Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

Research paper

Parameterized-action based deep reinforcement learning for intelligent traffic signal control

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

[a] College of Information Technology, United Arab Emirates University, Al Ain, United Arab Emirates
[b] Institute of Electrical and Electronics Engineering, University of Boumerdes, Algeria
[c] École de Technologie Supérieure, University of Quebec, Montreal, Canada

A R T I C L E   I N F O

Dataset link: https://github.com/abderraouf2che/Hybrid-Deep-RL-Traffic-Signal-Control

Keywords:
Reinforcement learning
Parameterized action space
Traffic signal control
Traffic optimization

A B S T R A C T

Traffic Signal Control (TSC) is a crucial component in Intelligent Transportation Systems (ITS) for optimizing traffic flow. Deep Reinforcement Learning (DRL) techniques have emerged as leading approaches for TSC due to their promising performance. Most existing DRL-based approaches typically use discrete action spaces to predict the next action phase, without specifying the signal duration. In contrast, some studies employ continuous action spaces to determine signal phase timing within a fixed light cycle. To address the limitations of both approaches, we propose a flexible framework that predicts both the appropriate traffic light phase along with its associated duration. Our approach utilizes a Parameterized-action based deep reinforcement learning architecture to handle the combination of discrete-continuous actions. We evaluate our method using the Simulation of Urban MObility (SUMO) environment, comparing its efficiency against state-of-the-art techniques. Results demonstrate that our approach significantly outperforms traditional and learning-based methods.

1. Introduction

The increasing population size and the growing number of vehicles in urban areas have led to a substantial rise in traffic volume, resulting in severe traffic congestion. This congestion has had significantly undesirable impacts, including prolonged travel time delays and considerable economic losses each year (INRIX Scoreboard, 2022). One of the primary objectives of Intelligent Transportation Systems (ITS) is to alleviate congestion and optimize traffic flow within cities. Within this framework, Traffic Signal Control (TSC) systems play a critical role in managing traffic flow at signalized intersections (Haydari and Yilmaz, 2022). Numerous researchers have proposed various architectures to enhance traffic flow and reduce congestion by developing more intelligent Traffic Signal Control (TSC) systems (Bouktif et al., 2023; Genders and Razavi, 2016; Vidali et al., 2019). These approaches often utilize optimization techniques such as machine learning frameworks and meta-heuristic algorithms. In the scope of machine learning, TSC systems primarily rely on the Reinforcement Learning (RL) and Deep Reinforcement Learning (DRL) decision making techniques to optimize the traffic flow and to reduce congestion, in contrast to traditional TSCs controlling methods (Casas, 2017; Kolat et al., 2023; Khamis and Gomaa, 2014). One significant advantage of RL techniques is their flexibility and ability to handle dynamic and non-uniform traffic flow patterns, unlike manually designed standard methods. Indeed, DRL-based traffic signal control systems have been shown to outperform traditional TSC methods (Kolat et al., 2023; Rasheed et al., 2020; Wei et al., 2019). In DRL-based TSC approaches, the agent perceives the surrounding traffic environment and makes decision based on either discrete or continuous action spaces. In the discrete approach, mainly using Deep Q-networks (DQN) (Mnih et al., 2013) and its extensions, the action space includes two options: the agent either maintain the current traffic signal phase of TSC or transitions to the next signal phase in a predefined cycle of phases (Liu et al., 2023). In a more flexible action space, the agent can choose any specific phase from the set of phases without adhering to the sequence order (Kolat et al., 2023). In the continuous approach, other research proposals employ the continuous DRL architecture (e.g., Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2019)) where the agent's actions are continuous. In this strategy, the agent predicts the next phase's duration with a fixed sequence of phases (Casas, 2017). Indeed, forecasting the subsequent phase without determining its duration, or predicting the duration of the next phase within a fixed sequence is insufficient for optimal traffic control. Hence, being motivated by this research gap, we propose leveraging a hybrid

* Corresponding author.
E-mail addresses: salahb@uaeu.ac.ae (S. Bouktif), abderraouf2cheniki@gmail.com (A. Cheniki), Ali.Ouni@etsmtl.ca (A. Ouni), helsayed@uaeu.ac.ae (H. El-Sayed).

https://doi.org/10.1016/j.engappai.2025.111422
Received 27 August 2024; Received in revised form 6 January 2025; Accepted 5 June 2025
Available online 7 July 2025

deep reinforcement learning (DRL) approach that combines discrete and continuous architectures to further optimize traffic signal control performance. This hybrid DRL approach allows to concurrently predict the forthcoming Traffic Signal Control (TSC) phase and its associated duration. This architecture belongs to the domain of the parameterized action space Deep Reinforcement Learning (DRL); specifically, DRL with parameterized action space (Masson et al., 2015). In particular, we use a variant of this architecture known as the Parameterized-DQN, or P-DQN (Xiong et al., 2018). The P-DQN framework, hence, predicts the optimal traffic phase while simultaneously determining its corresponding duration, providing a cohesive solution for traffic signal control optimization. We experimentally demonstrate the effectiveness of our approach using the SUMO simulator on the proposed framework. The proposed framework, along with the benchmark approaches are evaluated and compared based on standard metrics for Traffic Signal Control (TSC) evaluation, namely, the average travel time, the queue length and the average vehicular waiting time (Liu et al., 2023; Kolat et al., 2023). The results demonstrate that our framework remarkably enhances TSC performance compared to the benchmarks. The contributions of this research are summarized as follows:

- We present an innovative parameterized action space framework for traffic signal control, enabling the simultaneous optimization of traffic phase selection and its duration. This approach addresses a critical limitation in current methodologies by integrating discrete and continuous action spaces, offering greater flexibility and control in traffic management.
- We evaluate our framework under varying and dynamic traffic conditions, including non-uniform and dynamic scenarios.
- We compare the proposed method against traditional, learning-based, and meta-heuristic approaches (e.g., Fixed-Time, Discrete, Continuous, GA, PSO) using multiple evaluation metrics such as average travel time (ATT), average waiting time (AWT), and queue length (QL). Results consistently demonstrate superior performance across diverse traffic scenarios.

The structure of this paper is organized as follows: Section 2 provides a review of related literature. Section 3 outlines the theoretical background and preliminaries. Section 5 describes our proposed methodology. Section 6 presents the empirical validation of our approach, including a discussion of the results compared to state-of-the-art approaches. Finally, Section 8 concludes the paper and suggests directions for future research.

