI, XUEDONG YAN, LU MA, DEQI CHEN, AND YING WANG

MOT Key Laboratory of Transport Industry of Big Data Application Technologies for Comprehensive Transport, School of Traffic and Transportation, Beijing Jiaotong University, Beijing 100044, China

Corresponding authors: Chong Wei (chwei@bjtu.edu.cn) and Lu Ma (lma@bjtu.edu.cn)

This work was supported in part by the Fundamental Research Funds for the Central Universities under Grant 2019JBM040, and in part by the National Natural Science Foundation of China under Grant 71971017, Grant 91746201, and Grant 71621001.

ABSTRACT This study proposes a new adaptive traffic signal control scheme to effectively manage dynamically fluctuating traffic flows through intersections. A spatial-temporal representation of the traffic state at an intersection has been designed to efficiently identify traffic patterns from complex intersection environments, and a deep neural network (long short-term memory network, LSTM) is used to determine look-ahead signal control decisions based on the estimated long-term feedback from a given traffic state. The actor-critic algorithm, one of the reinforcement learning-based algorithms, is adopted to obtain the essential parameters of the LSTM deep neural network through multiple interactions between a simulated environment and the corresponding adaptive traffic signal controller. A realistic model environment comprising a 24-hour time-varying traffic demand including rush hour and non-rush hour situations served as the basis for traffic generation in the numerical experiments to confirm the effectiveness of the proposed scheme. The results of these experiments show that, compared to an optimized fixed time plan (Synchro), the proposed scheme can reduce waiting times at intersections by an astounding 50% with consequential benefits of reducing fuel consumptions, emissions, queue lengths, and vehicle delays whilst increasing mean speeds.

INDEX TERMS Adaptive traffic signal control, reinforcement learning, spatial-temporal traffic state representation, actor-critic, LSTM, 24-hour dynamic traffic fluctuations.

I. INTRODUCTION

It is generally accepted that the efficient management of traffic flows to reduce travel delay, especially through intersections, is an essential objective in traffic management. Attempting to achieve this objective is Adaptive Traffic Control which has an advantage in that it can take real-time and stochastic traffic demand into consideration and provide traffic light control decisions based on a wide range of algorithmic designs including dynamic programming, fuzzy logic and reinforcement learning as reported in various studies [1]–[5]. However, based on an examination of these previous studies, it is clear that the design of an adaptive traffic control algorithm is plagued by three crucial problems outlined as follows:

Firstly, an effective method is needed to represent, as accurately as practically possible, the time-varying traffic flow situation (usually referred to as the traffic state) at a target intersection to include additional operating information pertinent to the particular intersection [5]. In such a method, in order to maintain accuracy or rather preserve as much information as possible, it would seemingly be convenient to use an approach based on microscopic measures (such as the position and velocity) of individual vehicles rather than macroscopic measures (such as traffic flow). However, as an intersection is a complex system involving the positions and velocity profiles of multiple vehicles, such an approach can be adversely impacted by high computation costs (the curse of dimensionality), which in previous studies were avoided.

The associate editor coordinating the review of this manuscript and approving it for publication was Sabah Mohammed.
through the use of aggregate (macroscopic) measures such as flow rate, flow speed and vehicle queue length instead of the individual (microscopic) measures of vehicle positions and velocities to represent the traffic state [3], [6], [7]. Although these aggregate measures simplify traffic state representation, specific vehicular information at a target intersection is inevitably lost. On the other hand, even if individual vehicle positions and velocities are considered, delay information is lacking for each vehicle as well as for the intersection system resulting in the inability to effectively minimize the total delay for each traffic light control decision.

Secondly, although in recent years a considerable number of studies have employed macroscopic traffic flows in their underlying traffic flow models, there remains a problem in that these models cannot precisely reflect the actual traffic flow characteristics of a target intersection which negatively impacts evaluating the real-time performance of control policies in adaptive traffic signal control algorithms [8]–[11]. As already mentioned in the statement of the first problem, there is good reason for using macroscopic traffic flow models in that their computational costs are relatively low, so they can easily be implemented in adaptive control algorithms. This computational benefit was a contributory factor which effectively led to the implementation of some well-known adaptive traffic control systems, such as SCOOT [12], SCATS [13] and COP [14]. However, there still remains the problem that macroscopic traffic flow models cannot precisely reflect the traffic flow characteristics of a target intersection.

The third problem relates to the modelling of traffic demand at an intersection, which normally involves using vehicle arrival rates as the basis for such modelling. However, previous studies mainly assume that the arrival rate is constant throughout the day [3], [7], [15], [16], treating rush hour and non-rush hour traffic as the same homogeneous conditions. Clearly, this assumption can lead to the poor performance of traffic signal control schemes, especially during rush hours. Furthermore, although the Annual Average Daily Traffic (AADT), obtained from historical observation data, is used to provide aggregate information for traffic signal control, we found that few algorithms can exactly take into account the 24-hour time-varying traffic demands, arguably needed to further enhance the design of traffic signal control schemes.

Addressing these three problems, the research underpinning this study establishes a fundamental decision-making framework which uses a Reinforcement Learning (RL) approach supported by a deep neural network to implement an adaptive traffic control algorithm [17], [18]. The main reason behind this approach is the recognition that RL has recently contributed to effective decision-making in several other areas such as gaming and robotics, as well as in traffic control [19]–[24]. Furthermore, previous studies have confirmed that RL can effectively work with microscopic traffic flow models for adaptive traffic control and can make look-ahead control decisions for an intersection system [4], [25].

This study focuses on reducing the total delay of vehicles through the design of a signal controller that can identify both the spatial and temporal patterns in real-time traffic based on microscopic information so as to reduce information loss. To reflect spatial traffic situations as realistically as possible, individual vehicle delays are defined as the basic element of the traffic state, and an intersection is partitioned into cells that represent individual delays. Moreover, to capture the temporal traffic dynamics, we employ a series of spatial observations to enhance the representation of the traffic state, which is then used as input into a neural network to determine the control decisions at different time intervals. Note that the type of neural network used in this study is the LSTM network because it is especially suited to time sequence problem modelling. This type of network, therefore, provides the essential basis for dealing with complex traffic states as represented by microscopic vehicular and other operational information without suffering from the curse of dimensionality [26], [27]. The proposed adaptive signal controller enables control decisions based on the guidance of the trained neural network after it has learnt an optimal control policy through multiple trial-and-error interactions between the controller and the intersection environment.

Reinforcement learning (RL) is employed to determine the parameters of the LSTM network under a microscopic traffic simulation environment given dynamic traffic demand scenarios [28]. It should be noted that in previous studies, traffic demand scenarios for training procedures were usually generated assuming a constant vehicle arrival rate [7], [15], [16]. In this study, however, vehicle arrivals are generated following a Poisson distribution for a 24-hour time-varying traffic demand curve obtained from historical data. This method allows the training procedure to take into account daily traffic dynamics and reflects the stochastic nature of traffic demand.

To ensure the convergence of the RL algorithm, we employed an actor-critic strategy in the RL model to optimize the parameters of the LSTM network. Moreover, we used multistep bootstrapping technique and clipped surrogate objective technique to enhance the algorithm efficiency and robustness. We finally provide a framework for the RL that is specially designed for adaptive traffic control. The contribution of this paper can be concluded as follows:

Firstly, in previous RL studies, the consideration of individual vehicular delay information for processing by traffic controllers was usually ignored. Addressing this omission, we propose a novel traffic state definition to identify both the spatial and temporal patterns using microscopic traffic delay information. This method reduces information loss and provides individual vehicular delay information for each traffic light control decision.

Secondly, we have designed a RL algorithm framework to determine the parameters of the LSTM network in a microscopic simulation environment. This framework can guarantee the convergence of the learning process under complex traffic states.
Thirdly, the RL algorithm executes the learning process based on a 24-hour time-varying traffic demand, which incorporates both rush hour and non-rush hour situations. This enables realistic observations of commuting traffic in practice and can provide substantial practical benefits in the implementation of adaptive traffic control.

Following on from this introduction, the remainder of this paper is arranged as follows: Section II presents a literature review. Section III provides background information for the traffic signal control scheme, involving delay, cost, and state. In Section IV, we describe the optimization algorithm, an actor-critic algorithm combined with a multistep bootstrapping technique and a clipped surrogate objective technique. In Section V, we describe the setup of the numerical experiment and present the results to demonstrate the performance of the proposed method. In Section VI, we conclude our work on the proposed method and provide some suggestions for future work.

II. LITERATURE REVIEW
There are currently two basic types of traffic signal control: fixed-time signal control and adaptive signal control. In the case of fixed-time control, the controller utilizes historical traffic data to determine signal timing off-line [29]. Fixed-time control has been widely used in several well-known signal timing systems, such as the TRANSYT [30], SYNCHRO [31] and MAXBAND [32] systems. Fixed-time control performs stably if the traffic demand follows a fixed pattern; however, it cannot respond to stochastic traffic conditions, especially in situations where there is a sudden buildup of traffic.

