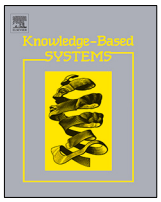
Knowledge-Based Systems 267 (2023) 110440

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys



Deep reinforcement learning for traffic signal control with consistent state and reward design approach

Salah Bouktif [a,*], Abderraouf Cheniki [a], Ali Ouni [b], Hesham El-Sayed [a]

[a] CIT, United Arab Emirates University, United Arab Emirates
[b] ETS Montreal, University of Quebec, QC, Canada

ARTICLE INFO

Article history:
Received 7 May 2022
Received in revised form 7 September 2022
Accepted 28 February 2023
Available online 5 March 2023

Keywords:
Traffic signal control
Traffic optimization
Reinforcement learning
Double deep Q-Network

ABSTRACT

Intelligent Transportation Systems are essential due to the increased number of traffic congestion problems and challenges nowadays. Traffic Signal Control (TSC) plays a critical role in optimizing the traffic flow and mitigating the congestion within the urban areas. Various research works have been conducted to enhance the behavior of TSCs at intersections and subsequently reduce the traffic congestion. Researchers recently leveraged Deep Learning (DL) and Reinforcement Learning (RL) techniques to optimize TSCs. In RL framework, the agent interacts with surrounding world through states, rewards and actions. The formulation of these key elements is crucial as they impact the way the RL agent behaves and optimizes its policy. However, most of existing frameworks rely on hand-crafted state and reward designs, restricting the RL agent from acting optimally. In this paper, we propose a novel approach to better formulate state and reward definitions in order to boost the performance of the traffic signal controller agent. The intuitive idea is to define both state and reward in a consistent and straightforward manner. We advocate that such a design approach helps achieving training stability and hence provides a rapid convergence to derive best policies. We consider the double deep Q-Network (DDQN) along with prioritized experience replay (PER) for the agent architecture. To evaluate the performance of our approach, we conduct series of simulations using the Simulation of Urban MObility (SUMO) environment. The statistical analysis of our results show that the performance of our proposal outperforms the state-of-the-art state and reward design approaches.

1. Introduction

In recent years, the rise of traffic congestion within urban areas has become a major and challenging issue especially in crowded cities. Traffic congestion causes a substantial economic loss due to the increasing amount of travel delays and fuel consumption in the world. For instance, it is found that the traffic congestion costs nearly $87 billions, with an average of $1,348 per driver yearly in the USA according to the 2018 global traffic scorecard report by INRIX company [1]. Therefore, cost-effective approaches aiming at optimizing traffic signal control (TSC) are of crucial importance to deal with such a costly traffic congestion. TSC is known to be a non-trivial and complex problem since the traffic environment is confronted with the characteristics of dynamic, complexity and uncertainty [2]. Traditional TSC systems that are mostly used nowadays all over the world, often fail to reduce the congestion cost as they are designed upon a set of pre-made assumptions about the traffic that usually deviate from the real world traffic conditions [3]. Examples of these systems include Fixed-Time [4] and adaptive traffic signal control systems (e.g., SOTL [5], Max-Pressure [6]). However, these systems mostly suffer from scalability issues especially with the increasing traffic volume and the subsequent congestion occurrences. The motivation for many researchers to propose smarter traffic signal controllers that can better perceive different situations of traffic states and better decide the TSC behavior, animates our current research works. Hence, in the last decade, the standard Reinforcement Learning (RL) technique (*i.e., Q-learning [7]*) has been widely used for many control and optimization problems, in particular, it was used to devise various solutions for the TSC problem [8,9]. Recently, with the flourishing Deep Learning (DL) as a technique having the potential of learning complex patterns in many domains, the combination of RL and DL (DRL) takes the TSC solution a step further towards more intelligent traffic intersections [10,11]. DRL-based solutions for TSC are mainly built on top of the deep Q-Network (DQN) architecture [12] and show a noticeable performance improvement over the traditional

* Corresponding author.
E-mail address: salahb@uaeu.ac.ae (S. Bouktif).

https://doi.org/10.1016/j.knosys.2023.110440

techniques since it can handle flexibly different traffic densities and dynamic flow patterns [13].

In the framework of reinforcement learning, the RL agent perceives the surrounding environment through the states of the environment, interacts by executing actions and receives immediate rewards that evaluate the associate actions correspondingly. These key elements are crucial as they impact the way the RL agent behaves and optimizes its policy (i.e., the rule of performing actions). A common issue that impacts the customization of reinforcement learning techniques to TSC problem, is how to formulate a proper design for both state and reward such that the agent learns optimal action-selection policy. Accordingly, discounted definition of state and reward will eventually worsen the convergence of the learning operation of the agent and consequently lead to a sub-optimal policy. On the other hand, the state and reward designs have to be realistic and applicable in real-world scenarios rather than a theoretical formula that uses assumptions that are more suitable in simulation contexts. In this work, we tailor the DRL framework in the context of TSC and propose an alternative definition for both state and reward based on a simple and intuitive idea that strives consistency among state and reward.

Striving for consistency among state and reward aims at defining an agent feedback that best guides the learning process to an optimal policy. By consistency, the reward functions reproduce similar properties as the state representation. The latter should be conducive to direct reward specification and provide the appropriate level of abstraction – (e.g., having the appropriate building blocks). For example, in the case of state representation based on single information (e.g., number of vehicles in each lane), consistency suggests to design a monolithic reward function that guides the learning process towards a monolithic goal which tends, in this case, to minimize the number of vehicles at the intersection. The consistency also suggests to define a continuous reward function if the state space is continuous, and similarly for the discrete case. A good reward function should be concise, simple and tied with how we define state space representation. Another dimension of the consistency, that we advocate in this paper, is the simplicity of defining reward and state. Indeed, avoiding complicated state representation and determining how much information is enough and effective to describe the state of an intersection, are interesting research questions.

What encourages striving for the consistency concept is the nature of the reward in the problem of traffic signal control (TSC). Although the reward may actually be a consequence of some complex temporally extended agent behavior, in our TSC problem, the reward functions are inherently Markovian (i.e., they depend mostly on the current state). They typically map current state, or state and action, to a scalar reward value, which incites simplicity. Therefore, with a simple representation of the state, the training process will explore a smaller state space, which reduces the learning time. Moreover, a consistent definition of reward that reflects the goodness of the current state, will be a relevant input for the learner to accurately estimate the best action-value appropriate for best-action selection. This means that at each step in the training process, the agent performance is bounded with no excessive fluctuations around action-value estimates. Such a training stability leads to a faster convergence to the best performance, hence, our consistency based approach is expected to derive an optimal policy for more intelligent TSC.

In short, the main contributions of this paper can be summarized in the followings:

- We propose a new way of tailoring RL for TSC that strives for consistency among state representation and reward definition.
- We particularly propose three pairs of definitions for state representation and reward supporting consistency and simplicity.
- We carry out an extended empirical study that illustrates how striving for consistency improves the training convergence and stability, and subsequently learns the optimal policy to best operate the TSC. The experiments conducted within this study aims also to evaluate our proposal approach on various non-uniform synthetic traffic flow data, and to compare it with state-of-the-art benchmarks.

The rest of this paper is organized as follows. In Section 2, we review the literature and discuss the related works. Section 3 provides preliminaries and theoretical backgrounds. Section 4 states the problem solved in this research work. Our proposed methodology is described in Section 5. In Section 6, we present an experimental evaluation of our proposal and discuss the results. Finally, in Section 8, we draw our conclusions and discuss our future works.

2. Literature review

Reinforcement Learning (RL) based approaches have received a considerable interest resulting in various contributions in several domains ranging from video games [14], and energy conservation [15], to VANETs security [16,17], especially after the Deep Learning (DL) flourishing. Recently, Deep reinforcement learning (DRL) have been leveraged in various research works in order to replace traditional traffic signal control systems by more intelligent ones [11,18–21]. These research works aim at improving TSCs in one or more of four enhancement paths, namely, (1) state representation, (2) reward definition, (3) action space definition, and (4) agent mission and architecture improvements. In the following, we will describe each of these aspects.

2.1. State representation

In the DRL-based TSC literature, the state represents what the agent perceives from traffic intersections in order to perform an appropriate action at each time step. The way of defining the state is critical since the agent behavior is highly dependent on the state received from the environment. For instance, Genders et al. [11] proposed the concept of discrete traffic state encoding (DTSE) which has been the most popular state definition since then. In DTSE, a length l of each lane is segmented into cells of length c. For each cell c, a set of information that is relevant to the state of a vehicle approaching the intersection is considered. Formally, DTSE is composed of three vectors, (1) the presence or absence of a vehicle, (2) its speed and (3) the current phase of the traffic. These vectors are then stacked to compose an image-like array to be fed into Convolutional Neural Networks (CNN) [22]. Beyond the DTSE resolution, other researchers [23] considered using a virtual snapshot image of the current intersection extracted from the simulator. The obtained image is divided into a grid of cells where each cell contains information like the *position* and *speed* of the vehicle. Instead of overwhelming the state by a dense matrix of information, it has been decided in other works [24] to use a feature-based value vector to represent the state of the traffic at the intersection. In such vector representation, each element of the vector v can store information about a given lane, like the number of vehicles, the queue length, and updated waiting time of vehicles. These are common choices as adopted in [25]. As long as the sensors and the induction loops are already installed, vector-based representation can be widely implemented at intersections as opposed to the aforementioned complex representations. Sometimes trying simpler state representation as advocated by Zheng et al. [24] can work

better and show superior performance of RL agent. Indeed, in the latter work, a concise state and reward design represented by the number of vehicles and queue length for state and reward respectively. Though the results by Zheng et al. [24] are superior, the state and reward definitions need to be re-examined and further enhanced as it is investigated in this work.

2.2. Reward definition

The reward is the feedback obtained from the environment to the agent after performing an action. It defines what are the good and bad events for the agent [26]. Though it is a scalar value, defining the reward has crucial implications on the learning of the reinforcement learning agent. When optimizing traffic control using RL, the common global objective of the TSC agent is to minimize the average travel time of vehicles starting from their origin location to their destination [11]. Since the average travel time is a long-term objective where the agent does not receive the action awards until the end of episode, it cannot be fed directly to the agent as a reward formula. Instead, it is required to use alternative reward definitions to better approximate the main objective. Various reward formulae have been suggested. Most common choices of reward definitions include change in total waiting time, change in queue length, and change in cumulative vehicle delay [10,11,22,24]. A more elaborated reward definition was proposed by Wei et al. [25] to define the reward as the weighted sum of multiple versions of reward. More recently, Zheng et al. [24] conducted a comparative of different formula and found that formulating reward as queue length achieves better performance than the above mentioned formulae including the weighted sum of different reward definitions. Supported by traditional algorithms, Wei et al. [13] adopted the concept of "pressure" as a reward formula. Their approach proves an improvement in performance over previous works. Our work is largely inspired by Zheng et al. [24]. In particular, we further extend the formulation and the design of state and reward together by simply striving consistency in both definitions. Explicitly, we adopt the idea of having concise and similar formulae for both state and reward. We hypothesize that such a design approach helps converging rapidly to good policies since the state and reward are of uncomplicated formulae and thus effortless search for optimal policies.

2.3. Action space definition

When behaving optimally, The RL agent should take the appropriate action for the current observed state that yields the optimal discounted cumulative reward. The definition of action space varies depending on the application. In RL with traffic signal control research, the action affects the phase of traffic signals in various ways. Mostly, action spaces fall into three types. First, a binary action which decides whether to extend the current phase or change into the next phase from a predefined sequence of phases [24,25]. A second type consists of a larger action space where the agent chooses among four or eight phases in a four-leg intersection where the agent executes an action without sticking to any order. These two types of discrete action spaces are the most common choices found in the literature [11,13,27]. A third type employed in other works, uses a continuous space where the agent predicts the most appropriate phase duration splits in the fixed cycle length [28].