2. Related work

Recent research efforts in the transportation field have increasingly focused on the application of artificial intelligence techniques to address various challenges in transportation systems. Notably, these techniques include: (*i*) deep reinforcement learning, and (*ii*) meta-heuristic algorithms.

2.1. Deep reinforcement learning for TSC

Typically, DRL-based traffic signal control systems perceive the traffic conditions at an intersection to learn optimal control policy and determine the suitable traffic signal phase and its duration. Research on DRL-based traffic signal control primarily varies across four key aspects: state representation, reward design, action selection, and agent's architecture. In this study, we focus specifically on analyzing action selection and agent architecture within the DRL-based framework.

2.1.1. Action space selection

In reinforcement learning, the agent observes the current state of the environment and takes the optimal action leading to the desired behavior. Specifically, in traffic light control, action selection varies according to the desired application. For instance, the action space can be formulated as a binary action, where the agent decides whether to maintain the current phase of traffic lights or to advance to the next phase (Liu et al., 2023). A more flexible approach allows the agent to select the next appropriate phase from a list of permitted phases at each time step (Kolat et al., 2023). A third form of action space belongs to the continuous domain, where the agent allocates time durations to each phase within the total cycle (Casas, 2017). Beyond the action spaces found in the literature, in this work, we propose an action space that combines discrete and continuous domains, allowing the agent to select both the next phase of the traffic light and its accompanying duration simultaneously (see Table 1).

2.1.2. Agent's architecture

A crucial component in deep reinforcement learning is the network architecture chosen for the agent. This network takes the environment's state as input and predicts the appropriate action accordingly. During the training process, the network learns the optimal action policy through interaction with the environment. Various network architectures have been developed depending on the type of action space selected.

In a discrete action space, the Deep Q-Network (DQN) (Mnih et al., 2013) is the most common agent architecture. It maps the state space input to Q-values of the target actions. However, DQN can suffer from overestimation bias, leading to suboptimal learning. To address this, Double-DQN (van Hasselt et al., 2015) was introduced, which uses two separate networks – one to select actions and another to evaluate them – thereby reducing overestimation. Another extension, Dueling-DQN, incorporates two separate streams in the network: one for estimating the state value and another for estimating the advantage of each action, improving learning in environments with many similar-valued actions. Additionally, the Normalized Advantage Function (NAF) (Gu et al., 2016) extends DQN to continuous domains by approximating the Q-value function using quadratic advantage terms.

For continuous action spaces, Actor-Critic-based architectures are commonly used. For instance, Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2019) combines an Actor network to predict continuous actions and a Critic network to evaluate the Q-values of state–action pairs. Soft Actor–Critic (SAC) (Haarnoja et al., 2018) improves upon DDPG by incorporating an entropy term in the objective function, encouraging exploration and resulting in more robust learning.

However, real-world application scenarios often present more complex action spaces where discrete and continuous spaces coexist. Recent research has explored the potential of composite action spaces using DRL with parameterized action space, which consists of a set of discrete actions, each parameterized by a continuous parameter. For instance, Hausknecht and Stone (2016) proposed an extension of DDPG for parameterized action spaces, enabling agents to select discrete actions while simultaneously predicting associated continuous parameters.

This approach was further improved by Xiong et al. through the development of the Parameterized Deep Q-Network (P-DQN) (Xiong et al., 2018), which combines the benefits of DQN and DDPG. However, P-DQN faces challenges related to joint-action representation. To address this, Bester et al. introduced the Multi-Pass DQN (MP-DQN) (Bester et al., 2019), which decouples the discrete actions and their continuous parameters by performing separate forward passes for each action-parameter pair, leading to better training stability and improved performance. Table 2 summarizes the key attributes, advantages, and limitations of these architectures.

Table 1
Comparison of action space selection techniques in DRL for traffic signal control.

Action space	Description	Advantages	Limitations	References
Binary	Maintain or switch phases	Simple and computationally efficient	Limited flexibility	Liu et al. (2023)
Discrete	Select next phase from predefined options	Suitable for static applications	May lack granularity for dynamic needs	Kolat et al. (2023)
Continuous	Allocate time duration for each phase	High flexibility for dynamic scenarios	Computationally expensive	Casas (2017)
Hybrid	Combines discrete and continuous action spaces	Combines flexibility with feasibility	Requires more complex architecture	Hausknecht and Stone (2016), Xiong et al. (2018) and Bester et al. (2019)

Table 2
Comparison of DRL agent architectures for TSC.

Attribute	Deep Q-Network (DQN)	Double-DQN	DDPG	P-DQN
Action space	Discrete	Discrete	Continuous	Hybrid
Advantages	Simple and widely adopted	Reduces overestimation of Q-values	Works for continuous action spaces	Handles parameterized actions effectively
Limitations	Not suitable for continuous action spaces	Similar limitations as DQN	Susceptible to instability during training	Computational complexity
References	Mnih et al. (2013)	van Hasselt et al. (2015)	Lillicrap et al. (2019)	Xiong et al. (2018)

Table 3
Comparison of meta-heuristic algorithms for TSC.

	GA	PSO	SA	ABC
Key feature	Evolutionary optimization	Swarm intelligence	Probabilistic hill climbing	Collective behavior of bees
Advantages	Robust to local minima	Fast convergence	Effective for discrete optimization	Balances exploration and exploitation
Limitations	Computationally expensive	Susceptible to premature convergence	Slow convergence	Sensitive to parameter tuning
References	Tan et al. (2016)	García-Nieto et al. (2012)	Oda et al. (1997)	Dell'Orco et al. (2014)

2.2. Meta-heuristic algorithms for TSC

Meta-heuristic algorithms are approximate search techniques developed for solving complex optimization problems (Osman and Kelly, 1996). The most common meta-heuristic techniques include the Genetic Algorithm (GA) (Holland, 1992), Particle Swarm Optimization (PSO) (Eberhart and Kennedy, 1995), Simulated Annealing (SA) (Kirkpatrick et al., 1983), and Artificial Bee Colony (ABC) optimization algorithms (Karaboga and Basturk, 2007). Over the past three decades, meta-heuristic algorithms have been widely applied to optimize transportation systems, including traffic signal control at intersections (Hussain et al., 2019; Shi et al., 2021). For example, Tan et al. (2016) developed a GA-based approach to optimize signal green timing cycle ratios at a crossed intersection under oversaturated conditions. Their results demonstrate improved signal timing and minimized average delay at the intersection as optimized by the GA algorithm. García-Nieto et al. (2012) proposed a PSO approach to find effective cycle programs for traffic lights. The solutions obtained by PSO were evaluated in two large and heterogeneous metropolitan areas and compared with cycle programs predefined by experts. The comparison results show significant benefits in terms of the number of vehicles reaching their destinations on time and the average travel time. Oda et al. (1997) adopted an SA-based approach to optimize TSC behavior at intersections, resulting in superior performance compared to the conventional method. Finally, Dell'Orco et al. (2014) proposed the ABC algorithm for finding the optimal setting of traffic signals in coordinated signalized networks. Results, which were compared with two other meta-heuristic algorithms (GA and Hill Climbing), showed a slightly better performance (see Table 3).