Adaptive signal control utilizes real-time data to determine an optimal signal timing to maximize a defined objective function and in recent decades it has gradually gained popularity due to its adaptability and flexibility. The controller managing adaptive signal control can be classified according to the type of control it provides, namely: responsive signal control, online optimization control, or revising frequency control [33]. Each of these control classifications is discussed as follows:

A. RESPONSIVE SIGNAL CONTROL
In the case of responsive signal control, each signal in a controller seeks a decision to extend the current green phase or not, based on the upstream actuated traffic demand. A typical signal control system is the modernized optimized vehicle actuation (MOVA) system [34]. However, a disadvantage of this system is that it fails to optimize globally because the control decision only considers the traffic demand in the current green direction whilst ignoring all other directions.

B. ONLINE OPTIMIZATION CONTROL
The online optimization algorithm utilizes model predictive control [35], traffic flow model control [10], [36] and Petri nets model control [37] to make control decisions considering detected and predicted future traffic. Typical existing systems using online optimization include the SCOOT [12] and SCATS [13] systems, which have shown significant improvements in the performance of traffic signal control. However, almost all of these systems were developed using macroscopic traffic flow models, which implies the loss of detailed information about individual vehicle movements at an intersection and, therefore, results in poor performance regarding control decisions.

C. REVISING FREQUENCY CONTROL
The revising frequency control approach uses a rolling horizon and starts an optimization every few seconds to maximize an objective function over the planning period. The main feature of the revising frequency control approach is that the traffic signal timing is optimized at a rather fast pace, and the resolution can be as short as 0.5 seconds [38]. This method makes it possible to control traffic signals effectively in many practical applications, such as the PRODYNE [39], OPAC [40], RHODES [41] and COP [14] systems. The techniques used to solve the rolling horizon problems include dynamic planning and reinforcement learning (RL), with the latter technique gaining popularity in recent research because of its computational feasibility and its adaptability in complex problems [38]. Extensive research, therefore, has been conducted in traffic signal control using RL [5]. In one such research, as a pilot study, a multiagent traffic signal control scheme with a model-based RL was developed to minimize the overall waiting time of vehicles. This study confirmed the effectiveness of its RL-based adaptive signal control algorithm through comparing the performance of the RL controller and non-adaptive traffic signal controllers. The author employed Q-learning to minimize the number of waiting vehicles for an isolated intersection with aggregate state information such as the queue length in its four approaches [42]. The advantage of RL in the revising frequency control method was further confirmed through the comparison of RL with function approximation and dynamic planning [38]. RL can be divided into three categories: value-based RL, policy-based RL and actor-critic RL. The author has shown that actor-critic RL, a combination of value-based RL and policy-based RL, outperforms the other two algorithms and has many benefits in terms of robustness, training speed and the generalization of new traffic scenarios [7]. Recent studies have also combined deep neural networks with RL to implement traffic control and have shown that such a combination can significantly improve the robustness and generalization ability of RL [7], [43]. In addition, a deep neural network enables RL to handle higher dimensions of traffic state representations more efficiently including the complex and stochastic characteristics of real-world traffic systems [25].

III. TRAFFIC SIGNAL CONTROL SCHEME
For a typical intersection (Section III-A), the traffic efficiency is largely influenced by the traffic signal control scheme and thus the traffic signal controller plays a critical role in building a safe, efficient, and environmental driving environment.
in city traffic conditions. In this study, we adopt vehicle delay to evaluate traffic efficiency and construct objective function by minimizing the total delay $J$ of surrounding vehicles at the intersection (as introduced in Section III-B).

Fig. 1 shows the proposed control scheme. We divide the time horizon into several discrete time intervals, and each time interval is indexed by $t$ with a duration of $\Delta t$. For each time interval $t$, the signal control is fulfilled by three stages: perception, decision-making, and execution. In the perception stage: the controller first detects the positions and speeds from surrounding vehicles information at time interval $t$ through several types of smart sensors, such as millimeter-wave traffic radar and computer vision-based traffic monitors; then the controller estimates the state $s_t$ using the method in Section III-C to extract representative information without losing too much information. In the decision-making stage, the controller gives action $a_t$ between two choices: to extend the current phase ($a_t = 0$) or change into the next phase ($a_t = 1$) based on the observed state $s_t$; the basic logic of this is that a trained neural network function can predict the optimal action $a_t$ using state $s_t$ as input, where its internal parameters are obtained using RL through multiple interactions between the controller and simulation environment (as introduced in Section IV). Finally, in the execution stage, the traffic light will carry out the planned action $a_t$ for $\Delta t$ seconds, observe the control feedback cost $j_t$ and begin the next iteration of time interval $t + 1$.

**A. CELL-BASED INTERSECTION LAYOUT**

Fig. 2 shows a typical single intersection that has four-direction legs. Each leg consists of multiple approaching links and departure links. We use $\Omega$ to denote the approaching link set. Furthermore, we divide each approaching link into multiple cells to collect useful information about the vehicle movements, where the cells are labeled with $0, 1, 2, \ldots, i, \ldots, N_i$. Since the minimum space headway of two successive vehicles is 7.5 m, each cell span 7.5 meters when dividing cells. This can benefit the precise state representation and reduce information loss through a high-resolution space division method.

**B. DELAY AND COST FUNCTION**

When approaching an intersection, vehicles might slow down as a result of catching up to the vehicles in front or they may have to stop because of a red light. We define the vehicle delay $d_t(k)$ as the amount of extra time for vehicle $k \in U_t$ to at time interval $t$:

$$d_t(k) = \Delta t(1 - \frac{v_t(k)}{v_{free}})$$  \hspace{1cm} (1)$$

where $\Delta t$ is the duration of time interval $t$, $v_t(k)$ is the average speed of vehicle $k$ during time interval $t$, $v_{free}$ is a constant that indicates the speed of a vehicle passing through the intersection under the free-flow conditions and $U_t$ is the set of the vehicles located on the approaching links of the intersection at the beginning of the time interval $t$. Note that for vehicle $k$, the average speed $v_t(k)$ is associated with phase decision $a_t$, and the current state $s_t$. If the corresponding phase is green, the vehicle might travel with a high speed, however, if the corresponding phase is red, the vehicle should slow down (low speed) and stop.
For each time interval $t$, the controller’s performance can be evaluated by cost $j_t$:

$$j_t = \sum_{k \in U_i} d_t(k)$$  \hspace{1cm} (2)

where cost $j_t$ considers the total delay for surrounding vehicles $U_i$.

Furthermore, based on the cost of each time interval, the total cost function of the controller can be derived as:

$$\min_{a_t} J = \sum_{t=0}^{T} j_t$$  \hspace{1cm} (3)

where $T$ is the total number of time intervals in the planning horizon. Minimizing cost $J$ is consistent with the fact that our controller aims at improving the traffic efficiency by manipulation phase extension or not over the planning horizon $[0, 1, 2, \ldots, T]$.

### 3. STATE DEFINITION

The traffic controller makes action decisions based on the representative state of the target intersection. We propose a method to collect the multidimensional information to represent the traffic state to reduce the information loss caused by partial observability. In previous works, the traffic state definition is usually simply represented by the aggregate information, such as the average queue length or waiting time of the vehicles located in the intersection. However, such representation ignores individual differences and spatial information. This study proposes a method that can take into account the travel time delay for each vehicle in the intersection and maintain the dimensions of the input as a constant even if the number of vehicles at an intersection is time-dependent.

Since the size of $U_i$ is time-dependent, it is not proper to use $d_t(k)$ directly as the representation of the state. We define delay $D_{t}$ for cell $i$ as follows:

$$D_{t}^{i} = \left\{ \begin{array}{ll}
\frac{\sum_{k \in U_{t}^{i}} d_{t}(k)}{|U_{t}^{i}|} & \text{if } |U_{t}^{i}| \neq 0, \\
0 & \text{if } |U_{t}^{i}| = 0.
\end{array} \right.$$  \hspace{1cm} (4)

where $U_{t}^{i}$ denotes the set of vehicles that are located at the cell $i$ at time $t$. If $|U_{t}^{i}| = 0$, it means that no vehicle has been checked in cell $i$ at time $t$ and therefore $D_{t}^{i}$ is equal to 0. In the case of $|U_{t}^{i}| \neq 0$, we use the mean delay value of the vehicles as the delay value of the cell $i$ at time $t$. We further use $D_t$ to denote the delay information for the intersection as follows:

$$D_t = [D_1^t, D_2^t, \ldots, D_i^t, \ldots, D_N^t]$$  \hspace{1cm} (5)

where $N_i$ is the total number of cells at the approaching links for the target intersection. $D_i^t$ can be obtained by observing the speeds of the vehicles in cell $i$ during time interval $t$.

Our proposed method does not require the tracking of the trajectories of each vehicle at an intersection. Therefore, the proposed state representation method is adaptive and easily implemented for several types of smart sensors, such as millimeter-wave traffic radar and computer vision-based traffic monitors.