2.4. Agent's architecture

The agent is the brain in reinforcement learning that learns from the environment and performs actions to get maximum cumulative reward. In deep RL based TSC, different agent architectures can be found according to the large number of developed deep learning architectures. For instance, for the vector-based feature state, the multi-layer perceptron (MLP) neural network is commonly used [24,25,28] where the output of the network is the Q-value of action–state pair. When using an image-like array of features as DTSE representation, an appropriate choice would be the convolutional neural networks CNN architecture. CNN can capture useful features from image pixels and map them into actions [11,25,29]. To capture temporal sequential dependencies of the intersection status, recurrent neural networks RNN (*e.g.*, Long Short-Term Memory LSTM) architectures are commonly utilized [30]. LSTMs are most suitable for improving the agent's decision making since they provide the ability to encode the history of intersection (different lengths of sequence of states) into one vector.

Unquestionably, existing research works in traffic signal control have been focusing on contributing in one of the four previously mentioned directions. The problem of how to formulate the state and reward in reinforcement learning specifically for TSC is still an open area for contribution. In this paper, we extend the work by [24] and propose an alternative way to formulate the state and reward definitions for traffic signal control which strives simplicity and consistency among both state and reward formulae.

3. Background and preliminaries

In this section we introduce essential notions and theoretical background in the field of reinforcement learning and deep reinforcement learning used throughout this paper.

3.1. Reinforcement learning

Reinforcement learning (RL) refers to both a learning problem generally aiming to control a system by maximizing a long-term objective. RL is a paradigm of machine learning [31] where the learner named the '*agent*' learns control policies by interacting with the surrounding environment through "trial and error" learning. Formally, RL problems can be formulated in the framework of Markovian Decision Processes (MDPs) [26], characterized by a set of terms $\langle \mathcal{S}, \mathcal{P}, \mathcal{A}, \mathcal{R}, \gamma \rangle$. Where $\mathcal{S}$ is the state space, $\mathcal{P}$ is the probability of transition, $\mathcal{A}$ is the action space, $\mathcal{R}$ is the reward and γ is the discount factor.

State Space $\mathcal{S}$: $\mathcal{S}$ is a finite set of Markov states s_t of the environment that can be used by the agent to decide what to decide next. if a state is Markovian, the history of states can be neglected, and the agent can rely on the current state s_t solely instead of the whole history.

Action Space $\mathcal{A}$: $\mathcal{A}$ is a set of legal actions a_t that can be taken by the agent. At a time step t, the agent selects the optimal action from action space $\mathcal{A}$ following a policy π which maximizes the long-term expected return.

Transition Probability $\mathcal{P}$. For each triplet $(s_t, a_t, s_{t+1}) \in S \times A \times S$, there is a probability of transition $\mathcal{P}(s_{t+1}|s_t, a_t)$ which gives the probability of moving from state s_t to state s_{t+1} by taking an action a_t.

Reward $\mathcal{R}$: The agent performs an action a_t at the time step t according to its policy and obtains an immediate reward R_t from the environment, where $\mathcal{R}_t \in \mathbb{R}$ is a scalar variable given by a reward function $r : S \times A \to \mathbb{R}$. The expected return G_t is defined as the total discounted reward starting from the time step t:

$$G_t = \mathcal{R}_t + \gamma\ \mathcal{R}_{t+1} + \cdots = \sum_{k=0}^{n} \gamma^k \mathcal{R}_{t+k} \tag{1}$$

where $\gamma \in [0, 1]$ is the discount factor multiplying the future expected rewards. It denotes the importance of future rewards versus the immediate rewards.

The goal of the agent is to find a policy π (rule of choosing an action given a state) in a way that maximizes the cumulative discounted reward. In our approach, we use Q-learning dynamic programming algorithm [7], a model-free value-based and off-policy (policy is learned implicitly) type of reinforcement learning. In Q-learning, the Q-function estimates the expectation of discounted rewards of an action given a state (*i.e.*, It measures how good is the chosen action given the current state).

$$Q^{\pi}(s, a) = \mathbb{E}[G_t | s, a, \pi] \tag{2}$$

Given that the optimal action-value function $Q^*(s, a)$ is the one which gives the maximum expected return, then the optimal policy π^* can be found by choosing the action a_t that gives maximum Q-value for the given state s:

$$a^* = \underset{a \in \mathcal{A}}{\text{argmax}} Q^*(s, a) \tag{3}$$

The optimal Q-function follows Bellman equation and it can be learned through an iterative update process.

$$Q_{t+1}(s, a) = Q_t(s, a) + \alpha\ \delta_{t+1} \tag{4}$$

where $0 < \alpha < 1$ is the learning rate, and δ_{t+1} is the temporal difference TD-error:

$$\delta_{t+1}(s, a) = \mathcal{R}_t + \gamma \max_{a_{t+1} \in \mathcal{A}} Q_t(s_{t+1}, a_{t+1}) - Q_t(s, a) \tag{5}$$

We denote $y_t = \mathcal{R}_t + \gamma \max_{a_{t+1} \in \mathcal{A}} Q_t(s_{t+1}, a_{t+1})$ as TD-target.

3.2. Deep reinforcement leaning

In the standard Q-learning algorithm [7], the values of state–action pairs (Q-values) are stored in a tabular structure which is computationally inefficient and expensive for high-dimensional state spaces. As an alternative to linear function approximators, deep neural networks can handle high dimensional state spaces and capture complex state features to approximate the Q-values more efficiently. Deep Q-network (DQN) [32] is the standard deep reinforcement learning algorithm which is composed of an input layer which takes states (images or vectors) as input, a number of hidden layers to learn features, and an output layer to approximate Q-values of state–action pair. DQN uses a neural network function approximator with weights $\boldsymbol{\theta}$ and an experience replay memory to store the past experiences by adding at every time step $e_t = (s_t, a_t, \mathcal{R}_t, s_{t+1})$ to a memory. During the learning process, a mini-batch b is randomly sampled from the experience replay memory to apply the Q-learning updates using target values y_t as defined in Eq. (4):

$${y_t}^{DQN} = \mathcal{R}_t + \gamma \max_{a_{t+1} \in \mathcal{A}} Q_t(s_{t+1}, a_{t+1}; \boldsymbol{\theta}) \tag{6}$$

The aim of the learning process is to have a Q-network that accurately predicts the target values in Eq. (6). Thus, the objective of the learning algorithm is to minimize the loss function:

$$\mathcal{L}_t(\boldsymbol{\theta}) = \mathbb{E}[(y_t - Q(s_t, a_t; \boldsymbol{\theta}_t))^2] \tag{7}$$

Due to some stability issues faced in deep Q-learning, Hasselt et al. [33] proposed an extension of DQN by adding a separate target network that is involved in calculating the target values y_t. Target network with weights $\boldsymbol{\theta}^-$ are fixed and updated after $\tau-$steps by cloning the main Q-network weights $\boldsymbol{\theta}$. Formally, ${y_t}^{DDQN}$ is defined as follows:

$${y_t}^{DDQN} = \mathcal{R}_t + \gamma\ Q_t\left(s_{t+1}, \underset{a_{t+1} \in \mathcal{A}}{\text{argmax}} Q_t(s_{t+1}, a; \boldsymbol{\theta}); \boldsymbol{\theta}^-\right) \tag{8}$$

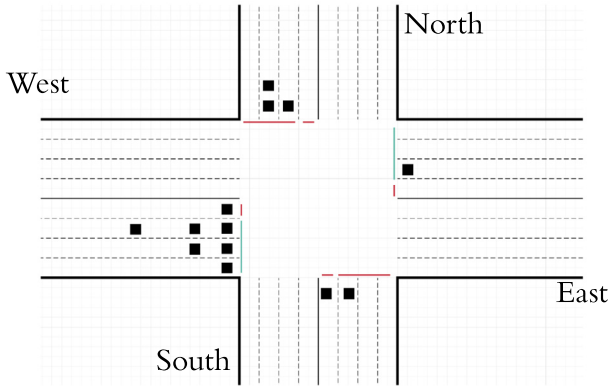


Fig. 1. Four-lane Four-armed Traffic Intersection.

4. Problem definition

Before diving into the details of our approach, we first provide, in this section, the problem definition for our DRL formulation for the TSC.

Definition. Given a traffic signal intersection as the environment E, the agent G receives a state $s_t \in \mathcal{S}$, performs an action $a_t \in \mathcal{A}$ and collects rewards R_t from the environment. The goal of the agent is to decide the optimal action a_t (selection of a phase P) for each perceived state s_t which results in minimizing the average vehicular travel time by maximizing the expected discounted return, where the discounted return is defined as follows:

$$G_t = \sum_{k=0}^{n} \gamma^k \mathcal{R}_{t+k} \tag{9}$$

where γ is the discount factor, t is the current time-step, n is the final time-step and $\mathcal{R}_{t+k}$ are the future rewards.

In order to sustainably ensure the best decision when taking actions, the agent perpetually learn and improve his policy of taking decision. The dynamic nature of the TSC environment needs a continuous learning of the decision policy.

In the following, we describe each key element from our problem definition, including the environment E, phase of signal P, agent G, and travel time TT.

Environment E. The environment E is defined as a four-direction (*i.e.*, East 'E', West 'W', North 'N', South 'S') and four-lane intersection. There are 8 distinct vehicle movements with a green signal for each movement (*e.g.*, straight East–West 'EW', East–West-left 'EW-L'). Restricted by road safety measures, we combine non-conflicting green signals forming 4 phases. Left-turn movements are considered separately, and right-turns are jointed to '*going-straight*' movements. As an illustrative example, Fig. 1 shows the environment structure with phase East–West activated.

Phase of signals P. The phase of signals P is defined as a set of signals ('G' for green, 'r' for red and 'y' for yellow) at an intersection. For instance, the phase 'GrGr' means green for north and south directions and red for east and west directions. For safety restrictions, green signals must be set such as no conflicting movements occur and must be followed by a yellow phase for a short period of time (*e.g.*, 'yryr' after 'GrGr').

Agent G. The agent G is the main component which observes the state of the environment E (intersection), selects an action (a phase P) according to its policy and receives an immediate reward $\mathcal{R}_t$ at each time step. A general structure of agent-environment interaction is shown in Fig. 2.

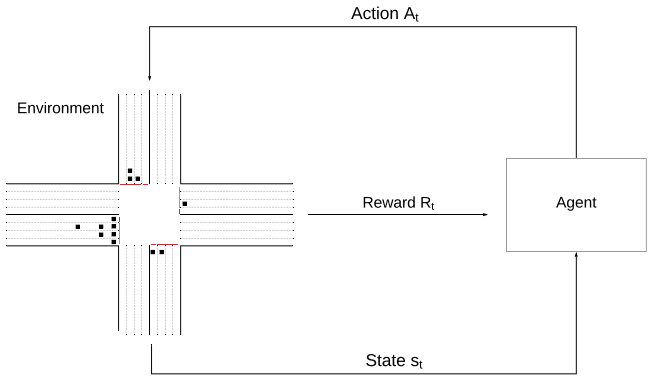


Fig. 2. Process of reinforcement learning for the traffic signal control problem.

Table 1
Notations.

Notation	Description
s	*State*
a	*Action*
R	*Reward*
G	*Agent*
E	*Environment*
P	*Phase of signals*
L	*Number of lanes*
H	*Horizon of detected vehicles on a lane*
EW	*East–West movement*
NS	*North–South movement*
EW-L	*East–West-Left movement*
NS-L	*North–South-Left movement*
v_l	*Number of vehicles on lane l*
q_l	*Queue length on lane l*
Wt_l	*Waiting time of vehicles on lane l*

Travel Time TT. The travel time for a vehicle can be defined as the time it takes for the vehicle to accomplish its planned route starting from an origin location until it reaches its destination. The travel time is calculated as follows:

$$\text{Travel Time} = t_{j\_start} - t_{j\_end} \tag{10}$$

where t_{j_start} is the time the vehicle j enters the environment and t_{j_end} is the time the vehicle j exited the environment. For the sake of clarity, a list of important notations used in this paper is summarized in Table 1.