3. Background and preliminaries

This section provides the necessary background and prerequisites to better understand our contribution.

3.1. Reinforcement learning

Reinforcement learning decision making problems are generally formulated as a Markovian Decision Process (MDP), characterized by the terms $\langle \mathcal{S}, \mathcal{P}, \mathcal{A}, \mathcal{R}, \gamma \rangle$. The term $\mathcal{S}$ stands for the state space, $\mathcal{P}$ is the Markov probability of transition, $\mathcal{A}$ is the action space, $\mathcal{R}$ is the reward and γ is the discount factor. The agent interacts with the environment through a "trial and error" kind of learning. At each time step t, the agent produces an action a_t based on its action-selection policy π and according to the perceived state of the environment s_t. Afterwards, the agent obtains an immediate reward r_t evaluating the performed action, and simultaneously, observes the next state s_{t+1} for upcoming decisions. In the learning process, the main objective of the agent is to learn a policy π so to maximize the cumulative discounted reward $G_t = \sum_{k=0}^{n} \gamma^k \mathcal{R}_{t+k}$ by taking optimal actions each time step t.

3.1.1. RL for discrete action space

For a RL with discrete-action space, Q-learning (Watkins and Dayan, 1992) is commonly the most used method to derive the optimal action-taking policy π of the agent. Q-learning is a value-based RL algorithm that employs a Q-function defined as:

$$Q^{\pi}(s,a) = \mathbb{E}[G_t | s, a, \pi]. \tag{1}$$

The Q-values evaluate how good are the state–action pairs and are computed recursively using dynamic programming to solve the following equation:

$$Q^{\pi}(s,a) = \mathbb{E}_{s_{t+1}}[R_t + \gamma \mathbb{E}_{a_{t+1} \sim \pi(s_{t+1})}[Q^{\pi}(s_{t+1}, a_{t+1})] | s, a, \pi]. \tag{2}$$

The optimal policy π^* can be found from $Q^*(s,a) = \max_\pi Q^\pi(s,a)$ by computing the optimal action a^* for each state s, where $Q^*(s,a)$ obeys the Bellman optimality equation:

$$Q^*(s,a) = \mathbb{E}_{s_{t+1}}[R_t + \gamma \max_{a_{t+1} \in \mathcal{A}} Q^*(s_{t+1}, a_{t+1}) | s, a], \tag{3}$$

$$a^* = \underset{a \in \mathcal{A}}{\operatorname{argmax}} Q^*(s,a). \tag{4}$$

We also define the state-value function $V(s)$ which evaluates the value of being in a given state s_t as following

$$V^\pi(s) = \mathbb{E}_{a\sim\pi(s)}[Q^\pi(s,a)], \text{ and } V^*(s) = \max_a V^\pi(s). \tag{5}$$

In the finite state space $\mathcal{S}$, the values of $Q(s,a)$ are stored in a table during the training process where the Q-function is updated iteratively. However, when the state space dimension is considerably large, a tabular storage is no longer feasible. Therefore, to solve the dimensionality-complexity problem, Deep Q-Networks (DQN) (Mnih et al., 2013) is adopted to approximate the Q-function using neural networks where the term $Q(s,a;\theta)$ is used, and θ refers to the weights or parameters of the neural network. In Deep Q-Networks, the objective is to learn a Q-network which precisely predicts $Q(s,a)$ values. Thus, at the tth iteration, the learning algorithm's goal is to minimize the following loss function:

$$\mathcal{L}_t(\theta) \ = \ \mathbb{E}[(Q(s_t,a_t;\theta) - R_t + \gamma \max_{a_{t+1}\in\mathcal{A}} Q^\pi(s_{t+1},a_{t+1}))^2]. \tag{6}$$

where $R_t + \gamma \max_{a_{t+1}\in\mathcal{A}} Q^\pi(s_{t+1},a_{t+1})$ denoted the temporal difference target y_t.

3.1.2. RL for continuous action space

In continuous action space, maximizing over a continuous (infinite) set of actions $\mathcal{A}$ is computationally intractable, which makes directly applying value-based RL algorithms ineffective for continuous space. Thus the policy-based RL turns out to be useful in continuous action space by considering the second policy type (i.e., the deterministic policy $\pi(a|s)$). Deterministic Policy Gradient (DPG) (Silver et al., 2014) is known as the standard algorithm for the policy-based RL. It follows the actor-critic approach by having a parameterized actor function μ_θ that identifies the current policy by deterministically mapping states into specific actions $\mu : \mathcal{S} \rightarrow \mathcal{A}$ (Lillicrap et al., 2019). The critic Q-function makes use of the recursive relationship similar to Bellman equation (3) with the difference that the inner expectation is replaced by the state–action mapping μ_θ:

$$Q^{\mu_\theta}(s,a) = \mathbb{E}_{s_{t+1}}[R_t + \gamma[Q^{\mu_\theta}(s_{t+1},\mu_\theta(s_{t+1}))]|s,a]. \tag{7}$$

The actor is updated by the gradient of the policy's performance J using the DPG policy gradient theorem which states that

$$\nabla_\theta J(\mu_\theta) = \underset{s\sim\rho^{\mu_\theta}}{\mathbb{E}}\left[\nabla_\theta \mu_\theta(s)\nabla_a Q^{\mu_\theta}(s,a)|_{a=\mu_\theta(s)}\right]. \tag{8}$$

A variant of the DPG, namely the Deep DPG (DDPG), is proposed by Lillicrap et al. (2019). DDPG provides a modification of the DPG by introducing a neural network as function approximators to learn over large state and action space.