Furthermore, in order to restrain the phenomenon that the phase keeps fast flip, we use $c_t$ to denote the cumulative repetition number of the current phase until the end of the last time interval $t - 1$ which can be expressed as:

$$c_t = \left\{ \begin{array}{ll}
c_{t-1} + 1 & \text{if } a_{t-1} = 0, \\
0 & \text{if } a_{t-1} = 1.
\end{array} \right.$$  \hspace{1cm} (6)

$c_t$ is 0 when the signal controller switches into the next phase ($a_{t-1} = 1$) and $c_t = c_{t-1} + 1$ when the signal controller decides to remain in the current phase ($a_{t-1} = 0$) at the last time interval $t - 1$.

In conclusion, we introduce the spatial observation $x_t$ and state $s_t$ definitions to capture the spatial and temporal dynamics in a complex traffic environment.

$$x_t = [\phi_t, c_t, D_t]$$

$$s_t = [x_{t-\xi+1}, x_{t-\xi+2}, \ldots, x_t]$$  \hspace{1cm} (7)

where $\phi_t \in \{0, 1, 2, 3\}$ is referred to the traffic signal phase at time $t$ (see Fig. 3). $\xi$ is the number of observed timesteps.

An example is provided to illustrate the evolution of the proposed state representation method. As shown in Fig. 3, the controller executes the signal phase $\phi_t$, sequentially from phase 0 to phase 3. The value of $\xi$ is set to 10 for illustration purposes. We assume that time interval $t-1$ begins with phase $\phi_{t-1} = 1$ and phase repetition number $c_{t-1} = 10$. After $\Delta t$ seconds, the controller is required to make a decision based on state $s_t$ at the beginning of time interval $t$. Because of action $a_{t-1} = 0$, the controller obtains $x_t$ where $\phi_t = 1$, $c_t = 11$ and $D_t = [3.86, 3.12, \ldots, 0.00]$. Then, based on the evaluation of the state $s_t = [x_{t-9}, x_{t-8}, \ldots, x_t]$, the controller selects an action $a_t = 0$, receives feedback $j_t$ and begins the next iteration of time interval $t + 1$. Subsequently, the controller selects to change the phase ($a_{t+1} = 1$) at the beginning of time interval $t + 1$ based on $s_{t+1}$ where $\phi_{t+1} = \phi_t = 1$ and $c_{t+1} = c_t + 1 = 12$ because of the extending phase decision of time interval $t$ ($a_t = 0$). After $\Delta t_{\text{yellow}} + \Delta t$ seconds, state $s_{t+2}$ for time interval $t + 2$ is obtained where $\phi_{t+2}$ and $c_{t+1}$ take values of 2 and 0 as a result of the phase change decision ($a_{t+1} = 1$). Notice that the yellow phase $\Delta t_{\text{yellow}}$ cannot be ignored considering safety issues for phase changes decisions ($a_t = 1$).

### IV. REINFORCEMENT LEARNING FOR TRAFFIC SIGNAL CONTROL

The decision-making stage is achieved through a trained neural network which can determine the best action $a_t$ for state $s_t$. Two neural networks are constructed to achieve the decision-making: a critic network $\hat{V}(s_t; \theta)$ to predict the expected cumulative cost and an actor network $\pi(a_t|s_t; \theta)$ to calculate the optimal action directly (Section IV-A). In terms of the network structure, we adopt the LSTM network to build the critic network and actor network (Section IV-B) to capture the temporal dynamics. Then, as the parameters are stochastically given initially, we adopt an actor-critic framework, one
type of the RL-based algorithms, to calculate the parameters $w$ and $\theta$ in order to reduce the estimated error for the critic network and reduce the future cost for the actor network (Section IV-C). The main logic is to obtain a set of optimal parameters $w$ and $\theta$ gradually to guide the process of decision making through training data, which is determined through multiple interactions between the decision-making agent and simulated environment. Furthermore, we adopt two techniques to enhance the algorithm efficiency and robustness, which are Multistep bootstrapping (Section IV-D) and Clipped surrogate objective technique (Section IV-E). Finally, the overall algorithm is given in Section IV-F to illustrate the training details.

A. BASIC COMPONENTS

RL, a type of machine learning technique, enables the controller to obtain optimal parameters for decision-making agent through trail and error interactions between the controller and simulated environment [18]. Two core concepts in RL are the actor and critic: the actor can guide the choice of action $a_t$ based on state $s_t$ at each time interval $t$ for the controller, and the critic can predict the cumulative cost of state $s_t$ to estimate the long term performance of current state.

The actor can be defined as a conditional probability, $\pi(s, a) = P(a_t = a|s_t = s)$, which can map system state $s$ to an action probability distribution over an action set $a_t \in \{0, 1\}$. The critic can estimate the average expected cumulative cost since state $s_t$ over the planning horizon $[t, t+1, \ldots, \tau, \ldots, T]$:

$$V(s_t) = \mathbb{E}\left[\sum_{t=\tau}^{T} j_t | s_t \right]$$  (8)

Furthermore, based on the Bellman equation, we can obtain $V(s_t)$ iteratively by using the cost $j_t$ and the next state value $V(s_{t+1})$.

$$V(s_t) = \mathbb{E}[j_t + V(s_{t+1})|s_t]$$  (9)

Note that the true values of $V(s_t)$ and $\pi(s, a)$ are difficult to observe in practice. Usually, we adopt function approximations $\hat{V}(s_t; w)$ and $\hat{\pi}(a_t|s_t; \theta)$ to estimate $V(s_t)$ and $\pi(a_t|s_t)$ where $w$ and $\theta$ are the parameters of the function approximations $\hat{V}(s_t; w)$ and $\hat{\pi}(a_t|s_t; \theta)$ (shown in Fig. 4).

FIGURE 3. Example of state evolution.

B. NEURAL NETWORK STRUCTURE USING THE LSTM

To capture the temporal dynamics of the sequences, a long short-term memory (LSTM) network is employed to construct the function approximation.

This study adopts the many to one structure to map a sequence vector $[x_1, x_2, \ldots, x_{\tau}]$ to vector $y_{\tau}$ [44], as shown in Fig. 5. The LSTM network can be referred as $\Theta^L : \mathbb{R}^{\tau \cdot M} \rightarrow \mathbb{R}^{1 \cdot N}$, where $\tau$ is the length of the time sequence, $M$ is the length of one vector and $N$ is the length of the output. In addition, fully connected layer can be referred as $\Theta^C : \mathbb{R}^M \rightarrow \mathbb{R}^N$, where $M$ denotes the length of the input vector and $N$ denotes the length of the output vector.

The function approximation architecture adopted in this study is shown in Fig. 5. For each time interval, $t$, the state
\( s_t = [x_{t-\xi+1}, x_{t-\xi+2}, \ldots, x_T] \) is obtained through stacking a series of spatial observations \( x_{t-\xi+1}, x_{t-\xi+2}, \ldots, x_T \) of different time intervals, where \( \xi \) is the length of the time series and the length of each \( x_t \) is \( N_t + 2 \) (see Section IV-C). Using state \( s_t \) as the input, the input matrix size is \( \xi \cdot (N_t + 2) \). First, we employ a fully connected layer \( \Theta^f \) with 64 neurons and rectifier nonlinear activation functions to compress observation \( x_t \in [t-\xi+1, t-\xi+2, \ldots, t] \). Then, we pass the stacked output to the LSTM layer, \( \Theta^l \), where the LSTM layer is composed of 64 units and unrolled for a \( \xi \) step input. Finally, the fully connected layer \( \Theta^C \) is used as the linear output layer with no activation function for the critic and softmax activation function for the actor, where the number of neurons is 1 and 2 respectively, corresponding to the number of outputs in the critic and actor. During the training process, the algorithm uses the Adam optimizer as the gradient descent algorithm with a learning rate of 0.0002 for the critic and 0.0001 for the actor. The critic and actor net use the same structure except for the output layer. The definitions of the critic and actor net are given as follows:

\[
\hat{V}(s_t; w) = \Theta^C \Theta^L([\Theta^C(x_{t-\xi+1}), \Theta^C(x_{t-\xi+2}), \ldots, \Theta^C(x_T)]) = \Theta_w(s_t; w) \tag{10}
\]

\[
\hat{\pi}(a_t|s_t; \theta) = \Theta^C \Theta^L([\Theta^C(x_{t-\xi+1}), \Theta^C(x_{t-\xi+2}), \ldots, \Theta^C(x_T)]) = \Theta_\theta(a_t|s_t; \theta) \tag{11}
\]

**C. PARAMETER OPTIMIZATION USING ACTOR-CRITIC ALGORITHM**

The actor-critic algorithm, one of the RL-based algorithms, is adopted in this study to determine the controller’s internal parameters \( (w \text{ and } \theta) \) in the interactive process between the traffic signal controller and the simulation environment. The reasons for adopting the actor-critic algorithm are that the convergency results in critic-only algorithms tend to be biased and that the convergency speeds in actor-based algorithms are usually rather slow. On the other hand, the actor-critic algorithm effectively provides an appealing trade-off between optimality and convergency speeds.