5. Approach: Deep reinforcement learning for traffic signal control

In this section, we describe the proposed approach that adopts a deep reinforcement learning framework as a general solution of the TSC problem. We also tailor this framework and propose a specific solution that strives Consistency among State and Reward Definitions "CSRD".

Fig. 3 depicts the reinforcement learning framework adopted for our specific CSRD solution. First, at the time step t the agent acquires the current state s_t, describing the intersection environment, as a vector. Using this vector and according to its current policy of controlling the traffic signal, the agent decides the optimal action a_t. The output action consists in selecting a particular light phase. The agent then observes the next state s_{t+1} as a result of its taken action, from which the reward $\mathcal{R}$ is at the same time extracted. The reward $\mathcal{R}$ eventually undergoes a penalty process and finally a tuple of $\langle s_t, a_t, \mathcal{R}, s_{t+1}\rangle$ is stored in the agent's memory M. At specific moments, the agent learns and updates its policy from minibatches sampled from the memory as described in Section 3.

The most essential components of the proposed framework include the state vector, reward function, penalty process, action space and agent's learning architecture. In the following paragraphs, we describe each of these elements.

5.1. State space

In our framework, we propose the state to be enough representative of the environment in the form of a simple vector. The idea is to define the state and the reward in a consistent and concise manner by which the agent is expected to converge to the optimal policy. Therefore, we define the state as one of the following alternatives:

(a) The number of all vehicles v_l in each horizon H of each lane l at time step t. We consider the total number of lanes $L = 16$.

(b) The queue length of vehicles q_l in each lane l at time step t. Queuing vehicles are those with speed less than 0.1 m/s.

(c) The waiting time Wt_l for all vehicles on the lane l at time step t.

In addition, the current phase of signals P_t at the intersection is included in the state vector and represented by an integer from the set {0, 1, 2, 3}. The notation of the state vector representation for a selected definition (e.g., $\lambda = v, q, Wt$) at a given time step t is defined as:

$$s_t(\lambda) = \begin{bmatrix} \lambda_{0t} \\ \lambda_{1t} \\ . \\ . \\ \lambda_{L-1t} \\ P_t \end{bmatrix} \tag{11}$$

where $s_t \in \mathbb{R}^{L+|P|}$ and $|P|$ is the dimension of the current signal phase vector (i.e., $|P| = 1$).

In the above suggested definitions of the state representation, consistency is achieved by the simplicity of specifying a reward function based on the information used to represent the state, for example, in Definition (a), the state is represented by a vector where each coordinate represents the number of vehicles per lane, and an extra coordinate for the current signal phase. For the sake of consistency, the reward will be defined using simply the number of vehicles. Explicitly, the reward is defined as the negative sum the number of vehicles, similarly for Definitions (b) and (c). In practical applications, the proposed state and reward function can be captured by sensors. A good criterion of choosing among the alternatives is the amount of error associate with the concerned traffic information. It also depends on the availability of infrastructure and the ease of the collection process. Accordingly, the appropriateness of each alternative can be evaluated. However, in this paper, our aim is rather to show that the consistency and simplicity of state representation and reward function for the TSC problem can be achieved with many definition alternatives of state and reward as long as they are striving for consistency.

5.2. Reward function

Striving for consistency among state and reward design, and according to the above state definitions, we propose three reward functions obtained from observing the state at the time step $t+1$.

Consistent with either (a), (b) or (c) state definitions, the reward $\mathcal{R}_t$ is defined as the negative sum of the number of vehicles,

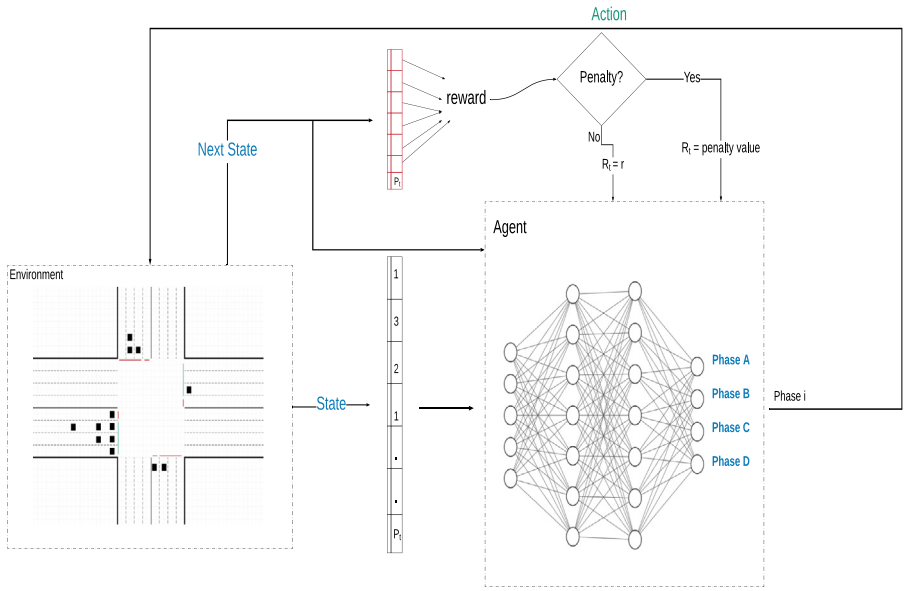


Fig. 3. The proposed framework for traffic signal control with consistent state and reward design.

queuing vehicles or total waiting time respectively, across all lanes $l \in L$ located in front of the intersection, formulated as:

$$\mathcal{R}_t = -\sum_{l=0}^{L-1} s_{t+1}(\lambda_l) \tag{12}$$

Where $\lambda = v, q$ or Wt according to the state definition. In the above formula, the negative sign of the reward means that the objective of the agent is to maximize the reward value.

5.3. Penalty process

Along with the defined reward, we introduce a penalty function that eventually assigns a negative score p called "*penalty value*" to the reward when the episodic sum of rewards $ep\_rewards$ is inferior to a certain threshold '*threshold*'. We refer to this penalty process as "Type 1". Penalizing the agent decision is analogous to mimic penalizing a player in game scenarios when the latter crashes and the episode ends. By applying such a penalty, we guide the agent to avoid taking worst actions in future when learning from experiences.

$$\mathcal{R}_t = \begin{cases} p, & \text{if } ep\_rewards \leq threshold \\ \mathcal{R}_t & \text{otherwise} \end{cases} \tag{13}$$

An alternative penalty process, namely, the *teleporting* can be applied in simulation. Teleporting consists of withdrawing the vehicle from the simulation and reinserting it into the network if there is enough place in the lane which allows to continue its drive [34]. Formally, a vehicle is said to be "teleported", if its waiting time $Wt_{vehicle}$ in the intersection exceeds a certain time interval $T_{teleport}$. If teleporting occurs during the simulation, a penalty value p' is assigned to the reward and the episode ends. We refer to the latter penalty process as "Type 2". It is important to note that teleporting is simply a phenomenon that works as a flag in the simulation to invoke the penalty process on the agent.

With respect to real intersection, the teleporting operation does not happen, but it can be replaced by any tool of detecting vehicles that wait for an excessive time (exceeding a certain threshold). Hence, by detecting these stuck vehicles, the agent's decision will be penalized.

$$\mathcal{R}_t = \begin{cases} p', & \text{if } Wt_{vehicle} \geq T_{teleport} \\ \mathcal{R}_t & \text{otherwise} \end{cases} \tag{14}$$

The p and p' in Eqs. (15) and (16) are determined following a rule of thumb manner. That means, when the traffic at the intersection reaches such a terrible state, this is a big mistaken action decision made by the agent, and it has to be given an extreme negative reward (e.g., 10, 20 or even 30 time less than the usual negative reward) so next time the agent has to avoid earlier these worsening actions that lead the traffic to such a bad state. the threshold value is chosen not to be very low that it curbs the agent from trial-and-error, and not to be very high with no impact on the learning process.

5.4. Action space

The agent's action a_t at the time t will be performed by selecting any light phase from a set $\mathcal{A}$ of four phases without any prefixed order (*i.e.*, east–west EW-green, north–south NS-green, EW-Left-green, NS-Left-green) where each phase allows a traffic movement to cross the intersection for a duration of time followed by a yellow signal for safety reasons. The actions are labeled by numerical labels as from the discrete action space $\mathcal{A} = \{0, 1, 2, 3\}$.

5.5. Agent's architecture

We consider the Double DQN architecture as the network structure of our agent, accompanied with a prioritized experience replay PER memory [35]. Two multilayer perceptron neural networks are employed, the first one which is the main network, is used to predict Q-values of actions. While the second network, namely, the target network, is involved in updating the first network as expressed in Eq. (8). Both networks enclose an input layer to receive the input vector of state of size $L + 1$, where L is the number of lanes. The input layer should fit the state vector of size $L + 1$, holding L information for L lanes (e.g., queue length in a lane), in addition to the signal phase information. The networks also enclose three hidden layers with 64 neurons each and a *ReLU*

activation function for each layer. The output layer is composed of 4 neurons and a *Linear* activation function to approximate the Q-values. During the learning phase, the agent perceives the state s_t of the environment at each time step t, selects and performs an action a_t according to ϵ-greedy action selection policy:

$$a_t = \begin{cases} \text{a random action from } \mathcal{A} & \text{with probability } \epsilon, \\ \underset{a_t \in \mathcal{A}}{\text{argmax}} Q(s_t, a_t; \boldsymbol{\theta}) & \text{with probability } 1-\epsilon, \end{cases} \quad (15)$$

The agent then receives an immediate reward $\mathcal{R}_t$ as a feedback for its action a_t. The tuple $\langle s_t, a_t, \mathcal{R}_t, s_{t+1} \rangle$ is used to update the model weights $\boldsymbol{\theta}$ of the main network. Target network is updated once each τ updates of the main network by copying the weights of the main network. Algorithm 1 provides a brief pseudo-code of our approach : Double-DQN with PER memory for traffic signal control regarding the state and reward definitions in addition to suggested penalty procedures during learning process (Algorithm 2).

Algorithm 1 DDQN-PER for TSC with CSRD

Initialize $\boldsymbol{\theta}$ and $\boldsymbol{\theta}^{-}$, PER Memory M, minibatch size b, Penalty_Type, τ, counter C, *n_updates*, *start_learning*.
for episode $= 1, \ldots,$ N **do**
  Observe initial state s_0, take initial action a_0
  for $t = 1, \ldots, T$ **do**
    Observe state s_t , take action a_t
    Observe next state s_{t+1} and get $\mathcal{R}_t$ from s_{t+1}
    Execute **Penalty Process** (Penalty_Type)
    Calculate TD_{error}

$$TD_{error} = {y_t}^{DDQN} - Q_t(s_t, a_t; \boldsymbol{\theta}_t)$$

    Store $< TD_{error}, < s_t, a_t, \mathcal{R}_t, s_{t+1} >>$ in M.
  If episode $\geq$ *start_learning* **do**
    for $i = 1, \ldots,$ *n_updates* **do**
      Sample a random *mini batch* of size b from M.
      Update the agent weights $\boldsymbol{\theta}$ and increment C.
      If mod(C)==τ **do**
        Copy $\boldsymbol{\theta}$ weights to $\boldsymbol{\theta}^{-}$ weights .
      End if
    end for
  End if

Algorithm 2 Penalty Process

Initialize *threshold*, penalty values p, p', $T_{teleport}$
If *Penalty_type* $==$ 1 **do** ▷ Type 1: episode rewards penalty

$$\mathcal{R}_t = \begin{cases} \text{penalty value } \boldsymbol{p}, & \text{if } ep\_rewards \leq threshold \\ \mathcal{R}_t & \text{otherwise} \end{cases}$$

Else If *Penalty_type* $==$ 2 **do** ▷ Type 2: teleporting penalty

$$\mathcal{R}_t = \begin{cases} \text{penalty value } \boldsymbol{p'}, & \text{if } Wt_{vehicle} \geq T_{teleport} \\ \mathcal{R}_t & \text{otherwise} \end{cases}$$

Else
  Pass

6. Experiments

In this section, we begin by posing the research questions we aim to answer and the hypotheses we seek to verify. We then describe the simulation settings used to evaluate the proposed CSRD framework and validate our assumptions. These settings include the intersection environment structure, the vehicular flow data, and a set of tuned hyper parameters used during training and validation processes. We finally present and discuss the performance results while comparing our approach to other benchmarks including traditional and DRL approaches.