3.1.3. RL for parameterized action space

In hybrid RL, the action space is a hybridization of both discrete and continuous action spaces. More precisely, we will consider the parameterized action space type of hybrid RL. The parameterized action space is composed of a set of discrete actions, each discrete action is attached (parameterized) by a continuous parameter value (Xiong et al., 2018). In this paper, we will formulate our problem as an MDP with a parameterized action space as proposed in P-DQN architecture (Xiong et al., 2018). The action space $\mathcal{A}$ is given as:

$$\mathcal{A} = \{(k,x_k)|x_k \in \mathcal{X}_k \text{for all } k \in [K]\}, \tag{9}$$

where (k,x_k) stands for the discrete action and continuous parameter respectively, following a hierarchical structure of actions. Thus, we have firstly a main discrete action selected from the set K ($k \in K = \{1,\ldots,K\}$), and a continuous sub-action $x_k \in \mathcal{X}_k$ selected from the action space $\mathcal{X}_k$. The Q-function notation becomes $Q(s,k,x_k)$, where $s \in \mathcal{S}$, $a \in \mathcal{A}$, $k \in [K]$ and $x_k \in \mathcal{X}_k$. To compute the continuous parameter x_k, we use a mapping function $x_k^Q : \mathcal{S} \rightarrow \mathcal{X}_k$ as in DDPG (Lillicrap et al., 2019) which maps the state space to specific continuous action parameters. Hence, the Q-function becomes:

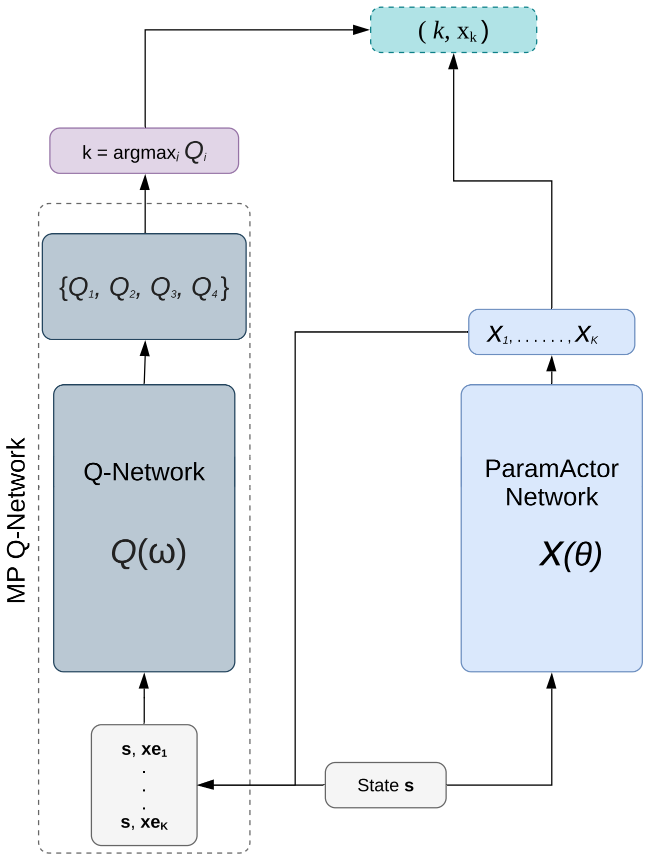

Fig. 1. Structure of MP-DQN architecture.

$$Q\left(s_t,k_t,x_{k_t}\right) = \mathbb{E}_{s_{t+1}}[R_t + \gamma \max_{k\in[K]} Q(s_{t+1},k,x_k^Q(s_{t+1}))|s_t = s]. \tag{10}$$

In a similar way as in DQN and DDPG, we use deep neural networks to approximate both discrete and continuous mapping functions $Q(s,k,x_k)$ and x_k^Q. Therefore, the mapping $Q(s,k,x_k)$ is approximated by $Q(s,k,x_k;\boldsymbol{\omega})$ with weights $\boldsymbol{\omega}$, and the mapping x_k^Q is approximated by $x_k(\cdot;\boldsymbol{\theta})$ with weights $\boldsymbol{\theta}$. Similar to DQN, the target value y_t is given by:

$$y_t = R_t + \gamma \max_{k\in[K]} Q(s_{t+1},k,x_k(s_{t+1};\boldsymbol{\theta}_t);\boldsymbol{\omega}_t), \tag{11}$$

Finally, the parameters $\boldsymbol{\omega}$ and the $\boldsymbol{\theta}$ respectively updated using the following loss functions:

$$\ell_t^Q(\boldsymbol{\omega}) = \frac{1}{2}[Q(s_t,k,x_k);\boldsymbol{\omega}_t - y_t]^2, \quad \text{and} \tag{12}$$

$$\ell_t^{\boldsymbol{\Theta}}(\boldsymbol{\theta}) = -\sum_{k=1}^{K} Q(s_t,k,x_k(s_t;\boldsymbol{\theta});\boldsymbol{\omega}_t)$$

Furthermore, as an extension to P-DQN architecture, Bester et al. (2019) proposed a modification of the P-DQN architecture, called Multi-Pass DQN (MP-DQN). MP-DQN comes to fix the joint-action issue found in P-DQN. Loosely speaking, the joint action issue appears since the Q-network receives as an input all action-parameters $(x_1,\ldots,x_K)$ of all discrete actions together, instead of having a separate each action-parameter x_k with its related discrete action k. The MP-DQN architecture solves this issue where each action-parameter x_k will be associated with the related discrete action k. The MP-DQN performs one forward pass per action k, where the input of the Q-network includes the state s and the action-parameter vector $\boldsymbol{xe}_k$. The vector $\boldsymbol{xe}_k = (0,\ldots,0,x_k,0,\ldots,0)$ stands for the standard basis for dimension k

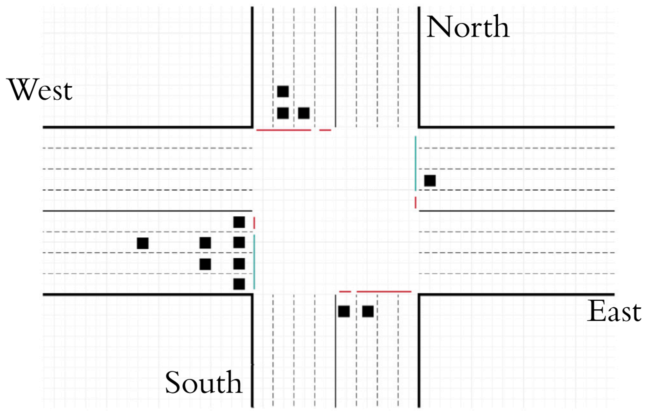

Fig. 2. Four-lane four-armed traffic intersection.