Two basic components are the actor \( \hat{\pi}(a_t|s_t; \theta) \) and critic \( \hat{V}(s_t; w) \). During the training process, the actor is employed to determine an action \( \hat{\pi}(a_t|s_t; \theta) \) for the current state \( s_t \); then the controller executes the command \( a_t \), receives the feedback \( j_t \) and collects the training sample \( (s_t, a_t, j_t, s_{t+1}) \); based on the collected sample, the critic can estimate the long-term performance \( \hat{V}(s_t; w) \) and get involved in the parameter optimizing process to help the actor and the critic achieve better performance in the future [45]. The actor-critic algorithm optimizes internal parameters \( \theta \) and \( w \) as follows:

1. The actor chooses action \( a_t \sim \hat{\pi}(a_t|s_t; \theta) \) at the beginning of time interval \( t \);
2. The controller receives the feedback \( j_t \) and \( s_{t+1} \) at the end of time interval \( t \);
3. The controller collects the training sample \( (s_t, a_t, j_t, s_{t+1}) \);
4. The critic evaluates the temporal difference (TD) error according to the sample;
5. The algorithm optimizes the actor and the critic’s internal parameters \( \theta \) and \( w \) respectively, according to the TD error.

In the following, we will show the details for the parameter optimization process.

For the critic, the parameter \( w \) is optimized to decrease the difference between the approximation \( \hat{V}(s_t; w) \) and true value \( V(s_t) \) by applying a gradient descent to the mean-square loss function \( L(w) \).

\[
\min_w L(w) = \mathbb{E}[(j_t + \hat{V}(s_{t+1}; w) - \hat{V}(s_t; w))^2] = \mathbb{E}[\delta_t^2] \tag{12}
\]

where \( j_t + \hat{V}(s_{t+1}; w) \) is an approximate estimate for the true value \( V(s_t) \) following Bellman Equation. \( \delta_t = j_t + \hat{V}(s_{t+1}; w) - \hat{V}(s_t; w) \) is temporal difference error (TD error) to express the difference between approximation and true value. The critic optimizing can be regarded as a regression problem to build the mapping between state \( s_t \) and value \( V(s_t) \) using the training sample \( (s_t, a_t, j_t, s_{t+1}) \), where the input label is \( s_t \) and the output label is \( j_t + \hat{V}(s_{t+1}; w) \). The batch
stochastic gradient descent technique (BGSD) is employed to obtain an optimized parameter $\omega$: $w \leftarrow w + \alpha_w \nabla_w V(s_t; w)$, where $\alpha_w$ is the learning rate of the critic.

For the actor, parameter $\theta$ is adjusted to minimize the total cost $J$ by applying gradient ascent to the cross-entropy loss $L(\theta)$:

$$
\min_{\theta} L(\theta) = \mathbb{E} [\log \pi(a_t|s_t; \theta) \delta_t] \tag{13}
$$

Eq. 13 can optimize $\theta$ using the BGSD technique following $\theta \leftarrow \theta + \alpha_\theta \delta_t \nabla_\theta \log \pi(a_t|s_t; \theta)$, where $\alpha_\theta$ is the learning rate in the actor. For further theoretical explanation, readers can refer to the work by Sutton [18].

The whole procedure of the actor-critic algorithm is summarized in Algorithm 1.

### Algorithm 1 Actor Critic Algorithm

**Input:** a differentiable state value function, $\hat{V}(s_t; w)$ a differentiable policy function, $\hat{\pi}(a_t|s_t; \theta)$

**Output:** updated parameters $w$ and $\theta$

1. for $day = 0, 1, 2, \ldots, M$
do
2. for $t = 0, 1, 2, \ldots, T$
do
3. // Training sample collecting
4. Sample action $a_t \sim \hat{\pi}(a_t|s_t; \theta)$
5. Execute the action $a_t$, receive the feedback $j_t$, and obtain the next state, $s_{t+1}$
6. // Parameter optimizing
7. Compute TD error $\delta_t$ according to the training sample $(s_t, a_t, j_t, s_{t+1})$
8. Optimize parameters $w$ and $\theta$ using Eqs. 12 and 13
9. end for
10. end for
11. return Parameters $w$ and $\theta$

### D. MULTISTEP BOOTSTRAPPING TECHNIQUE

Since the batch size is too small (only one training sample) for parameter optimizing process, we are inspired by the multistep bootstrapping technique to increase batch size when optimizing parameters. This can further improve the convergence speed and algorithm stability in the above algorithm (Algorithm 1).

The original method updates parameters $w$ and $\theta$ using only one sample $(s_t, a_t, j_t, s_{t+1})$ at the end of each time interval. Inspired by the multistep bootstrapping technique, this study updates parameters $w$ and $\theta$ at the beginning of the time interval $t = qn$ for $q = 0, 1, 2, \ldots, Q$, where $n$ is a prespecified number (step) of time intervals. Consider the environment is at time interval $t = qn$, the algorithm progresses as follows: the controller selects actions $a_t \sim \pi(a_t|s_t; \theta_{qn})$ during the intervals $t = qn, qn+1, \ldots, qn+n-1$ and executes parameter optimizing until the end of the time interval $t = qn+n-1$. The RL algorithm collects the samples $(s_t, a_t, j_t, s_{t+1})$ from $t = qn$ to $qn + n - 1$ and saves these samples into training buffer $B_{(qn, (q+1)n)}$:

$$
B_{(qn, (q+1)n)} = \left[ \begin{array}{c}
\begin{bmatrix}
(s_{qn}, a_{qn}, j_{qn}, s_{qn+1}) \\
(s_{qn+1}, a_{qn+1}, j_{qn+1}, s_{qn+2}) \\
\vdots \\
(s_{qn+n-1}, a_{qn+n-1}, j_{qn+n-1}, s_{(q+1)n})
\end{bmatrix}
\end{array} \right].
$$

Therefore, we can obtain a batch of $n$-step-TD errors at $t = qn + n - 1$ based on $B_{(qn, (q+1)n)}$ using the $n$-step return:

$$
\delta_t = -\hat{V}(s_t; w) + \sum_{i=n}^{n-1} j_t + \hat{V}(s_{(q+1)n}; w) \quad i \in [qn, qn + 1, \ldots, qn + n - 1] \tag{14}
$$

Using Eq. 14, we can obtain $\delta_{qn}$, $\delta_{qn+1}$, ..., $\delta_{qn+n-1}$ as:

$$
\delta_{qn} = -\hat{V}(s_{qn}; w) + j_{qn} + \ldots + j_{qn+n-1} + \hat{V}(s_{(q+1)n}; w)
$$

$$
\delta_{qn+1} = -\hat{V}(s_{qn+1}; w) + j_{qn+1} + \ldots + j_{qn+n-1} + \hat{V}(s_{(q+1)n}; w)
$$

$$
\vdots
$$

$$
\delta_{qn+n-1} = -\hat{V}(s_{qn+n-1}; w) + j_{qn+n-1} + \hat{V}(s_{(q+1)n}; w)
$$

Then, we obtain the expanded training buffer $\tilde{B}_{(qn, (q+1)n)}$:

$$
\tilde{B}_{(qn, (q+1)n)} = \left[ \begin{array}{c}
\begin{bmatrix}
(s_{qn}, a_{qn}, j_{qn}, \delta_{qn}, s_{qn+1}) \\
(s_{qn+1}, a_{qn+1}, j_{qn+1}, \delta_{qn+1}, s_{qn+2}) \\
\vdots \\
(s_{qn+n-1}, a_{qn+n-1}, j_{qn+n-1}, \delta_{qn+n-1}, s_{(q+1)n})
\end{bmatrix}
\end{array} \right].
$$

We then draw random samples uniformly from the expanded training buffer $\tilde{B}_{(qn, (q+1)n)}$ and adopt these samples to construct a training batch to optimize the parameters $w$ and $\theta$ using the BGSD technique based on Eqs. 12 and 13. Compared to optimizing using only a single training sample (see Algorithm 1), batch optimizing calculates a series of TD errors $[\delta_{qn}, \delta_{qn+1}, \ldots, \delta_{qn+n-1}]$ to calculate parameters $w$ and $\theta$.

The process of the multistep bootstrapping technique is visualized in Fig. 6.

### E. CLIPPED SURROGATE OBJECTIVE TECHNIQUE

Despite the Actor-Critic Algorithm serves as an effective framework to obtain optimized parameters in the RL agent, the extreme large loss (Eqs. 12 and 13) may cause the parameters to change unstably between two successive parameter optimizing processes. This may have a negative impact on the stability of algorithm and convergence speed [46]. Therefore, to alleviate the influence of this problem, this study further employs a clipped surrogate objective technique to improve algorithm robustness. The clipped surrogate objective technique uses clipped probability ratios to enhance the performance of parameter updating.