6.1. Hypotheses

By performing the set of the upcoming experiments using our approach and the benchmarks, we aim to answer the following research questions, (1) how does the proposed CSRD design approach anticipate the Travel Time TT minimization objective, (2) how much of improvement will the proposed CSRD approach introduce as compared with the Fixed Time and the Deep RL benchmarks, (3) how an agent trained using penalty process performs against an agent trained without penalty process, and finally (4) how the selection of green duration affects the performance of the traffic signal control. Based on these research question, we proposed a set of hypotheses to be tested on four traffic simulation experiments. In the four performed experiments, we assume that our proposed approach is, on one hand, performs better than the Fixed Time approach, and on the other hand, is a good as the best approach in the benchmarks. In line with these assumptions, we formulated and tested the following hypotheses on the four experiments stated in Table 2.

1. H_1 : The proposed CSRD design approach anticipates well the travel time TT minimization objective i.e., $Curve_{CSRD} \propto Curve_{Travel\ Time}$.
2. H_2 : The proposed CSRD approach performs notably better than the Fixed Timed approach in all simulated experiments.
3. H_3 : The performance of our CSRD approach is at least as high as the state-of-the-art Pressure-based approach in all simulated experiments.
4. H_4 : The agent trained with penalty process performs at least as good as the agent trained without penalty process.

6.2. Simulation settings

To experiment our approach in traffic signal control, we use the well-known open source Simulation of Urban MObility (SUMO) simulator,[1] to simulate the intersection environment [36]. SUMO is a flexible simulator that helps to customize traffic environment and integrate intelligence in order to model and investigate several intelligent solutions in different modes and circumstances of the traffic network including flow characteristics. With SUMO, we implement and experiment our proposed algorithms to control traffic signal functions. The different settings are described in the following paragraphs:

6.2.1. Intersection environment structure

We consider a 4-way geometry of intersection (*i.e.*, East, West, North, South), where each incoming/outgoing road have 4 lanes. All lanes are of 750 meters length. The maximum lane speed is set to 13.89 m/s (*i.e.*, the urban areas common speed limit [37]) and all lanes have the same priority and the same width. The left-most lane is dedicated for turning left solely and the rest of lanes can be occupied by straight or right-turn movements. Each green phase duration is 15 seconds followed by a yellow phase of 3 seconds.

[1] https://www.eclipse.org/sumo

Table 2
Traffic flow configurations.

Distribution	Type	Generated flow q	Flow rate category
Weibull Dist	Flow1	4000	High
	Flow2	1500	Low
Normal Dist	Flow3	4000	High
	Flow4	1500	Low

Table 3
Values used for training parameters.

Parameter	Description	Value
N	*Number of training episodes*	501
max_size	*PER Memory maximum size*	20000
b	*Minibatch size*	32
lr	*Learning rate*	0.001
γ	*Gamma factor*	0.95
eps_min	*minimum value of epsilon*	0.02
C	*Target network update counter*	5
yellow duration	*Yellow phase duration*	3 s
green duration	*Green phase duration*	15 s

6.2.2. Traffic flow

In order to simulate what happens in an intersection, the traffic flow of vehicles must be close enough to real traffic scenario. An ordinary traffic flow starts with low vehicular density at the beginning and increases until the peak of the flow, it then decreases back until the end of simulation. Common choices for such traffic flow scenarios that are widely reported in the literature [11,25,27] include the following types of random distributions: Burr distribution, Poisson random distribution, and Weibull Distribution. The latter type as well as the Normal distribution are selected to be adopted in our experiment as they simulate well the traffic flow behavior over time. Traffic flow is expressed in vehicles/hour and is generated as a configuration of the simulation. For each distribution type, we generate a synthetic traffic flow in every episode for a duration of 3600 seconds. Each vehicle of the flow has an Origin point and a Destination point (OD) and follows a route from origin to destination. Vehicle routes include going straight movements (North–South and East–West) and turning movements (Left-turns and Right-turns). We generate a percentage of 75% and 25% for straight movements and turning movements, respectively. Details about traffic flow configurations are shown in Table 2.

6.2.3. Parametric and training settings

After several runs of experiments and tuning of algorithm parameters, we set the latter as follows: the number of training episodes $N = 501$, where an episode lasts for 3800 seconds. After the episode $start\_learning = 5$ and at the end of every episode, the agent updates its main network and learns its new policy on minibatches of $b = 32$ experiences (i.e., one experience is a tuple of $\langle s_t, a_t, \mathcal{R}_t, s_{t+1}\rangle$) for $n\_updates = 3$ iterations. The agent's target network is updated every $\tau = 5$ updates of the main network. For both networks, we used a learning rate of 0.001 using RMSProp [38]. The discount gamma factor is set to $\gamma = 0.95$ for updating the Q-values. The size of the prioritized experience memory M is set to $max\_size = 20{,}000$ to store experiences along with their temporal difference error TD-error. In $\epsilon-$greedy action selection policy, we anneal ϵ from 1 down to $epsilon\_min = 0.02$. The geographic horizon H of state detection using "number of vehicles" is set to a distance of 150 m which was shown to be more practical choice than all-lane long. When we use the penalty process type 1, the *threshold* value is set empirically depending on the chosen reward definition. While in the type 2, we penalize the agent if we have more than 1 teleported vehicles. Table 3 summarizes various parameters used with their associated values.

Table 4
Scenarios of flow and state-reward definitions.

Scenarios	State and reward	Flow config.
Combinations of S&R with flow configurations	*Definition*(a) *Definition*(b) *Definition*(c)	[Flow1, Flow2]
Combinations of S&R with flow configurations	*Definition*(a) *Definition*(b) *Definition*(c)	[Flow3, Flow4]

Upon the traffic flow configurations listed in Table 2, we setup the training and testing scenarios based on state and reward definitions proposed in Section 5. To strive consistency, the definition options of state and reward are combined as follow (state option (a) with reward option (a)), (state option (b) with reward option (b)), and (state option (c) with reward option (c)). The three combinations are respectively termed in Table 4, *Definition (a)*, *Definition (b)*, and *Definition (c)*.

6.3. Performance evaluation metrics

To evaluate the performance of different methods of controlling the traffic signals, three metrics are considered in several previous studies [11,13,24] that we will use in our experiments, (1) the average travel time used as the main performance metric, since it is the prime metric to be minimized (2) the queue length and (3) the waiting time of vehicles. The different performance measures are defined as follows.

6.3.1. Average travel time (ATT)

It is defined as the total travel time of all vehicles divided by the number of vehicles, formally expressed by the following equation:

$$ATT = \frac{1}{N_{veh}} \sum_{j=0}^{N_{veh}} (t_{j,start} - t_{j,end}). \tag{16}$$

where N_{veh} is the total number of vehicles.

6.3.2. Queue length (QL)

The queue length of a lane is the total number of vehicles queuing on a lane. The queuing vehicles are those with a speed less than 0.1 m/s on a given lane (known in SUMO as vehicle in 'halting' state).

6.3.3. Waiting time (WT)

This measure corresponds to the waiting time spent by all vehicles on lanes. A vehicle is considered as waiting if its speed is less than 0.1 m/s. We will be using both accumulated and average waiting time of all vehicles across the intersection.

6.4. Benchmarks

For a practical evaluation of our proposed method, we select the following state-of-the-art benchmarks: Discrete Traffic State Encoding (DTSE), Light Intelligent Traffic (LIT) and Pressure-based (PressLight) approaches. In this paper we opt for the comparison of our approach with the above benchmarks rather than with traditional TSC methods because the aforementioned benchmarks were already compared to the traditional methods such as Green Wave, SOTL and Fixed Time, and shown to be more performant than them. However we keep comparing our approach to one of the traditional TSC methods to record and emphasize again

the significant outperformance of RL based approaches and in particular ours over the traditional TSC methods.

We introduce the benchmarks approaches in the following paragraphs.

6.4.1. DTSE-based

This approach uses deep reinforcement learning DQN agent, with the state design proposed by Genders et al. [11]. We utilize the implementation of DTSE provided by Vidali A. et al. [27] and we modify the DDQN-PER architecture to be similar to their architecture. We build the learning network architecture as a fully connected neural network of 4 layers of 400 neurons each, trained longer for more than 700 episodes. The state vector is based on both DTSE representation with variable cell size and the current phase state. The reward is defined as the change in waiting time between two action steps.

6.4.2. Pressure-based (PressLight)

This approach uses the pressure across the intersection as the reward function. This is the state-of-the-art reward definition proposed by [13]. The state formula involves both the number of incoming and outgoing vehicles in each lane as well as the phase status P_t. We use the DDQN-PER as the architecture for the pressure-based approach instead of the DQN used in their proposal to ensure a fair comparison to our approach among the proposed state and reward formula.

6.4.3. Light Intelligent Traffic (LIT)

This approach is proposed by Zheng et al. [24] for the state and reward definitions. Namely, they proposed the state formula based on the number of vehicles in each lane and, the sum of queue lengths for the reward formula. They stated that this design is sufficient for the RL agent to find the optimal policy under uniform traffic. Additionally, we use the DDQN-PER for the agent architecture of the LIT approach.

6.4.4. Fixed time approach

is the most common traditional traffic control approach that uses fixed phase duration with fixed cycle length and fixed order [39]. The duration of green phases is set to 30 seconds and the yellow phase duration is 3 seconds.

6.5. Results and discussion

To verify the hypotheses and answer the research questions, we present the obtained results of the experiments using the proposed approach based on the performance metrics mentioned earlier. The agent is trained and evaluated using different synthetic flow configurations and various state and reward formulae where the training is made on high flow Flow1 and evaluation is made on all the flow configurations. Further evaluations are made by comparing our approach to the benchmarks during the scenarios described in Table 4. The effect of adding the penalty process is studied and, the performance of our approach using different values of phase duration is evaluated as well.

6.5.1. How does the CSRD approach facilitate the TT minimization objective compared to alternative approaches during the learning process?

We tested the hypothesis (1) related to this question for three variants of state and reward formulae: Queue state and reward QSR (*Definition (a)*), Number of vehicles state and reward NSR (*Definition (b)*), and Waiting time state and reward WSR (*Definition (c)*).

Table 5
Statistical analysis results of QSR, NSR and WSR curves versus Travel Time curve during training.