(see Fig. 1 where the input state s is spread over the action-parameters vectors $\boldsymbol{xe}_k$ then the MP Q-network performs multi passes over these tuples $(s, \boldsymbol{xe}_k)$, once per each. This MP-DQN trick fixes the P-DQN issue and lets Q_k depending only on its associated x_k, where:

$$Q(s, k, \boldsymbol{xe}_k) \approxeq Q(s, k, x_k). \tag{13}$$

4. Problem definition

To formulate our problem, we consider the following elements.

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 (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. Fig. 2 shows the environment structure with phase East-West activated.

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), takes an action a_t according to its policy and receives an immediate reward $\mathcal{R}_t$ at each time step t.

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.

$$\text{Travel Time} = t_{j\_start} - t_{j\_end} \tag{14}$$

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.

Problem Definition *Given a traffic signal intersection as the environment E, the agent G receives a state $s \in \mathcal{S}$, performs a joint action $a \in \mathcal{A}$* and collects rewards R from the environment. The goal of the agent is to decide the optimal joint action $a = (P, d_P)$ for both phase P and its associated duration d_P in order to minimize the average vehicular travel time by maximizing the expected return, where the discounted return is given as follows (see Table 4):

$$\sum_{k=0}^{n} \gamma^k \mathcal{R}_{t+k} \tag{15}$$

Table 4
Notations.

Notation	Description
s	State
a	Action
R	Reward
G	Agent
E	Environment
L	Number of lanes
P	Phase of signals
d_P	duration of phase P
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

5. Parameterized action based deep reinforcement learning for TSC

Here in this section, we describe our approach which adopts the Parameterized action space Deep Reinforcement Learning. This framework allows to predict the proper phase P along with its associated duration d_P.

Framework Overview. Fig. 3 illustrates the essential components of our framework to control the traffic lights at signalized intersections. Initially, the agent perceives the state of the intersection at time-step t. According to the perceived state and based on its policy π, the agent predicts a joint action (P, d_P), where P is the selected phases from the set of phases, and d_P stands for the corresponding phase duration. Consequently, at the time-step $t+1$, the environment rewards the agent by a reward signal $\mathcal{R}_t$ evaluating the performed joint action $a = (P, d_P)$. Finally, the agent receives the state s_{t+1} for the next action selection procedure. In the learning process, the agent stores the resulting experiences $\langle s_t, a_t = (P_t, d_{P_t}), \mathcal{R}_t, s_{t+1}\rangle$ in its memory M from which the agent learns and updates its policy π.

In the next paragraphs, we describe the essential elements in our framework, namely, (1) the state space, (2) the action space, (3) the reward function, and (4) the agent architecture.

5.1. State space

The state space is defined as a simple and straightforward vector instead of complicated definitions. Hence, the state vector $\boldsymbol{s}_t$ represents the current traffic conditions where each element of the vector counts 'the incoming vehicles' in each lane l towards the traffic lights intersection. The detection horizon of 'number of vehicles' v is limited to a given range H counting only the vehicles within the detection horizon H. Moreover, the state vector includes the current phase of traffic signals $P_t \in \{0, 1, 2, 3\}$. In a four-lane four-road intersection, the total number of lanes is $L = 16$, thus the state vector $s_t \in \mathbb{R}^{L+1}$, is given as follows:

$$s_t(v) = \begin{bmatrix} v_{0t} \\ v_{1t} \\ \cdot \\ \cdot \\ v_{L-1t} \\ P_t \end{bmatrix} \tag{16}$$

5.2. Reward function

For the reward signal $\mathcal{R}_t$, and similar to the pressure based reward definition as advocated by Wei et al. (2019), we define the reward function as follows:

$$\mathcal{R}_t = -Pressure = -|\sum_{l=0}^{L-1} q_l - \sum_{l=0}^{L-1} v_{Out_l}| \tag{17}$$

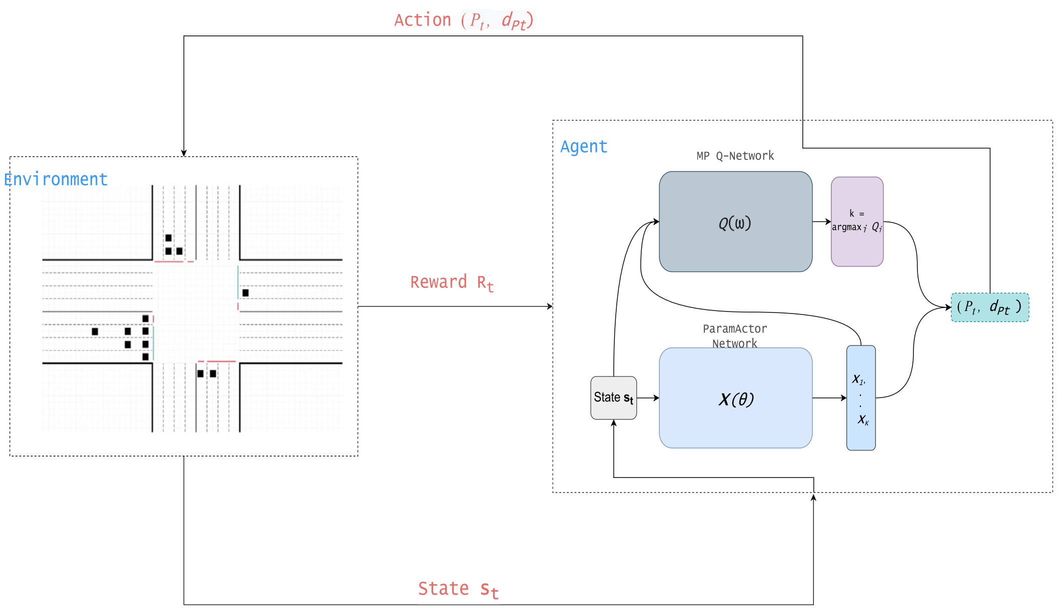

Fig. 3. The overall structure of our framework for optimizing the Phase selection and the time duration prediction in TSC system.

where $\sum_{l=0}^{L-1} q_l$ gives the sum of queuing vehicles in front of the intersection and $\sum_{l=0}^{L-1} v_{Out_l}$ is the sum of outgoing vehicles from the intersection. Loosely speaking, minimizing the *Pressure* will eventually optimize the degree of equilibrium between vehicles on the incoming and outgoing lanes and thus the vehicular throughput is boosted.