After implementation in the clipped surrogate objective technique, the actor loss function $L(\theta)$ is modified as follows:

$$
\min_{\theta} L(\theta) = \mathbb{E} \left[ \min \{\delta_t \varepsilon_t(\theta), \delta_t \text{clip}(\varepsilon_t(\theta), 1-\epsilon, 1+\epsilon)\} \right] \tag{15}
$$

where the clip ratio is defined as $\varepsilon_t(\theta) = \frac{\pi(a_t|s_t; \theta)}{\pi(\tilde{a}_t|s_t; \tilde{\theta}_{old})}$ and $\epsilon$ is the clip rate. $\theta_{old}$ is the previous value of the policy neural network parameters.
network parameter. This indicates a conservative attitude in parameter optimizing [47]. The clip function is defined as follows:

\[
\text{clip}(x, \min x, \max x) = \begin{cases} 
\min x & \text{if } x < \min x, \\
 x & \text{if } \min x < x < \max x, \\
\max x & \text{if } x > \max x.
\end{cases}
\]  

(16)

Therefore, \(\text{clip}(\varepsilon_i(\theta), 1 - \epsilon, 1 + \epsilon)\) can guarantee that the effective \(\delta_i(\theta)\) is in the interval \([1 - \epsilon, 1 + \epsilon]\), which allows the algorithm to avoid the trap of excessive parameter changes. Finally, the minimum value between \(\delta_i\varepsilon_i(\theta)\) and clipped \(\delta_i\varepsilon_i(\theta)\) is adopted in calculating the loss function of parameter \(\theta\), which can provide a pessimistic bound for the loss function \(L(\theta)\) and avoid the problem of oscillation caused by large parameter changes.

**F. OVERALL ALGORITHM**

In this section, we summarize the above content and provide an overall algorithm for the training of the controller internal parameters \(w\) and \(\theta\). The overall algorithm, shown in Algorithm 2, is an actor-critic algorithm combined with a multistep bootstrapping technique and a clipped surrogate objective technique.

In Algorithm 2, the time interval \(t\) is the minimum traffic signal control unit in the planning period with a duration of \(\Delta t\). We define the operator \(*\) to show the real-time of time interval \(t\) in the environment where the relationship of time interval \(t + 1\) and time \(t\) is \((t + 1)^* = t^* + \Delta t\) if \(a_t = 0\) or \((t + 1)^* = t^* + \Delta t_{yellow} + \Delta t\) if \(a_t = 1\). The controller consists of two parts: an actor \(\hat{\pi}(a_t | s_t; \theta)\) and a critic \(\hat{V}(s_t; w)\). Moreover, we adopt a deep neural network to construct the function \(\hat{V}(s_t; w)\) and \(\hat{\pi}(a_t | s_t; \theta)\) (see Section IV-B).

The control process is as follows: at the beginning of each time interval \(t\), the controller observes the state \(s_t\) and selects an action \(a_t\) and sends it to the traffic light; after \(\Delta t\) (for \(a_t = 0\)) or \(\Delta t_{yellow} + \Delta t\) (for \(a_t = 1\)) seconds, the controller receives the feedback \(j_t\) and observes the next state \(s_{t+1}\) to begin the next iteration. The duration of the planning period is referred to as \(\Gamma\), and the simulation of one day terminates when the actual time \(T^* \geq \Gamma\). As a common practice in RL, the state value of the terminal state \(V(s_T)\) is equal to 0. The action \(a_t\)

---

**Algorithm 2** Training Controller Internal Parameters by RL

**Input:** a differentiable state value function \(\hat{V}(s_t; w)\) a differentiable policy function \(\hat{\pi}(a_t | s_t; \theta)\)

**Output:** optimized parameters \(w\) and \(\theta\)

1: Initialization \(\theta = \theta_0, w = w_0, q = 0\)

2: for day = 0, 1, 2, ..., \(M\) do

3: \(q = 0\)

4: for \(t = 0, 1, 2, \ldots, T\) do

5: // Training sample collecting

6: Sample action \(a_t \sim \hat{\pi}(a_t | s_t; \theta)\)

7: Execute the action \(a_t\), receive the feedback \(j_t\) and obtain the next state \(s_{t+1}\)

8: Store sample \((s_t, a_t, j_t, s_{t+1})\) to the training buffer \(B_{(qn,(q+1)n)}\)

9: // Parameter optimizing

10: if \(t + 1 = (q + 1)n\) or the simulation of one day ends then

11: Compute a batch of n-step TD errors \([\delta_{qn}, \delta_{qn+1}, \ldots, \delta_{qn+n-1}]\) and construct the expanded training buffer \(B_{(qn,(q+1)n)}\) based on (14).

12: Optimize parameters \(w\) and \(\theta\) using the expanded training buffer \(B_{(qn,(q+1)n)}\) based on (12) and (15).

13: \(q = q + 1\)

14: end if

15: end for

16: end for

17: return Parameters \(w\) and \(\theta\)
is a binary variable, where the value 0 denotes remaining in the current phase and the value 1 denotes switching into the next phase.

Using a multistep bootstrapping technique, we can divide the planning period into a training sample collecting period (interaction process between traffic environment and controller) and a parameter optimizing period (the parameter optimizing process to optimize parameters $\theta$ and $w$).

1. Training sample collecting period (time duration $t = [qn, qn + 1, \ldots, qn + n - 1]$)

For each time interval $t$, the controller selects an action $a_t \sim \pi(a_t | s_t; \theta_{qn})$ and sends the command $a_t$ to the traffic light to control the movement of the vehicles; the controller then receives the feedback $j_t$ and begins with $s_{t+1}$ after $\Delta t$ or $\Delta t_{\text{yellow}} + \Delta t$ seconds. Then, the controller collects the sample $(s_t, a_t, j_t, s_{t+1})$ and stores it in the training buffer $B_{(qn, (q+1)n)}$. The controller starts at time interval $qn$ and repeats the above interaction process $n$ times until time interval $qn + n - 1$.

2. Parameter optimizing period (the end of time interval $qn + n - 1$)

If the next time interval $t + 1 = (q + 1)n$ or the controller terminates in the next time interval (when $(t + 1)^{\ast} > \Gamma$), the controller computes a batch of advantaged estimators $[\delta_{qn}, \delta_{qn+1}, \ldots, \delta_{qn+n-1}]$ and constructs the expanded the training buffer $B_{(qn, (q+1)n)}$.

Finally, the controller optimizes parameters $w$ and $\theta$ using the BGSD technique.

The traffic signal control algorithm is summarized in Algorithm 2.

V. EXPERIMENTS AND RESULTS

In this section, we provide a series of numerical examples to demonstrate the performance of the proposed traffic signal control method. The microscopic traffic simulator SUMO (Simulation of Urban Mobility) was used as the simulation environment in our experiments [48]. The vehicle agent communicates with the traffic environment through the TraCI package of SUMO. We implemented the LSTM networks using TensorFlow 1.5 to approximate the functions $\hat{V}(s_t; w)$ and $\pi(a_t | s_t; \theta)$ [49]. The tests were executed on a desktop PC with a 4.20 GHz i7-7700 CPU, 32 GB of RAM running Windows 10. The whole procedure was implemented in Python 3.5.

A. STUDY AREA

This numerical example is based on the test bed introduced by Jeffrey Glick [50]. The selected intersection is located at the intersection of Palm Drive and Arboretum Road, in Stanford CA 94305, USA. The corresponding SUMO configurations were exported from OpenStreetMap.org, as shown in Fig. 7. The given intersection has four direction legs, and each approaching and departure leg comprises two links. In each approaching link, the movement $m$ can choose to move left, through or right in accordance with their route $U$.

B. TRAFFIC DEMAND AND VEHICLE GENERATION

In this section, we introduce a vehicle arrival generation method based on the realistic traffic demands (hourly traffic volume data of different vehicular movements). In practice, it is common to collect the hourly traffic volume data (vehicles per hour, vph) for 12 movements (4 left-turn movements, 4 through movements and 4 right-turn movements), as shown in Fig. 7. The 24-hour traffic volume data for the 12 movements is used for the test, as shown by the gray points in Fig. 8. These traffic flow data were sourced from the Github project by Jeffrey Glick [50]. To describe the traffic flow time-dependent characteristics, we adopt 12 polynomial fitted functions $[\rho_0(t), \rho_1(t), \ldots, \rho_m(t), \ldots, \rho_{11}(t)]$ to approximate the time-dependent traffic demand of the

![Approaching links: 10, 11, 20, 21, 30, 31, 40, 41
Departure links: 12, 13, 22, 23, 32, 33, 42, 43
Movement:](image-url)
FIGURE 8. Traffic flow of the 12 movements.