	QSR_{Curve}	NSR_{Curve}	WSR_{Curve}
Mean	52.69	3324.01	47675.73
STDEV	56.88	1058.75	47564.59
ρ	0.96	0.98	0.96
p-value	< 0.0001	< 0.0001	< 0.0001
TT vs. *	(Two-tail)		

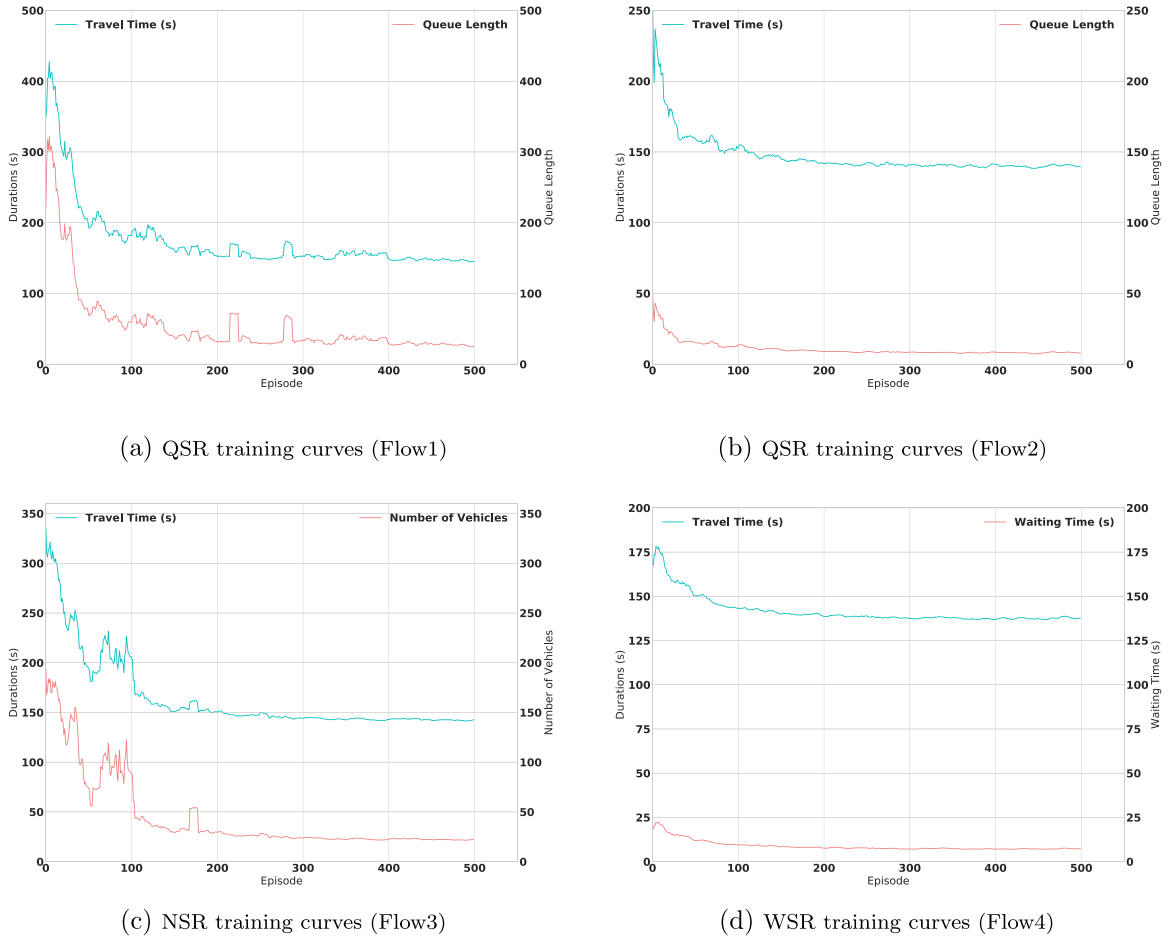
6.5.1.1. Impact of the consistency on the training performance. In order to show the impact of consistency among state and reward on the training performance, and subsequently on the long-term travel time optimization, we conducted four experiments to investigate the training process of QSR, NSR and WSR agents on four traffic configurations (i.e., Flow 1,2,3 and 4). The results are illustrated in the subplots of Fig. 4. As it can be noticed from these subplots, the reward's training curve related to QSR, NSR and WSR stably converge over episodes with nearly identical trends as the average travel time curve, which demonstrates that maximizing the proposed rewards (i.e., minimizing its opposite) leads directly to minimizing the travel time. This illustration supports, on one hand, that the idea of having consistent state and reward implemented in QSR, NSR and WSR, facilitates the training of the agent by speeding up the latter's convergence. On the other hand, the four subplots of Fig. 4 show, in different traffic configurations, that the proposed state and reward formulae are good estimators of the long-term travel time objective.

6.5.1.2. Training performance comparison: convergence speed and stability. To investigate on the benefits of consistent state and reward design on the training convergence speed and stability, we have conducted three experiments, each in which, we compare our approach implementations, namely, QSR, WSR and NSR, to one of the state-of-the-art benchmarks (DTSE, LIT, and PressLight). The training curves resulting from each experiments are depicted in one sub-figures (a), (b) and (c) of Fig. 5. In terms of the training performance, stability and convergence speed, one can observe from that our approach implementations (QSR, NSR and WSR) outperform the LIT and DTSE as shown in Fig. 5(a) and Fig. 5(b) and compete with the PressLight as shown in Fig. 5(c). Indeed, PressLight convergence is slightly slower than our approach convergence, however, it achieves a very comparable training performance after several episodes. Such results are expected since they reflect the effective impact of consistency among state and reward. Indeed, PressLight is the closest competitor to our approach thanks to its reward function that, to some extent, strives for consistency more than LIT and DTSE.

The statistical analysis of the results using t-test (see Table 5) shows that the null hypothesis H_0, assuming that the correlation between QSR/NSR/WSR curves against the Travel Time curve is zeros i.e., $\rho = 0$, is rejected with a very strong evidence (i.e., p-value < 0.001) in favor of the alternative hypothesis. This fact is true since the correlation coefficient is very high among the curves (see Table 5). As a response to question (1), it can be said that the proposed CSRD design approach facilitates well the travel time TT minimization objective and thus validating our hypothesis (1).

6.5.2. How is the performance of our approach compared with the performance of the state of the art benchmarks?

To answer the above research question, we present the testing performance of the trained agent on the different aforementioned flow scenarios. In this empirical evaluation, we would like to compare our approach to the three mentioned benchmarks



(a) QSR training curves (Flow1)

(b) QSR training curves (Flow2)

(c) NSR training curves (Flow3)

(d) WSR training curves (Flow4)

Fig. 4. Learning curves during Training of the QSR, NSR, and WSR agents on Flow1, Flow2, Flow3, and Flow4 respectively (curves are smoothed with 10 episodes window size).

Table 6
Performance comparison of our approaches to others with respect to average travel time (s).

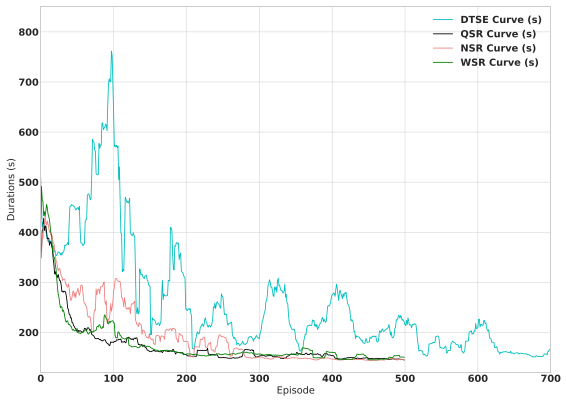
	Flow1	Flow2	Flow3	Flow4
Fixed-Time	260.69	164.48	217.30	161.42
DTSE	156.59	140.1	155.3	140.4
LIT	159.26	145.96	156.32	139.18
PressLight agent	145.54	138.4	144.16	136.11
QSR (ours)	147.20	140.36	144.81	138.57
NSR (ours)	**142.77**	**137.30**	**141.45**	**135.16**
WSR (ours)	145.20	139.68	143.44	138.26

namely, DTSE, LIT and PressLight agents in terms of testing performance. This comparison will serve to verify the Hypotheses H_2 which states that "the proposed CSRD approach performs notably better than the Fixed Timed approach in all simulated experiments" and H_3 which states that "the performance of our CSRD approach is at least as high as the state-of-the-art PressLight approach in all simulated experiments".

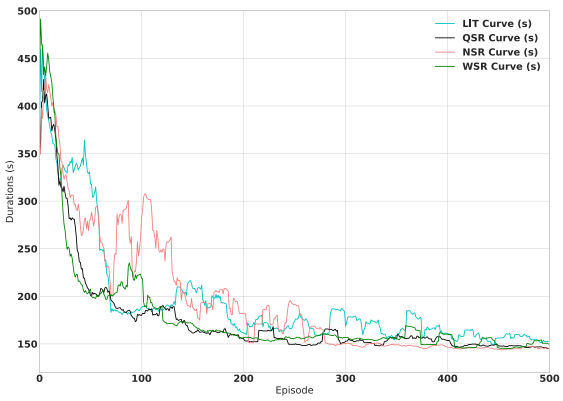
6.5.2.1. Comparison based on travel time. In our comparison we use the average travel time as the main performance measure being the primary term to be minimized. We run the simulations for the four flows Flow1 to Flow4 using 10 random seeds for all the proposed agents and the benchmarks. Table 6 summarizes the average travel time performance results obtained by our approach as well as by other benchmarks. The lower the average travel time score, the better is the performance. The comparison shows clearly that the Fixed-Time based traffic control algorithm achieved the lowest performance due to its static behavior that does not consider the traffic flow dynamic settings. With respect to deep reinforcement based benchmark approaches, the performance is significantly higher than that of the Fixed-Time solution due to the readiness of Deep RL based frameworks to deal flexibly with dynamic traffic flow patterns. However, our proposed solution outperformed methods like PressLight agent, LIT, and DTSE by better minimizing the average travel time. Moreover, our proposed variants of our approach, namely, QSR, NSR and WSR outperformed the other methods including the PressLight approach which achieved the highest performance among the benchmarks.

6.5.2.2. Interpretation of consistency analysis. From consistency perspective, the designs of state and reward in Pressure-based agent approach are more consistent than those of DTSE and LIT approaches, which explains the reason why it out-performs both of them. Indeed, the Pressure-based approach uses the number of vehicles as the state definition and, pressure (which is a function based on the number of incoming/outgoing vehicles) for the reward definition. The impact of consistency, is emphasized by our proposed design approach where the state and the reward are more consistent than in the other approaches.

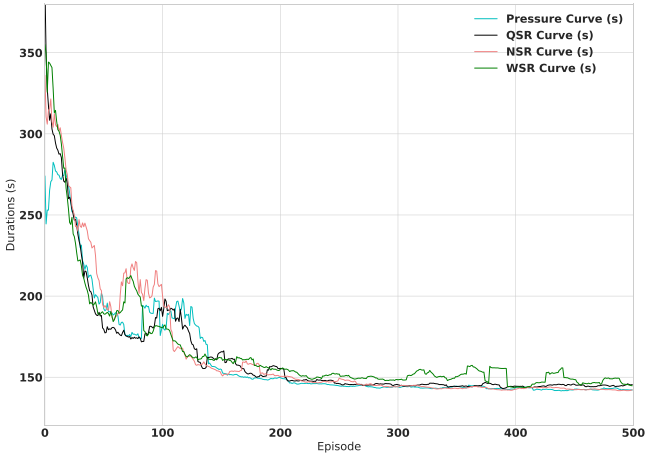
6.5.2.3. Comparison based on queue length. In this comparison we use the queue length as a performance measure at the intersection which is another important factor that is a good available estimator of the travel time. The results showing the Queue length performance achieved by our approach as well as the other approaches during an episode of simulation are depicted



(a) DTSE Learning curve v.s {QSR, NSR and WSR}



(b) LIT Learning curve v.s {QSR, NSR and WSR}



(c) PressLight Learning curve v.s {QSR, NSR and WSR}

Fig. 5. Learning curves of each benchmark compared to QSR, NSR and WSR agents during various traffic scenarios. (curves are smoothed with 10 episodes window size).