The agent's main objective is to maximize the total rewards in the long term starting from the time-step t i.e., the return. The return G is given as the total sum of discounted rewards starting from time-step t,

$$G = \sum_{k=0}^{n} \gamma^k \mathcal{R}_{t+k} \tag{18}$$

where n is the total time steps, γ is the discount factor and $\mathcal{R}$ is the reward signal.

5.3. Action space

In our framework, the action space is unlike the previous proposed action spaces discussed in the literature, where the agent either performs a discrete action (i.e., phase selection) or a continuous action (i.e., phase timing prediction). Instead, the action space is in the form of parameterized hierarchical action space. In parameterized action space, we define one sub-space for the possible finite traffic phases and the other is an interval for predicting the timing as a parameter associated with the selected phase. Hence, the first sub-space include four possible traffic phases represented by integers $P \in \{0, 1, 2, 3\}$, and the second continuous sub-space as a bounded time interval for associated phase, where $d_P \in [t_{min}, t_{max}]$. Consequently, the total action space is given by $\mathcal{A} = \{\{0, 1, 2, 3\} \cup \{[t_{min}, t_{max}]\}\}$. An example of the joint action is illustrated in Fig. 4 where the phase P constitutes a set of non-conflicting signals ('G' for green, 'r' for red and 'y' for yellow) to control each traffic movement, and the duration d_P falls in the interval [0 s, 45 s].

5.4. Agent's architecture

The most crucial part in the framework consists in the architecture of the agent. The agent's role is to learn an optimized policy that maps the observed states into specific actions. According to the parameterized action space defined earlier, most suitable architectures include parameterized DQN (P-DQN) (Xiong et al., 2018) and Multi-Pass-DQN (MP-DQN) (Bester et al., 2019) as describe in Section 3. Basically,

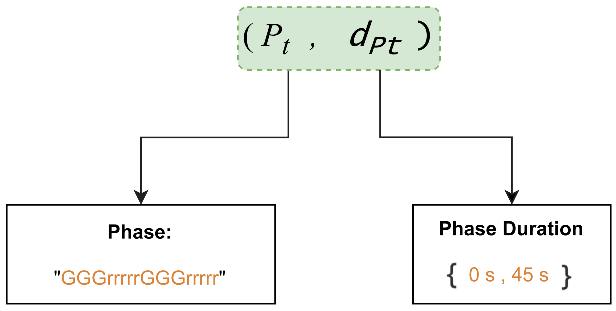

Fig. 4. Example of the agent's action that is applied to the traffic light.

MP-DQN is an improved version of P-DQN which notably performs better than p-DQN and allows the implementation of parameterized action space proposed in our framework. We adopt and customize the MP-DQN agent architecture implementation available online[1] to fit in our TSC framework. MP-DQN architecture exploits two neural networks according to the dual action spaces. The first, which is the Actor network, approximates the Q-values of the finite primary actions in order to select optimal phase P. The Actor network is denoted as $Q(s, P, d_P; \boldsymbol{\omega})$. The second network, named the ParamActor network, is used to approximate the policy mapping $x_{d\,P}$ so to predict the associated continuous parameters, denoted by $x_{d\,P}(s; \boldsymbol{\theta})$. Both networks come with target networks which improve the convergence performance and stability of the agent. The Actor network $Q(\boldsymbol{\omega})$ consists of an input layer of size (21), (16) inputs for the number of lanes at the intersection, (1) input for phase state and (4) inputs for the number of associated parameters of primary actions. The hidden layer of the Actor has (256) neurons with a *Relu* activation function, and the output layer is of 4 neurons representing Q-values for the four discrete primary actions. The ParamActor is of a similar structure except that the input layer is of size $16 + 1$ and the output is dedicated for predicting the continuous action-parameters.

The pseudo-code of our proposed framework is described in Algorithm 1. At first, the training and simulation settings $(\{lr_Q, lr_x\}, \epsilon, B, \zeta,$

[1] https://github.com/cycraig/MP-DQN.

$\boldsymbol{\omega}_0, \boldsymbol{\theta}_0$) are set in order to start interaction with simulation environment and begin training operation of the agent. In each episode of the training episodes, the agent interacts with the environment as a TSC by perceiving the intersection's traffic state s_t, and applying the dual action $a_t = (P_t, d_{P_t})$ to the traffic settings based on ϵ-greedy policy, where $a_t = (P_t, d_{P_t})$ is selected as:

$$a_t = \begin{cases} \text{a sample from} \quad \zeta \qquad \text{with probability } \epsilon, \\ (P_t, d_{P_t}), \quad P_t = \underset{P}{\operatorname{argmax}} Q(s_t, P, d_{P_t}; \boldsymbol{\omega}_t), 1-\epsilon. \end{cases} \tag{19}$$

Algorithm 1 Traffic Signal Control Using DRL with Parameterized Action Space

```
1:  Initialize: Stepsizes $\{\alpha_t, \beta_t\}_{t\geq 0}$, exploration parameter $\epsilon$, minibatch
    size $B$, a probability distribution $\zeta$, flow configurations, network
    weights $\boldsymbol{\omega}_0$ and $\boldsymbol{\theta}_0$.
2:  for episode $e = 1, \dots E$ do
3:      Start Simulation.
4:      Observe the initial state $s_0$ and take an initial joint action $a_0$.
5:      for $t = 1, \dots T$ do
6:          Compute action parameters $d_{P_t} \rightarrow x_{d\,P}(s_t; \boldsymbol{\theta}_t)$.
7:          Choose action $a_t = (P_t, d_{P_t})$ following the $\epsilon$ -greedy policy.
            $a_t = \begin{cases} \text{a sample from} \quad \zeta \text{with probability} \epsilon, \\ (P_t, d_{P_t}), \quad P_t = \underset{P}{\operatorname{argmax}} Q(s_t, P, d_{P_t}; \boldsymbol{\omega}_t), 1-\epsilon. \end{cases}$
8:          Apply the $a_t$ action, Obtain next state $s_{t+1}$ and get $\mathcal{R}_t$.
9:          Store the experience$< s_t, a_t, \mathcal{R}_t, s_{t+1} >$ in memory $M$.
10:         Randomly sample $B$ experiences from $M$.
11:
            $y_t = \begin{cases} \mathcal{R}_t \quad \text{if} \quad t = T, \\ \mathcal{R}_t + \max_P \gamma Q(s_{t+1}, P, x_{d\,P}(s_{t+1}; \boldsymbol{\theta}); \boldsymbol{\omega}_t) \text{ otherwise.} \end{cases}$
12:         Compute $\nabla_{\omega} \ell_t^Q(\boldsymbol{\omega}_t)$ and $\nabla_{\theta} \ell_t^Q(\boldsymbol{\theta})$ using $\{y_t, s_t, a_t\}$.
13:         update weights $\boldsymbol{\omega}_{t+1} \leftarrow \boldsymbol{\omega}_t - \alpha \nabla_{\omega} \ell_t^Q(\boldsymbol{\omega}_t)$ and $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t -$
            $\beta \nabla_{\theta} \ell_t^Q(\boldsymbol{\theta})$.
14:     end for
15: end for
```