12 traffic movements. The black curve in Fig. 8 represents the time-related characteristics of the traffic demand $\rho_m(t)$. Therefore, we can insert the vehicles into the network dynamically and stochastically to reflect the morning and evening peaks, which creates a realistic and varying flow pattern for the agent to control, as opposed to using a constant hourly demand.

To simulate a series of stochastic traffic scenarios in SUMO, we assume that the vehicles arrive upstream of the approach to the intersection following a Poisson distribution with a time-dependent arrival rate of $\lambda = \rho_m(t)$. For vehicles $k \in U^m$, they follow the direction of movement $m$, where $U^m$ denotes all the vehicles in movement $m$. Therefore, for vehicles $k \in U^m$ in movement $m$, the headway $\Delta \zeta$, of two adjacent vehicles $k + 1$ and $k$, follows the negative exponential distribution:

$$\Delta \zeta \sim \lambda e^{-\lambda \Delta \zeta} \quad \text{where } \lambda = \rho_m(\zeta(k)) \quad (17)$$

where $\Delta \zeta$ indicates the headway between two adjacent vehicles $k + 1$ and $k$. $\zeta(k)$ indicates the departing time of vehicle $k$. The arrival rate $\lambda$ is estimated using the fitted function $\rho_m(t)$ (see the black curve in Fig. 8).

C. PARAMETER SETTINGS
To train the parameters in the LSTM network using RL, we run the numerical experiment for an arbitrary figure of 200 simulated days, using the standard 86400 seconds for the duration of each day. The exploration rate drops from 1.0 to 0.0 linearly during the first 150 days. After the 150th day, the agent stops exploring and exploits the.
environment completely. The learning rates are 0.0002 and 0.0001 for critics and actors, respectively. The hyper-parameter settings are shown in Table 1.

**D. BENCHMARK GENERATED BY SYNCHRO**

We evaluate the effectiveness of the algorithm relative to the fixed-time controller generated by Synchro [31]. Synchro has been widely recognized as a practical software tool for developing a fixed time schedule for traffic signal control. Based on historical demand, the software calculated the best cycle length to be 60 seconds. To reflect fluctuations in the traffic demand, we divide the 24 hours demand into 7 subsets, and each configuration is shown in Table 2. Additionally, the yellow transition is set to 4 seconds between two different phases, which is included at the end of each phase.

**E. RESULTS ANALYSIS**

1) **CONVERGENCE VALIDATION**

The performance of the agent during the training process is shown in Fig. 9. The horizontal axis of the figure reflects the training days, ranging from 0 to 200, where a day represents a simulated day. The vertical axis reflects the average cost \( J_T \) (overall delay at the intersection during each time interval). The gray curve illustrates that the value of the objective function can increase significantly during the learning process. The magnitudes of the oscillations decreases

![Learning process over 200 days.](image.png)

**FIGURE 9.** Learning process over 200 days.

| Hyper-parameters | Meaning | Values |
|------------------|---------|--------|
| \( T \)          | duration of the whole planning period | 24-60-60 seconds |
| \( \Delta t \)    | duration of each time step | 7 seconds |
| \( \Delta t_{\text{yellow}} \) | duration of yellow transition | 4 seconds |
| \( n \)          | length of sampling period | 32 |
| \( \xi \)        | the length of time sequences in LSTM | 40 |
| \( \epsilon \)   | clip rate | 0.2 |

**TABLE 1.** Hyper-parameters setting.

| Hour | Cycle | PH0 | PH1 | PH2 | PH3 |
|------|-------|-----|-----|-----|-----|
| 0 h-3 h | 60 | 6 | 4 | 16 | 4 | 6 | 4 | 16 | 4 |
| 5 h-7 h | 60 | 6 | 4 | 16 | 4 | 5 | 4 | 17 | 4 |
| 7 h-10 h | 60 | 6 | 4 | 17 | 4 | 4 | 4 | 17 | 4 |
| 10 h-15 h | 60 | 5 | 4 | 18 | 4 | 4 | 4 | 17 | 4 |
| 15 h-18 h | 60 | 6 | 4 | 17 | 4 | 4 | 4 | 17 | 4 |
| 18 h-20 h | 60 | 6 | 4 | 18 | 4 | 4 | 4 | 16 | 4 |
| 20 h-24 h | 60 | 6 | 4 | 16 | 4 | 6 | 4 | 16 | 4 |

*PH: Phase Duration (seconds); YT: Yellow Transition Duration (seconds).*

**TABLE 2.** Synchro signal time settings.
with growing training days. The results confirm that our algorithm can learn from experience and optimize the policy gradually.

To describe the above results in detail, in Fig. 9 (b), the cost $j_t$ during one day, for different days is visualized to provide insight into the noise and performance changes as the algorithm progresses. The horizontal axis represents the simulation time for one day, while the vertical axis is the cost $j_t$ obtained per time interval. Note that we use the moving average technique to illustrate the change tendency of costs during a day. Fig. 9 (b) shows that the cost increases significantly during the morning and evening peaks of all the training days because of commuting traffic, which is consistent with the tendency of the traffic demand in Fig. 8. However, the magnitude of increased delay during the period of commuting, decreases as the training days progress as illustrated through the comparisons between day 0, 49, 99, 149 and 199. Therefore, we can conclude that the influence of the commuting traffic is significantly weakened using our algorithm.

Moreover, Fig. 9 (b) shows the convergence of the proposed RL algorithm. Initially, on training day 0 (day 0), the agent explores the environment and selects actions randomly most of the time. Therefore, the performance of the agent is poor, and the magnitude of the oscillations is very large. As the training days progress, the agent learns more from experience and selects more exploitative actions with a lower exploratory rate. Thus, the agent achieves a lower cost, lower variance and a near-optimal and stable performance to control the environment. Finally, after training day 83, the agent begins to converge to an optimal policy as indicated in Fig. 9 (a).
2) PERFORMANCE COMPARISON

We compare the RL controller’s performance to that of the fixed-time plan generated by Synchro. As the RL controller has a different cycle length with a fixed-time plan, we collect data every 5 seconds in the RL controller and use a fixed-time plan to build the same test bench. To test the effectiveness...
TABLE 3. Performance comparisons of RL controller and fixed time plan (Synchro).

| Day | FU | HC | MS | QL | WT |
|-----|----|----|----|----|----|
|     | SY | RL | SY | RL | SY | RL | SY | RL | SY | RL |
| 0   | 9.35 | 8.10 | 3.43 | 2.53 | 28.89 | 31.71 | 2.44 | 1.32 | 42.90 | 18.46 |
| 1   | 8.86 | 8.39 | 3.02 | 2.61 | 28.89 | 30.94 | 1.93 | 1.68 | 31.47 | 21.20 |
| 2   | 9.33 | 8.28 | 3.38 | 2.63 | 28.47 | 31.12 | 2.39 | 1.46 | 43.03 | 18.20 |
| 3   | 8.92 | 8.12 | 3.11 | 2.55 | 29.10 | 31.36 | 2.07 | 1.36 | 34.92 | 16.55 |
| 4   | 9.93 | 8.07 | 3.80 | 2.54 | 26.68 | 31.42 | 2.92 | 1.37 | 52.78 | 16.47 |
| 5   | 8.92 | 8.73 | 3.07 | 2.89 | 28.79 | 30.78 | 1.99 | 1.75 | 32.95 | 23.49 |
| 6   | 8.90 | 8.36 | 3.06 | 2.68 | 28.71 | 30.94 | 1.98 | 1.51 | 32.77 | 19.20 |
| 7   | 9.09 | 8.44 | 3.18 | 2.68 | 28.83 | 31.39 | 2.11 | 1.47 | 35.83 | 19.45 |
| 8   | 9.12 | 8.35 | 3.19 | 2.64 | 28.62 | 31.24 | 2.12 | 1.42 | 36.03 | 18.73 |
| 9   | 9.02 | 8.35 | 3.13 | 2.63 | 28.77 | 31.22 | 2.05 | 1.40 | 34.71 | 17.91 |

Mean  9.15  8.34  3.24  2.66  28.78  31.17  2.20  1.47  37.74  18.77

Impro (%)  8.85  17.92  8.31  33.09  50.27

a FU: Fuel (ml/s); HC: hydrocarbon emission rate (mg/s); MS: mean speed (m/s); QL: queue length (veh); WT: waiting time (s).

b SY: synchro plan; RL: reinforcement learning plan; Impro: Improvement.