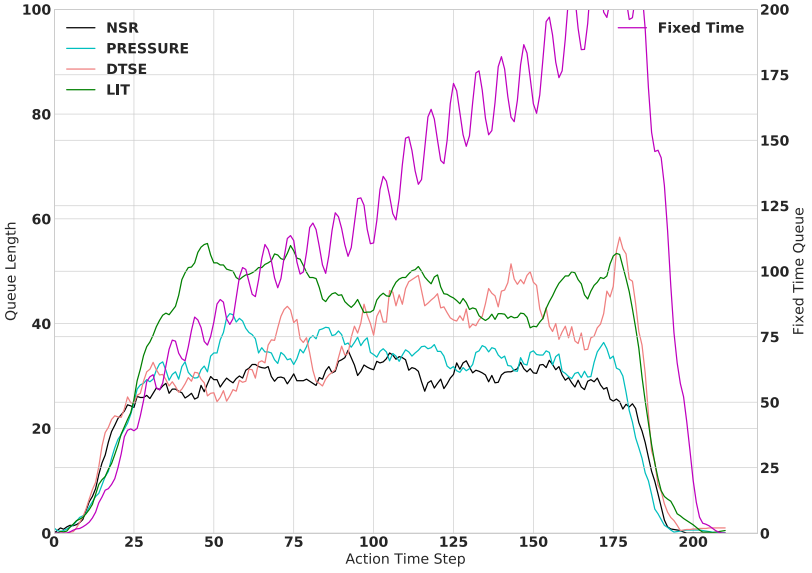


Fig. 6. Performance comparison of our approach and benchmarks during traffic simulation based on queue length (smoothed with 10 episodes window size).

in Fig. 6. At first glance, we notice that the Fixed-Time approach performance is out of competition as it cannot deal with dynamic traffic flow. Out of the Deep reinforcement learning approaches, we specifically observe that NSR agent has a lower and steady queue length curve (better performance and stability) as opposed to PressLight and the rest of benchmarks. In addition, one

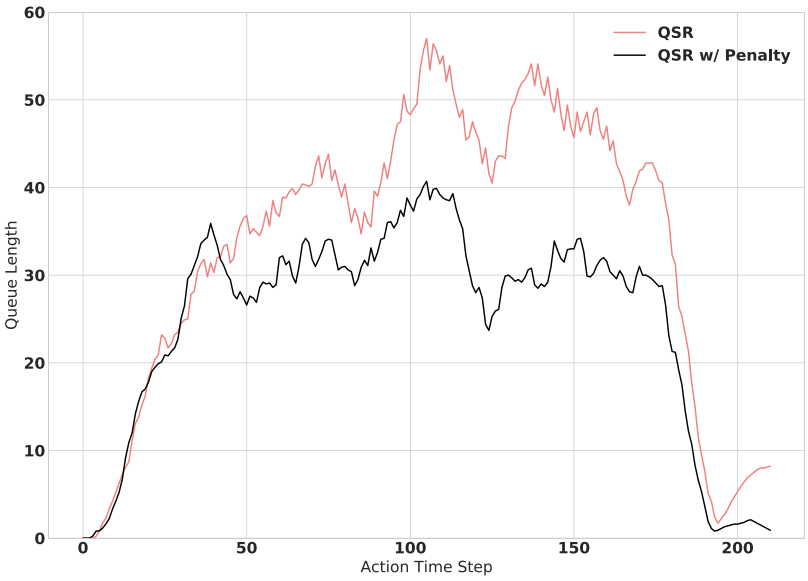


Fig. 7. Queue Length comparison of QSR with and without adding the penalty process (smoothed with 10 episodes window size). Agent trained with penalty has better queue length performance.

Table 7
Statistical analysis of CSRD approach against Pressure and Fixed benchmarks on high/low flow (Flow1/Flow2).

	NSR	*Fixed Time*	*PressLight*
Mean	142.7/137.3	260.6/164.4	145.5/137.7
STDEV	0.57/0.56	16.4/2.2	1.0/0.6
p-value		2.3E−9/5.9e−12	7.3e−6/2.5e−4
NSR vs. *	(One-tail)		

Table 8
Statistical analysis of CSRD approach against Pressure and Fixed benchmarks on high/low flow (Flow3/Flow4).

	NSR	*Fixed Time*	*PressLight*
Mean	141.4/135.1	217.3/161.4	144.1/136.1
STDEV	0.36/0.40	8.77/0.57	0.72/0.32
p-value		5.2e−10/1.3e−15	5.8e−8/4.7e−5
NSR vs. *	(One-tail)		

Table 9
Statistical Queue comparison of QSR trained with penalty process versus QSR without.

	QSR	QSR with PP
Mean	40.8	26.5
STDEV	20.61	12.38
p-value		0.001142

major advantage of our design approach is its simplicity to be implemented in real-life intersections since it requires a simple environment perception that uses few sensors for both state and reward formulae. Responding to research question (2), we verify the hypotheses (2) and (3) using the statistical analysis from Table 7 and Table 8. Using the results of the t-test, the null hypothesis H_0, which assumes that the proposed CSRD approach presents no improvements over the Fixed-Time approach and is not at least as high as the state-of-the art PressLight approach in all simulated experiments, is rejected with a very strong (p-value < 0.001) in the four experiment scenarios (see Table 7 and Table 8).

6.5.3. How an agent trained using penalty process performs against an agent trained without penalty process?

To evaluate the effect of adding a penalty process to our approach and verify the hypothesis (4), we trained the QSR agent along with the penalty type 1 in a flow configuration Flow1. When the episodic reward reaches a value inferior to some threshold it will penalize and feed the agent with a critical negative penalty value. This process notifies the agent to avoid such bad actions affecting not only the penalized current action but also a number of previous actions. Fig. 7 presents the queue length performance of a QSR agent with and without the penalty process. Clearly, adding the penalty process during the learning of the agent keeps it away from negative awards and as a result will affect the final performance of the agent. The statistical analysis in Table 9 shows that the null hypothesis H_0, assuming that the effect of adding penalty process presents no improvement, is rejected using t-test with a strong evidence (i.e., p-value < 0.01) at a significance level of 1%. This result holds up our assumption that the agent trained with penalty process performs at least as good as the agent trained without it. The idea of introducing penalty can be adopted not only in video games but also in other reinforcement learning frameworks such as control and robotic applications.

6.5.4. How does the selection of green duration affect the performance of the TSC agent?

Previously, the applied green duration is fixed for all phases at the intersection. However, the choice of green phase duration can indeed improve/worsen the behavior of the traffic signal control. We attempt to evaluate the effect of varying the green phase duration by training and testing the agent on multiple green durations of 10, 15, and 30 s long. In Fig. 8, one can observe how does adjusting the phase duration impact the performance of traffic signal control. We notice that small green phase duration can be insufficient for all queuing vehicles to cross the intersection. On the other side, having long green phase duration might be over sufficient for emptying the vehicles queue, where the green phase might be meaninglessly turn-on for no waiting vehicles. Accordingly, as a response to research question (4), it turns out

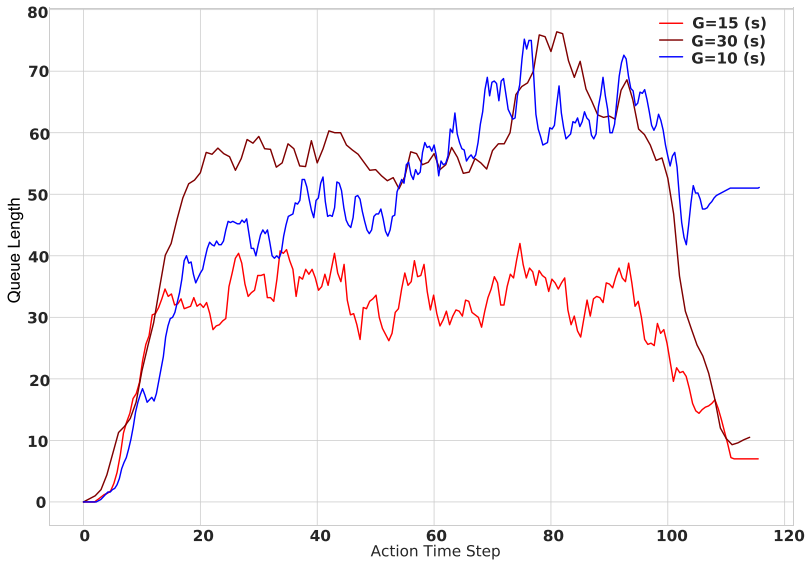


Fig. 8. Queue Length comparison of WSR on Flow1 with **different green phase durations** smoothed with 10 episodes window size (Lower curve is better).

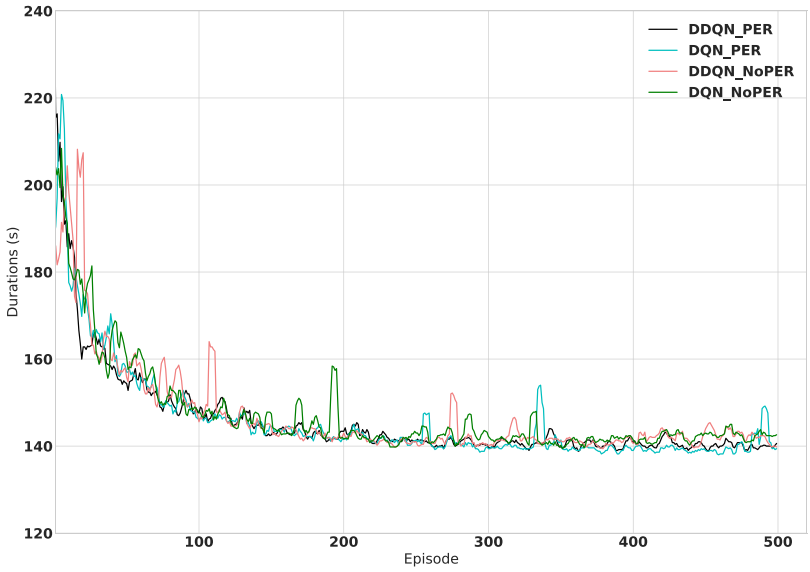


Fig. 9. Training curves of the conducted Ablation study: Double DQN with PER, DQN with PER, Double DQN without PER and DQN without PER respectively.

that properly choosing the right green phase duration affects the performance of the TSC and is essential when designing the traffic signal control agent.

6.6. Ablation study

In the ablation study in general, we conduct a set of experiments in which some components of the machine learning system are removed/replaced in order to measure the impact of these components on the performance of the system [40]. In our DRL system, the ablation study involves excluding respectively the use of Double DQN, the Prioritized Experience Replay memory, and both of the components. The results are reported in the global Fig. 9, and detailed in Fig. 10, Fig. 11 and Fig. 12 where the ablation of one of the two options or both shown in the following subsections.

6.6.1. Ablation of double DQN

The ablation of Double DQN from our DRL framework results in DQN with PER memory based system. We run the training with these ablation settings and the obtained results are depicted in Fig. 10. When compared to our choice (DDQN with PER), the ablation of DDQN resulted in a slightly slower convergence with very few fluctuations of the performance.

6.6.2. Ablation of PER

The ablation of Prioritized Experience Replay memory (PER) from our DRL system results in DDQN without PER memory (a simple replay memory is used instead). After executing the training based on the aforementioned settings, the learning curves are illustrated and shown in Fig. 11. The curve of DDQN without PER shows that the ablation of PER causes the DRL system to be slightly less stable, with a very similar convergence speed.

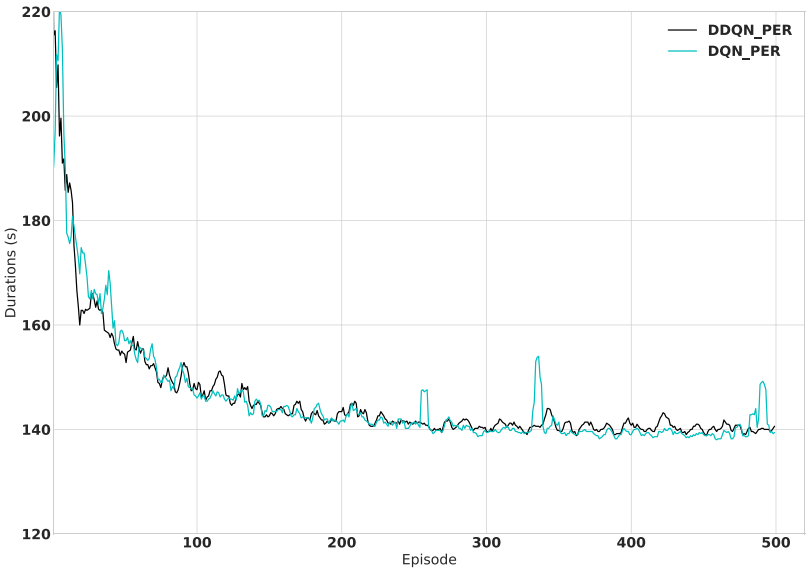


Fig. 10. Training curves of the conducted Ablation study: Effect of removing Double DQN.