where ζ is a uniform random distribution over a continuous interval $[t_{min}, t_{max}]$. Upon performing an action to the TSC, the agent gets a reward $\mathcal{R}_t$ and the new state of traffic s_{t+1} is observed. The undergone experience is saved in a memory M as a tuple $(\langle s_t, (P_t, d_{P_t}), \mathcal{R}_t, s_{t+1} \rangle)$ for the training purpose. The agent remains in its current state until the predicted duration d_{P_t} elapses, after which it proceeds to observe the next state s_{t+1} predict the next action a_{t+1}. When the size of the memory reaches an initial memory threshold, the agent randomly samples a batch of size B that is used for training the agent and update its policy. The gradients $\nabla_{\omega} \ell_t^Q(\boldsymbol{\omega}_t)$ and $\nabla_{\theta} \ell_t^Q(\theta)$ are computed and used besides the learning rates $\{lr_Q, lr_x\}$ to update $\boldsymbol{\omega}$ and $\boldsymbol{\theta}$ weights. For time complexity, the dominant computational cost in the proposed framework arises from the two neural networks the Actor network $Q(s, P, d_P; \boldsymbol{\omega})$ and the ParamActor network $x_{d\,P}(s; \boldsymbol{\theta})$. The forward passes contribute $O(m_{\omega} \times n_{\omega} + m_{\theta} \times n_{\theta})$, where m_{ω} and m_{θ} are the number of layers, and n_{ω} and n_{θ} are the number of neurons per layer in the Actor and ParamActor networks, respectively.

During training, backpropagation adds additional computational cost due to gradient computations and weight updates for both networks. The backward passes scale with the batch size B, adding $O(B \times (m_{\omega} \times n_{\omega} + m_{\theta} \times n_{\theta}))$. Hence, the overall computational complexity can be expressed and simplified as:

$$O(E \times T \times (B \times (m_{\omega} \times n_{\omega} + m_{\theta} \times n_{\theta})))$$

6. Experiments

In this section, we begin by presenting the research questions we aim to answer and the hypotheses we seek to validate. We then introduce the simulation environment setup, the different parameters used and the evaluation metrics upon which we assess the performance of the framework. We evaluate the overall performance of the proposed approach for managing the traffic signal control systems based on the obtained results. Further evaluations are made by comparing our framework to other benchmarks reported in the literature including DRL based approaches and meta-heuristic based algorithms. At the end, we discuss the results obtained from various experimentation scenarios as well as the results of benchmarks comparison.

6.1. Hypotheses

By conducting the set of upcoming experiments using our approach and the benchmarks, we aim to address the following research questions. (1) How does the proposed approach perform compared to fixed-time approach and deep learning based approaches? (2) How efficient is our approach compared to meta-heuristic based approaches?. To answer these research questions, we have formulated to two hypotheses:

- H1: Based on travel time, our hybrid approach is more performant than both fixed time methods and the deep learning-based approaches, namely, discrete approach (DQN) and continuous approach (DDPG).
- H2: Based on travel time, our hybrid approach is more performant than metaheuristic Based Approaches, namely GA and PSO.

6.2. Experimental setup

The evaluations of our framework are based on simulations made using the popular open source SUMO traffic simulator[2] to simulate the intersection environment (Behrisch et al., 2011).

For the signalized intersection, we have 4 lanes for each incoming/outgoing road all are 750 m long and a maximum speed of 13.89 m/s. Three right-most lanes in each road are devoted for going straight and right turn movements of vehicles, and the left lane is kept for the turning left vehicles.

Using custom scripts, the traffic flow of vehicles is generated prior to each simulation episode. The traffic flow is generated such that it mimics the real traffic flow. In real traffic scenario for instance, the traffic flow starts with low vehicular density, and it keeps rising until the peak value traffic, finally the traffic starts to relief by lower vehicular densities. We generate such a traffic flow (i.e., low, high then low) by assigning one hour for low traffic, one hour for high traffic and one hour for low traffic again, a total of 10800 s of generated traffic. The vehicle routes (i.e., the trip from origin to destination) are generated such that a portion of 25% of vehicles turn left or right, and a portion of 75% decide to go straight. Table 5 describes the generated traffic flow scenarios with different configurations.

6.3. Parameters tuning and setting

Various training parameters and settings have to be carefully adjusted and correctly tuned so to fit the customized MP-DQN architecture in our framework. We set the training parameters and settings after a number of simulation experiments as following, the agent is trained on $N = 301$ episodes with each episode lasts for 3800 s. During the training, the action selection of the agent is based on $\epsilon - greedy$ for

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

Table 5
Simulated traffic flow.

Distribution Type	Scenario	Generated flow (veh/h)	Start time (s)	End time (s)
Weibull Dist	SC1	1500	0 s	3800 s
	SC2	4000	0 s	3800 s
	SC3	SC1-SC2-SC1	0 s	11 000 s
Normal Dist	SC4	1500	0 s	3800 s
	SC5	4000	0 s	3800 s
	SC6	SC4-SC5-SC4	0 s	11 000 s

Table 6
Values used for training parameters.

Parameter	Description	Value
N	Number of training episodes	451
M	Replay Memory	20 000
b	Mini-batch size	64
lr_Q	Actor Learning rate	0.001
lr_x	ParamActor Learning rate	0.00001
γ	Gamma factor	0.95
eps_min	Minimum value of epsilon	0.01
epsilon episodes	Number of epsilon episodes	300
yellow duration	Yellow phase duration	3 s

the discrete actions and random uniform selection for the continuous parameters. The exploration parameter ϵ is annealed from 1 down to 0.01 for a period of 270 episodes. The agent's memory size is set to 20,000 experiences, and the agent starts training and updating its policy if the number of stored experiences exceeds 128. After each time-step, the agent randomly samples a mini-batch of $b = 64$ experiences to learn from and update its networks' weights. Both Actor and ParamActor networks weights are updated using the RMSProp (Tieleman and Hinton, 2012) stochastic gradient decent method where the learning rates are set to $lr_Q = 0.001$ and $lr_x = 0.00001$ respectively. To keep the continuous action parameters inside the bounded interval, we use the inverting gradients method as proposed in Hausknecht and Stone (2016). The gradient clipping method is also applied with a value of 1 in order to speed up the training process. Different used parameters' values are summarizes in Table 7 (see Table 6).