TABLE 4. Performance comparisons of the RL controller and fixed-time plan (Synchro) for the different directions.

|     | FU | HC | MS | QL | WT |
|-----|----|----|----|----|----|
|     | SY | RL | SY | RL | SY | RL | SY | RL | SY | RL |
| South 6.89 | 12.01 | 2.36 | 3.44 | 28.04 | 36.98 | 1.63 | 1.32 | 25.16 | 15.80 |
| North 10.61 | 7.44 | 4.66 | 2.37 | 25.22 | 30.04 | 4.25 | 1.31 | 79.68 | 15.16 |
| West 6.69 | 6.31 | 2.34 | 2.13 | 30.18 | 28.73 | 1.37 | 1.37 | 20.81 | 15.64 |
| East 13.58 | 6.62 | 4.47 | 2.16 | 32.12 | 31.10 | 2.00 | 1.64 | 45.95 | 15.76 |
| Mean  9.39 | 8.10 | 3.43 | 2.52 | 28.89 | 31.71 | 2.46 | 1.32 | 42.90 | 16.46 |

a FU: Fuel (ml/s); HC: hydrocarbon emission rate (mg/s); MS: mean speed (m/s); QL: queue length (veh); WT: waiting time (s).
b SY: synchro plan; RL: reinforcement learning plan.

either result is consistent with the fact that the northerly traffic (flows 3,4 and 5) and east traffic (flows 9,10 and 11) occupy a large proportion of the traffic in the morning and evening peaks (see Fig. 8). The result shows that the RL controller can address the unbalanced time-space problem caused by commuting traffic. 

The delay evolution is illustrated in Fig. 11 (a), and the results show that the RL controller performs better in comparison to the Synchro plan because of the rapid and flexible changing phases. As Fig. 11 (a) shows, the delay in both controllers increases with increasing traffic demand in the morning and evening peaks, whereas the RL controller increases more slowly when compared to the fixed-time controller, especially in the morning peak. Fig. 11 (b) shows that the RL controller can increase the duration of phase 1 (time duration from 31600 to 31900) and phase 3 (time duration from 32000 to 32400) to accommodate the traffic demand in the morning peak. However, the Synchro plan is based on the fixed-time plan, which can cause an oversaturated traffic flow at the intersection, and delays using the Synchro plan can increase significantly (i.e., the time duration from 31600 to 31900). Additionally, Fig. 11 (c) shows that the RL controller outperforms the Synchro plan by giving more priority to left-turn traffic (phase 0 and phase 2) in the evening peak. We can conclude that the RL controller can outperform the fixed-time plan because of its adaptability and rapid responsiveness.

VI. CONCLUSION

In this paper, we propose a new adaptive traffic signal control scheme to produce optimized traffic control policies in order to minimize the delay of vehicles passing through intersections. The scheme employs an enhanced algorithm which uses spatial-temporal network information to define the traffic state, where individual vehicle delay is used as a basic measure rather than the aggregate measures of flow rate, flow speed and vehicle queue length as used in previous studies. The proposed method to identify traffic patterns can reduce information loss (such as vehicle delay) when characterizing high-dimensional features in the definition of traffic state. Furthermore, we adopted a deep neural network (the LSTM) to construct a decision-making agent in which its intrinsic parameters are determined through a RL framework; thus optimizing the ability of a traffic controller to decide whether to extend the current phase or switch into the next phase. Specifically, the RL framework uses an actor-critic algorithm to obtain a balance between a biased convergence result (critic-based RL algorithms) and a high variance result (actor-based RL algorithms). Additionally, we modified this algorithm with a multistep technique and a clipped surrogate objective technique to improve its performance.

In the simulations, we built experiments based on a representation of the intersection of Palm Drive and Arboretum...
Road, where simulated vehicles entered the network based on the assumption that the vehicle arrival rate follows a Poisson distribution that was derived from the 24-hour traffic flow of the prevailing time-varying daily commuting distribution. The road model reflects the instability of traffic flow during non-rush hour periods, which characterizes the flow peak in rush hours and traffic disruption events, as shown in the numerical examples for 10 random samples, compared to the optimized fixed-time plans obtained using Synchro, that the proposed method can reduce such vehicle delay times by over 50%. This significant reduction in delay times also has additional knock-on effects; fuel consumptions were reduced by over 8%, emission rates down by over 17%, vehicle queues down by over 33%, whilst mean speeds were increased by over 8%. The results, therefore, strongly indicate that the proposed scheme should be effective for traffic signal control at isolated intersections where significant traffic fluctuations are prevalent.

The proposed scheme in this study essentially focused on an isolated intersection. However, in future work the scheme could be extended to control a regional network similar to that of the OPAC and PRODYN systems. To overcome the significant computational load problems, a distributive RL framework should also be considered to speed up the training process.

### Notation

In this section, the notation is given in Table 5 to clarify the whole scheme.

### Relationship Between δW, δθ and L(θ)

\[ \Delta \theta = a_0 \delta \nabla \log \pi(a_t | s_t; \theta) \]
\[ = E[\log \nabla \pi(s, a) \delta_t] \]
\[ = \nabla_\theta L(\theta) \] (18)

Eq. 18 shows that \( \Delta \theta \) is the derivative of \( L(\theta) \), where the update gradient \( a_0 \delta \nabla \log \pi(a_t | s_t; \theta) \) is the common updating rule of the policy-based RL algorithm. In the study, the automatic differentiable software achieves max\( \theta \) by obtaining the derivative \( \Delta \theta \) to update \( \theta \). In other words, the automatic differentiation software (such as TensorFlow, Keras, PyTorch) achieves min\( \theta \) by obtaining the derivative \( \Delta \theta \).

### References

1. P. Diakaki, D. Kotsialos, and Y. Wang, “Review of road traffic control strategies,” Proc. IEEE, vol. 91, no. 12, pp. 2041–2042, Dec. 2003.
2. A. L. C. Bazzan, D. de Oliveira, and B. C. da Silva, “Learning in groups strategies,” Eng. Appl. Artif. Intell., vol. 23, no. 4, pp. 560–568, Jun. 2010, doi: 10.1016/j.engappai.2009.11.009.
3. S. Araghi, A. Khosravi, and D. Creighton, “A review on computational intelligence methods for controlling traffic signal timing,” Expert Syst. Appl., vol. 42, no. 3, pp. 1538–1550, Feb. 2015.
4. D. Zhao, Y. Dai, and Z. Zhang, “Computational intelligence in urban traffic signal control: A survey,” IEEE Trans. Man Cybern. Syst., vol. 42, no. 4, pp. 485–494, Jul. 2012.
5. K.-L.-A. Yau, J. Qadir, H. L. Khoo, M. H. Ling, and P. Komisarczuk, “A survey on reinforcement learning models and algorithms for adaptive traffic signal control,” in Autonomic Road Transport Support Systems. Springer, 2016, pp. 47–66.
6. M. Aslani, M. S. Mesgari, and M. Wiering, “Adaptive traffic signal control with actor-critic methods in a real-world traffic network with different traffic disruption events,” Transp. Res. C, Emerg. Technol., vol. 85, pp. 732–752, Dec. 2017.
7. K. Aboudolas, M. Papageorgiou, A. Kouvelas, and E. Kosmatopoulos, “A rolling-horizon quadratic-programming approach to the signal control problem in large-scale congested urban road networks,” Transp. Res. C, Emerg. Technol., vol. 18, no. 5, pp. 680–694, Oct. 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0968090X09000813

### Table 5. Variables and meanings.

| Variable | Meaning |
|----------|---------|
| \( s \) | cell index |
| \( \Omega \) | set of approaching links |
| \( N_t \) | total number of cells at the approaching links |
| \( m \) | movement index illustrated in Fig. 1 |
| \( \rho_{\text{on}}(t) \) | approximated time-dependent traffic demand function of movement \( m \) |
| \( t \) | discrete time index |
| \( T \) | total number of time intervals in the planning horizon |
| \( \Gamma \) | duration of the whole planning period and the planning horizon consists of a sequence of time intervals \( 0, 1, 2, \ldots, T \) |
| \( t^* \) | real time corresponding to time \( t \) (seconds) |
| \( \Delta t \) | duration of a time interval \( t \) (seconds) |
| \( \Delta t_{\text{yellow}} \) | duration of a yellow transition (sec) |
| \( k \) | index of a vehicle entering the intersection |
| \( v_{\text{avg}}(k) \) | average speed of vehicle \( k \) during the time interval \( t \) (m/s) |
| \( \delta_i(k) \) | delay time of vehicle \( k \) at time \( t \) (seconds) |
| \( \zeta(k) \) | arriving time of vehicle \( k \) (seconds) |
| \( U_t(i) \) | set of vehicles of cell \( i \) at time \( t \) |
| \( U_t \) | set of vehicles in the whole network at time \( t \) |
| \( U^m \) | set of vehicles of movement \( m \) |
| \( U \) | set of vehicles of all movements in the whole network |
| \( \alpha_t \) | control variable denoting the action of switching or not, \( \alpha_t \in \{0, 1\} \) |
| \( \phi_t \) | phase index, \( \phi_t \in \{0, 1, 2, 3\} \) |
| \( c_t \) | cumulative repetition number of the current phase until the end of the last time interval |
| \( \Delta L_{\text{obs}}(i) \) | delay observation of cell \( i \) at time \( t \) (seconds) |
| \( D_t \) | delay of the whole network at time \( t \) |
| \( s_t \) | state at time \( t \) |
| \( j_s \) | cost obtained at time \( t \) (seconds) |
| \( \pi \) | stochastic policy \( \pi(s, a) = P(a_t = a | s_t = s) \) |
| \( \hat{\pi} \) | actor network function with parameters \( \theta \) |
| \( B_{\text{qn}}(q+1)n \) | training buffer to store the samples from time \( qn \) to \( (q+1)n - 1 \) |
| \( V(s_t) \) | true state value, \( V(s_t) = E[\sum_{\tau=t}^{T} \gamma^\tau r(s_\tau)] \) |
| \( V(s_t; \theta) \) | critic network function with parameters \( \theta \) |
| \( \delta_t \) | temporal difference error (TD error) |
| \( n \) | length of the sampling period |
| \( \epsilon \) | clip ratio in the clipped surrogate objective technique |
| \( \epsilon \) | clip ratio in the clipped surrogate objective technique |
| \( \Theta \) | fully connected layer |
| \( \Theta_w \) | LSTM layer |
| \( \ell \) | the length of the time sequences in the LSTM network |
Y. Yin, “Robust optimal traffic signal timing,” Transp. Res. A, vol. 35, no. 2, pp. 148–164, May 2001. [Online]. Available: http://transci.