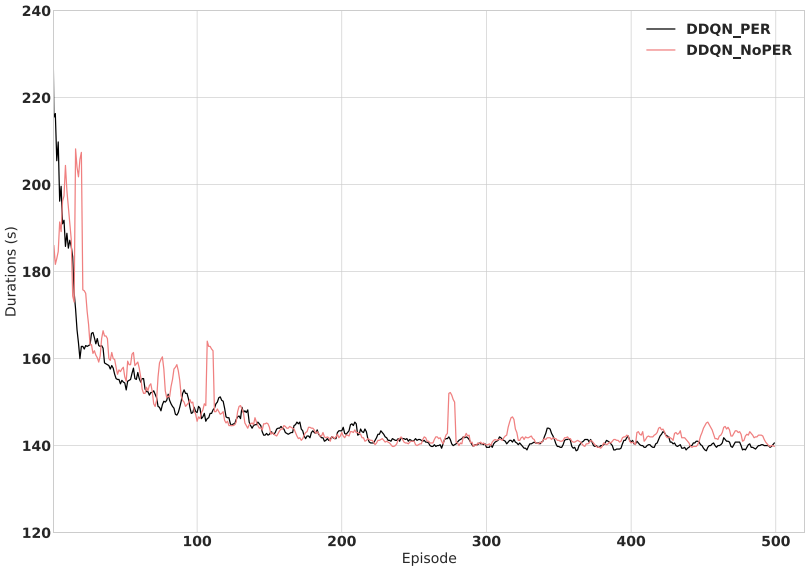


Fig. 11. Training curves of the conducted Ablation study: Effect of removing PER.

6.6.3. Ablation of both DDQN and PER

In this case, we exclude both of Double DQN and PER components from the DRL system. We run the simulated training with these new settings and the resulting training curves are plotted in Fig. 12. The impact of removing both DDQN and PER options from the DRL system is also similar to the previous ablations as it introduces few fluctuations of the performance causing a slight instability and a slight slowness of the convergence.

The ablation study shows that the adoption of the DDQN and PER options improve slightly the stability of learning. But it is still not easy to conclude the positive impact on performance as well as the speed of convergence. As a first finding, is that with non-ideal settings, striving for consistency is the main cause of the convergence improvement, however, we do not exclude that the use DDQN and PER can be at the origin of a limited improvement of the training stability on the top of consistency. Therefore use of DDQN and PER within our consistency based approach is still recommended.

7. Threats to validity

In this section, we discuss the potential threats to validity that can affect our experimental results. In the current work the threats to internal validity, is related to the extent to which we can be confident that striving for consistency among the state and reward definitions will improve the performance of traffic signal control. To make sure that this improvement cannot be explained by other factors, we proposed three pairs of state and reward definitions preserving consistency. the obtained results showed reasonably higher performance over the alternative approaches. With regard to construct validity of our study, we have discussed and selected a number of performance measures that reflect the efficiency of the traffic light performance measures. Indeed, we have used three different evaluation measures that are commonly used in the literature for evaluating the traffic signal control efficiency, namely (1) the average travel time [11], (2) the queue length [24], and (3) the waiting time [13]. Thus, we believe that the selected multiple performance measures represent

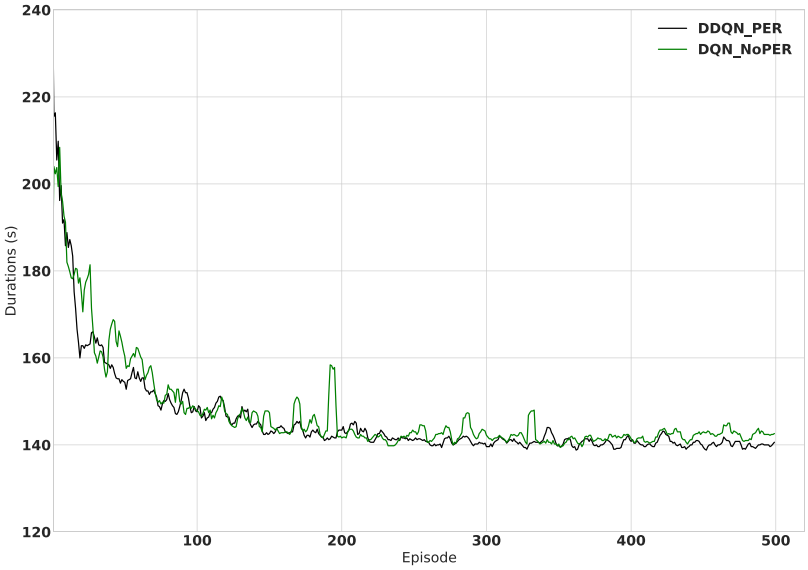


Fig. 12. Training curves of the conducted Ablation study: Effect of removing both DDQN and PER.

what would be different perceptions of traffic light signal performances, which makes negligible the threat to construct validity. Threats to external validity, mostly concerned with the generalizability of study treatments, conditions, and findings. Indeed our experiments are founded on a widely used open source tool, Simulation of Urban MObility (SUMO) simulator [36]. Together with synthetic traffic datasets, SUMO efficiently simulates typical intersection environment with possibility to cover a wide range of traffic signal configurations. Towards generalizing our findings, we have compared our approach to state of the art benchmarks with high conformity replication. We used the same experiment setting and implementation choices provided by Vidali et al. in [27] to replicate the DTSE-based state [11] with the corresponding DDQN-PER architecture. As for the PressLight agent [13] and LIT [24] agent, we also used the double deep Q-Network (DDQN) along with prioritized experience replay (PER) to ensure a fair comparison with our approach. Besides, our approach is using SUMO, and founder on DDQN-PER implementation for easing replication and comparison. With respect to conclusion threats to validity in our work, we are concerned at drawing the right conclusion about the relationship between the performance of our approach and those of the benchmarks. Indeed, in our empirical evaluation, we statistically analyzed the obtained results using the t-test statistical analysis which provided strong evidence for validating our ideas and assumptions. Hence, we believe that there is negligible threat to the validity of our conclusions.

8. Conclusions and future works

In this paper, we address the state and reward formulation in reinforcement learning based traffic signal control. We introduced an effective and practical Deep RL traffic signal control approach based on the idea of striving consistency among state and reward. Three variants of consistent dual definition for both state and reward are implemented. They are respectively based on, queue length, number of vehicles and waiting time (*i.e.*, QSR, NSR, and WSR). The proposed approach variants are evaluated on synthetic traffic flow and compared to other state-of-the-art approaches. The results show that our approach outperforms the benchmarks in all experiment instances. Moreover, a new mechanism of penalty is proposed to enhance our approach by mimicking the behavior of reinforcement learning in gaming environment where the agent is forced to avoid bad experiences and thus, the learning convergence as well as the performance are improved. A major advantage of our design approach is that it is straightforward and cost-effective to be installed in real world intersections compared with the benchmarks. One limitation of our traffic signal control approach is related to its restriction by the duration of the signal phase. As we discussed earlier, the duration of the green phase does indeed impact the performance of the traffic signals.

Future works would be directed towards including a novel agent architecture able to predict simultaneously the proper phase as well as its right duration. This can be achieved using both the discrete and continuous deep reinforcement learning algorithms in a hybrid architecture. Moreover, the experiments would be extended by testing the framework on multi-intersection scenarios and on traffic data collected from real intersections for instance. Another extension of our work will be concerned with defining metrics to assess the economical, energetic and even environmental gains resulting from application of our TSC approach.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported by the to Emirates Center for Mobility Research (ECMR) of the United Arab Emirates University (grant number 31R225).

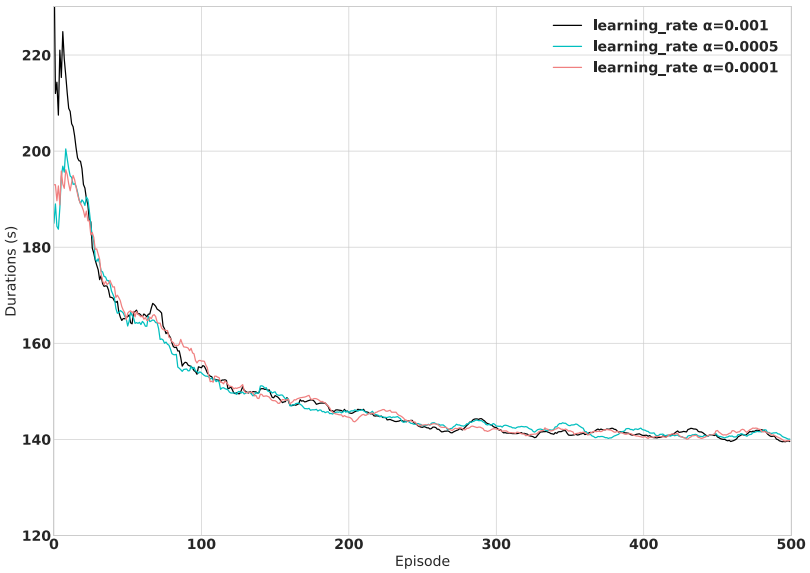


Fig. A.13. Travel time curves corresponds to different learning rates α during the learning process.

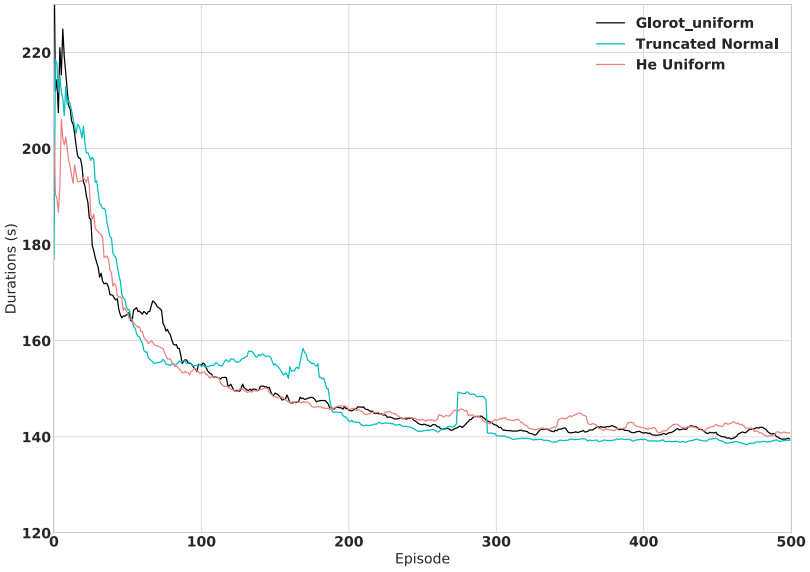


Fig. A.14. Travel time curves corresponds to the initialization methods Glorot_uniform, Truncated_Normal and He_Uniform during the training process.

Appendix. Training stability across different hyper-parameters

The parameters selection in the DRL framework has a great overall impact on the learning process and affects consequently the resulting performance of the agent [41]. The DRL framework parameters include: the environment settings, the action selection, the reward function, the objective function and the agent hyper-parameters. In the following we will focus on experimenting different agent hyper-parameters accompanied with the illustrating figures.

A.1. Effect of learning rate

The learning rate is a very important hyper-parameter that have to be taken into account when preparing the training process. it is a hyper-parameter that controls how much to change the model in response to the estimated error each time the model weights are updated [42]. When the learning rate is too small it can cause the model to get stuck at certain point, whereas a learning rate that is too large can cause the model to quickly converge to a sub-optimal solution. Fig. A.13 shows three training curves of our model where each curve is corresponded to one learning rate from the set $\{0.001, 0.0005, 0.0001\}$. It can be observed from the curve that there is not much of performance fluctuation when changing the learning rate, which leads to the fact that our approach is more stable with respect to the learning rate hyper-parameter.

A.2. Effect of initialization techniques

The selection of the weight initialization technique is another important factor that should be well taken when developing the deep learning model. The weight initialization role is to prevent the outputs of activation layer from exploding or vanishing gradients during the forward propagation which are both a real problem in may delay the convergence of the model if it converges

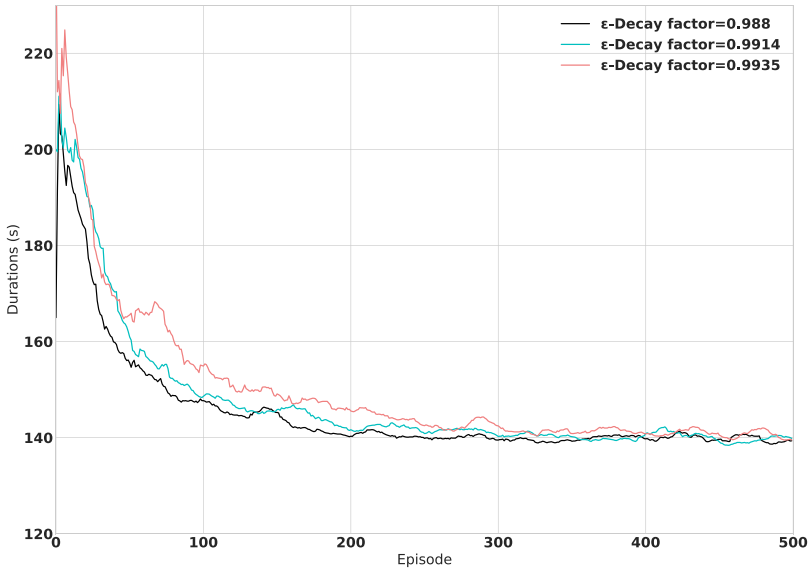


Fig. A.15. Travel time performance curves corresponds to different $\epsilon - decaying$ rates during the training process.

at all. One of the most common techniques in weight initialization are the Glorot-Uniform initialization [43] (also known as Xavier technique), He initialization [44] and Truncated initialization technique. These three techniques are use during the training process and are illustrated together in Fig. A.14. Notably, changing the initialization technique hyper-parameter causes a small fluctuation in the learning performance among the training curves of our DRL agent. This supports the stability and robustness of our approach with respect to the change of the initialization technique.

A.3. Effect of $\epsilon - decay$ rate

$\epsilon - decay$ rate parameter another hyper-parameter that takes control of the exploration policy. Tuning this parameter affects the exploration/exploitation policy of the decision-maker (the agent) during the learning process, as the agent follows the $\epsilon - greedy$ action selection policy. This policy balances between the exploration and exploitation by choosing an action between them randomly. A probability of ϵ leads the agent to take a random action and explore, while the $1 - \epsilon$ probability forces the agent to exploit what the agent have learned so far. During the learning process, ϵ value decays from the value of 1 (full exploration) to a very small value (exploitation) using a decaying rate r i.e. $\epsilon = 1 * r^{episode}$. Fig. A.15, illustrates three training curves where each curve corresponds to a $\epsilon - decaying$ rate. When changing the $\epsilon - decaying$ rate value, there is a noticeably small amount of performance fluctuations during the training process of the agent, which can be an indication to the stability of our approach with respect to the change of this hyper-parameter.

References

[1] INRIX scoreboard 2018, PRESS RELEASES, 2018, URL https://inrix.com/press-releases/scorecard-2018-us/.
[2] L.-H. Xu, X.-H. Xia, Q. Luo, The Study of Reinforcement Learning for Traffic Self-Adaptive Control under Multiagent Markov Game Environment, 2013, Hindawi Publishing Corporation, 962869, http://dx.doi.org/10.1155/2013/962869.
[3] H. Wei, G. Zheng, V. Gayah, Z. Li, A survey on traffic signal control methods, 2019, arXiv:1904.08117.
[4] F. Dion, H. Rakha, Y. Kang, Comparison OF DELAY ESTIMATES AT UNDER-SATURATED AND OVER-saturated PRE-TIMED signalized intersections, Transp. Res. B 38 (2004) 99–122.
[5] C. Gershenson, Self-organizing traffic lights, 2004, arXiv:nlin/0411066.
[6] P. Varaiya, The max-pressure controller for arbitrary networks of signalized intersections, 2013.
[7] C.J.C.H. Watkins, P. Dayan, Q-learning, 8 (3) 279–292, http://dx.doi.org/10.1007/BF00992698.
[8] P. Balaji, Urban traffic signal control using reinforcement learning agents, IET Intell. Transp. Syst. 4 (2010) 177–188, (11). URL https://digital-library.theiet.org/content/journals/10.1049/iet-its.2009.0096.
[9] I. Arel, C. Liu, T. Urbanik, A.G. Kohls, Reinforcement learning-based multi-agent system for network traffic signal control, IET Intell. Transp. Syst. 4 (2) (2010) 128–135, http://dx.doi.org/10.1049/iet-its.2009.0070.
[10] J. Gao, Y. Shen, J. Liu, M. Ito, N. Shiratori, Adaptive traffic signal control: Deep reinforcement learning algorithm with experience replay and target network, 2017, arXiv:1705.02755.
[11] W. Genders, S. Razavi, Using a deep reinforcement learning agent for traffic signal control, 2016, arXiv:1611.01142.
[12] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis, Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533, http://dx.doi.org/10.1038/nature14236.
[13] H. Wei, C. Chen, G. Zheng, K. Wu, V. Gayah, K. Xu, Z. Li, PressLight: Learning max pressure control to coordinate traffic signals in arterial network, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 1290–1298.
[14] K. Shao, Z. Tang, Y. Zhu, N. Li, D. Zhao, A survey of deep reinforcement learning in video games, 2019, http://dx.doi.org/10.48550/ARXIV.1912.10944, URL https://arxiv.org/abs/1912.10944.
[15] A. Perera, P. Kamalaruban, Applications of reinforcement learning in energy systems, Renew. Sustain. Energy Rev. 137 (2021) 110618, http://dx.doi.org/10.1016/j.rser.2020.110618, URL https://www.sciencedirect.com/science/article/pii/S1364032120309023.
[16] X. Lu, L. Xiao, T. Xu, Y. Zhao, Y. Tang, W. Zhuang, Reinforcement learning based PHY authentication for VANETs, IEEE Trans. Veh. Technol. 69 (3) (2020) 3068–3079, http://dx.doi.org/10.1109/TVT.2020.2967026.
[17] L. Xiao, X. Lu, T. Xu, W. Zhuang, H. Dai, Reinforcement learning-based physical-layer authentication for controller area networks, IEEE Trans. Inf. Forensics Secur. 16 (2021) 2535–2547, http://dx.doi.org/10.1109/TIFS.2021.3056206.
[18] Y. Lin, X. Dai, L. Li, F.-Y. Wang, An efficient deep reinforcement learning model for urban traffic control, 2018, arXiv:1808.01876.
[19] M. Gregurić, M. Vujić, C. Alexopoulos, M. Miletić, Application of deep reinforcement learning in traffic signal control: An overview and impact of open traffic data, Appl. Sci. 10 (11) (2020) http://dx.doi.org/10.3390/app10114011, URL https://www.mdpi.com/2076-3417/10/11/4011.

[20] S. Bouktif, A. Cheniki, A. Ouni, Traffic signal control using hybrid action space deep reinforcement learning, Sensors 21 (7) (2021) 2302.
[21] S. Bouktif, A. Cheniki, A. Ouni, H. El-Sayed, Traffic signal control based on deep reinforcement learning with simplified state and reward definitions, in: 2021 4th International Conference on Artificial Intelligence and Big Data, ICAIBD, IEEE, 2021, pp. 253–260.
[22] 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, http://dx.doi.org/10.1016/j.trip.2019.100020, URL http://www.sciencedirect.com/science/article/pii/S259019821930020X.
[23] X. Liang, X. Du, G. Wang, Z. Han, Deep reinforcement learning for traffic light control in vehicular networks, 2018, arXiv:1803.11115.
[24] G. Zheng, X. Zang, N. Xu, H. Wei, Z. Yu, V. Gayah, K. Xu, Z. Li, Diagnosing reinforcement learning for traffic signal control, 2019, arXiv:1905.04716.
[25] H. Wei, G. Zheng, H. Yao, Z. Li, IntelliLight: A reinforcement learning approach for intelligent traffic light control, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '18, Association for Computing Machinery, New York, NY, USA, 2018, pp. 2496–2505, http://dx.doi.org/10.1145/3219819.3220096.
[26] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, The MIT Press Cambridge, Massachusetts London, London, England, 2018.
[27] A. Vidali, L. Crociani, G. Vizzari, S. Bandini, A deep reinforcement learning approach to adaptive traffic lights management, in: WOA, 2019.
[28] N. Casas, Deep deterministic policy gradient for urban traffic light control, 2017, arXiv:1703.09035.
[29] J. Gao, Y. Shen, J. Liu, M. Ito, N. Shiratori, Adaptive traffic signal control: Deep reinforcement learning algorithm with experience replay and target network, 2017, arXiv:1705.02755.
[30] Y. Wang, T. Xu, X. Niu, C. Tan, E. Chen, H. Xiong, STMARL: A spatio-temporal multi-agent reinforcement learning approach for cooperative traffic light control, 2019, arXiv:1908.10577.
[31] C. Szepesvári, Algorithms for reinforcement learning, Synth. Lect. Artif. Intell. Mach. Learn. 4 (1) (2010) 1–103, http://dx.doi.org/10.2200/S00268ED1V01Y201005AIM009, arXiv:https://doi.org/10.2200/S00268ED1V01Y201005AIM009.
[32] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, Playing atari with deep reinforcement learning, 2013, arXiv:1312.5602.
[33] H. van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double Q-learning, 2015, arXiv:1509.06461.
[34] Why vehicles are teleporting, SUMO, 2020, URL https://sumo.dlr.de/docs/Simulation/Why_Vehicles_are_teleporting.html. (Accessed 22-August-2020).
[35] T. Schaul, J. Quan, I. Antonoglou, D. Silver, Prioritized experience replay, 2015, arXiv:1511.05952.
[36] M. Behrisch, L. Bieker, J. Erdmann, D. Krajzewicz, SUMO - Simulation of urban mobility: An overview, in: In SIMUL 2011, the Third International Conference on Advances in System Simulation, 2011, pp. 63–68.
[37] Speed limits by country — Wikipedia, The Free Encyclopedia, 2020, URL https://en.wikipedia.org/wiki/Speed_limits_by_country. (Accessed 22-August-2020).
[38] T. Tieleman, G. Hinton, Lecture 6.5 - rmsprop: Divide the gradient by a running average of its recent magnitude, 2012.
[39] R. Gordon, W. Tighe, Traffic Control Systems Handbook (2005 edition), 2005.
[40] Running an Ablation Study , 2022, https://pykeen.readthedocs.io/en/stable/tutorial/running_ablation.html. (Accessed: 6 September 2022).
[41] N. Ashraf, R. Mostafa, R. Sakr, M. Rashad, Optimizing hyperparameters of deep reinforcement learning for autonomous driving based on whale optimization algorithm, PLoS One 16 (6) (2021) e0252754, http://dx.doi.org/10.1371/journal.pone.0252754.
[42] Understand the Impact of Learning Rate on Neural Network Performance, 2022, https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-eep-learning-neural-networks/. (Accessed: 31 August 2022).
[43] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: AISTATS, 2010.
[44] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification, 2015, CoRR abs/1502.01852. arXiv:1502.01852.