J. Ke, H. Zheng, H. Yang, and X. M. Chen, “Dynamic network traffic control,” Transp. Res. A, vol. 35, no. 8, pp. 721–744, Sep. 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S096585640000148

H. Lo, “A cell-based traffic control formulation: Strategies and benefits of on-demand ride services,” Transp. Sci., vol. 35, no. 2, pp. 148–164, May 2001. [Online]. Available: http://transci.

J. Kober, J. A. Bagnell, and J. Peters, “Reinforcement learning in robotics: architectures for urdu acoustic modeling,” Int. J. Speech Technol.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and K. Kavukcuoglu, “Deep reinforcement learning for adaptive traffic signal control and automatic vehicle
decision-making,” in Proc. 16th Int. IEEE Conf. Intell. Transp. Syst. (ITSC ), Oct. 2013, pp. 1737–1742.

J. Balka, J. J. Henry, “Design by Petri nets of an intersection signal controller,” Transp. Res. C, Emerg. Technol., vol. 4, no. 4, pp. 231–248, Aug. 1996.

J. Cai, C. K. Wong, and B. G. Heydecker, “Adaptive traffic signal control using approximate dynamic programming,” Transp. Res. C, Emerg. Technol., vol. 17, no. 5, pp. 456–474, Oct. 2009, doi: 10.1016/j.trc.2009.01.005.

J. J. Henry, “PRODYN tests and future experiments on ZELT,” in Proc. Vehicle Navigat. Inf. Syst. Conf., 1989, pp. 292–295.

S. Li, “Real-time traffic signal control and automatic vehicle

M. Abadi, P. Barham, J. Chen, Z. Chen, X. Duan, L. Fang, J. A. Gilmer, T. Goodfellow, H. Haghpanahi, B. Hesse, G. Huang, E. Jang, M. J. Johnson, P. Jones, A. V. Kaplan, M. Kearnes, G. Kaiser, M. Kaiser, S. Kang, J. Keutzer, M. A. Krause, A. Kusy, J. B. Lin, L. Liu, M. S. Lipson, W. Liu, J. Liu, A. Loshchilov, J. Loshchilov, X. Muller, R. X. N. Najibi, D. N. Nokhodchi, D. Qi, B. Qi, Y. Qi, V. Ramanan, A. Ramesh, A. Rawlik, J. Reddy, N. Rewal, G. Sastry, R. Sanner, A. Sato, Q. Shao, J. Shazeer, N. Shaw, L. B. Shlens, T. T. Shlens, P. Shi, A. Shochet, E. Shorikas, C. Singla, S. Sokolsky, J. C. Stueckler, J. R. Sun, T. Tsoi, V. Ulyanov, Y. Yu, Y. Yuan, E. Zhavoronkov, A. Zhavoronkov, A. Zhavoronkov, M. Zhavoronkov, and B. Zhavoronkov, “TensorFlow: A system for large-scale machine learning,” in Proc. OSDI, vol. 16, 2016, pp. 265–283.

J. Glick. (2015). 

https://www.tensorflow.org/docs/2.7.1/learn

Paratransit and planning engineering from the Bei-

SHURONG LI received the B.S. degree in transpor-
tation and planning engineering from the Beijing

Institute of Technology, China, in 2016. She is cur-
rently pursuing the Ph.D. degree with the

MOT Key Laboratory of Transport Industry of

Big Data Application Technologies for Compre-
hensive Transport, School of Traffic and Trans-
portation, Beijing Jiaotong University, Beijing,
China. Her current research interests include adap-
tive traffic signal control and automatic vehicle
trajectory planning.

T. Zeng and U. Zahid. “Long short-term memory recurrent neural network architectures for urdu acoustic modeling,” Int. J. Speech Technol., vol. 22, no. 1, pp. 21–30, Nov. 2018.

B. Bakker. “Reinforcement learning with long short-term memory,” in Proc. Adv. Neural Inf. Process. Syst., vol. 2002, pp. 1475–1482.

Y. Yin. “Robust optimal traffic signal timing,” Transp. Res. B, Methodol., vol. 42, no. 10, pp. 911–924, Dec. 2008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0191261508000350

D. I. Robertson. “‘TRANSYT’ method for area traffic control,” Traffic Eng. Control, vol. 10, no. 6, pp. 181–182, 1969.

D. Husch and J. Albeck. Synchro 6: Traffic Signal Software, User Guide. Albany, CA, USA: Trafficware Corporation, 2003.

J. D. C. Little, M. D. Nelson, and N. M. Gartner. “MAXBAND: A versatile program for setting signals on arteries and triangular networks,” Transp. Res. Rec., no. 795, pp. 40–48, 1981.

K. Kyamakya, W. Mathis, R. Stoop, J. C. Chedjou, and Z. Li, Recent Advances in Nonlinear Dynamics and Synchronization: With Selected Applications in Electrical Engineering, Neurocomputing, and Transportation, vol. 109. Springer, 2017.

R. A. Vincent and C. P. Young. “Self-optimising traffic signal control using microprocessors. The CSIRO MOTA strategy for isolated intersections,” Traffic Eng. Control, vol. 27, nos. 7–8, pp. 385–387, 1986.

Y. Wang, D. Wang, S. Jin, N. Xiao, Y. Li, and E. Frazzoli, “Iterative tuning with reactive compensation for urban traffic signal control,” IEEE Trans. Control Syst. Technol., vol. 25, no. 6, pp. 2047–2059, Nov. 2017.

S. Timoteo, C. G. Panayioutou, and M. M. Poulacopou, “Towards distributed online cooperative traffic signal control using the cell transmission model,” in Proc. 16th Int. IEEE Conf. Intell. Transp. Syst. (ITSC ), Oct. 2013, pp. 1737–1742.
CHONG WEI received the M.S. and Ph.D. degrees from Kobe University, Kobe, Japan, in 2007 and 2009, respectively. He was appointed as the Ph.D. Supervisor, in 2015. He is currently an Associate Professor with the School of Traffic and Transportation, Beijing Jiaotong University, Beijing, China. His research areas include dynamic traffic network modeling, vehicle routing problem, and traffic flow theory. He also serves as the Editorial Board Member for multiple international journals, such as the *International Journal of Intelligent Traffic System* and the *International Journal of Transport*.

XUEDONG YAN received the Ph.D. degree in civil engineering from the University of Central Florida, Orlando, FL, USA. He is currently a Professor with the School of Traffic and Transportation, Beijing Jiaotong University, Beijing, China, where he is also the Executive Director of the MOT Key Laboratory of Transport Industry of Big Data Application Technologies for Comprehensive Transport and the Vice President. His research interests include traffic safety, driving simulation, urban planning, and management.

LU MA received the Ph.D. degree in civil engineering from the University of Florida. He is currently an Associate Professor with the School of Traffic and Transportation, Beijing Jiaotong University, Beijing, China. His research interests include intelligent transportation systems, transportation planning, traffic flow theory, and traffic accident analysis.

DEQI CHEN received the B.S. degree in transportation and planning engineering from Inner Mongolia University, China, in 2016. He is currently pursuing the Ph.D. degree with the MOT Key Laboratory of Transport Industry of Big Data Application Technologies for Comprehensive Transport, School of Traffic and Transportation, Beijing Jiaotong University, Beijing, China. His current research interests include data-driven short-term traffic prediction and intelligent transportation systems.

YING WANG received the B.S. degree from the School of Engineering and Technology, Northeast Forestry University, Harbin, China, in 2017. She is currently pursuing the Ph.D. degree with the MOT Key Laboratory of Transport Industry of Big Data Application Technologies for Comprehensive Transport, School of Traffic and Transportation, Beijing Jiaotong University, Beijing, China. Her research interest includes the trajectory planning of intelligent vehicles and nonlinear optimization.