6.4. Performance evaluation metrics

To evaluate the performance of our approach, we use three common evaluation metrics for traffic signal control approaches (1) Average Travel Time, (2) the queue length and (3) the average waiting time of vehicles (Liu et al., 2023; Kolat et al., 2023). In the following, we describe of these performance metrics.

6.4.1. Average Travel Time (ATT)

Travel time of a vehicle is the time it spends to arrive to its destination starting form its origin point. Average travel time of vehicles is the sum of travel time of all vehicles divided by the total number of vehicles, defined as:

$$ATT = \frac{1}{N_{veh}} \sum_{j=0}^{N_{veh}} (t_{j,start} - t_{j,end}), \tag{20}$$

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

6.4.2. Average Waiting Time (AWT)

The waiting time of a vehicle over an episode is computed by summing up all times its speed was less than 0.1 m/s. The Average Waiting Time is then calculated by summing waiting time of all vehicles divided by the total number of vehicles.

$$AWT = \frac{1}{N_{veh}} \sum_{j=0}^{N_{veh}} WT_j, \tag{21}$$

where WT_j represents the waiting time of each vehicle j over a simulation episode.

Table 7
Training parameters for Meta-Heuristics.

Parameter	Value
Genetic algorithm	
Population size	10
Number of generations	30
Mutation probability	0.1
cross-over probability	0.5
PSO algorithm	
Population size	10
Number of generations	30
Number of particles	0.01
c1 & c2 coefficients	300

6.4.3. Queue Length (QL)

This metric corresponds to the total number of vehicles queuing on all lanes. A vehicle is in queuing state if its speed is less than 0.1 m/s on a particular lane. We consider the average queue length over all recorded values during the training episodes as well.

6.5. Benchmarks

For a practical validation of our proposed framework, we compare it to state-of-the-art approaches, the Fixed-time approach (Gordon and Tighe, 2005), the DQN discrete approach (van Hasselt et al., 2015), the DDPG continuous approach (Lillicrap et al., 2019) and two meta-heuristic approaches: genetic algorithm (GA) (Holland, 1992) and particle swarm optimization (PSO) (Eberhart and Kennedy, 1995).

6.5.1. Fixed time approach

Fixed-time approach is a static approach where we have a fixed sequence order of phases with each phase duration is fixed (Gordon and Tighe, 2005). In the experiment, We set the green phase to 30 s, and 3 s for the yellow phase.

6.5.2. Discrete approach

In the discrete DQN approach for TSC, the agent is trained to pick a proper phase from the list of phases without any particular order, but the phase duration is fixed. In this approach, for the agent architecture we use the Double-DQN with Prioritized Experience Replay Memory. The formulae of state and reward are both based on the queue length as in our proposed framework.

6.5.3. Continuous approach

In contrast with the discrete approach, the continuous approach only predicts the next phase's duration within a fixed order of phases. To implement this approach, we leverage the DDPG continuous architecture for the agent, and the rest is similar to the discrete approach.

6.5.4. Meta-heuristics

We also compare our framework to two meta-heuristic benchmarks, the Genetic Algorithm (GA) and the Particle Swarm Optimization algorithm (PSO) which have been discussed in Section 3. In both algorithms, we set the size of population to 10 individuals, and the number of generations to be 30 generations. The GA mutation probability is set to 0.1 and the cross-over is based on the discrete recombination with probability of 0.5. For the PSO, each swarm encloses 4 particles, and the acceleration coefficients c1, and c2 are set to 0.5. The resulting solutions of both algorithms represent the phase timing of traffic lights, bounded from 5 s to 45 s. These phase durations are integrated into each traffic simulation episode to be tested and the fitness function is obtained as a result to evaluate the candidate solutions and update the population over iterations/generations.

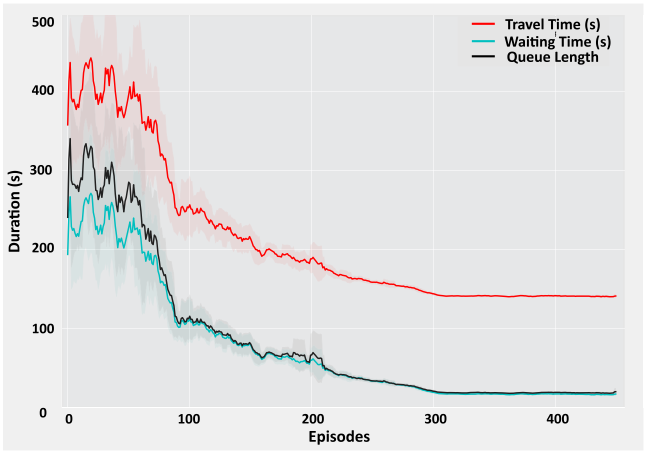

Fig. 5. Training curves of the proposed framework based on queue length, waiting time and travel time metrics with respect to episodes.

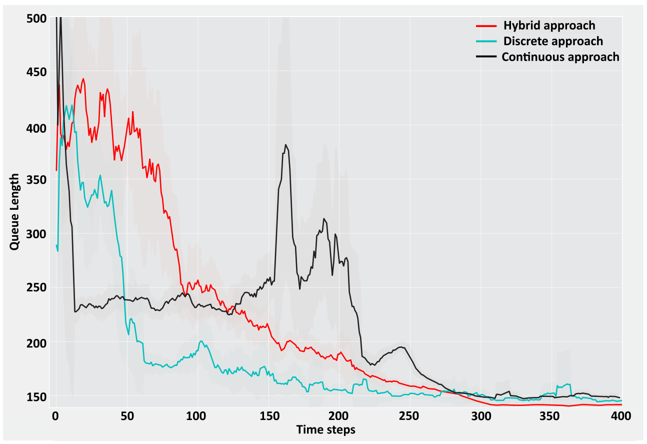

Fig. 6. Comparison of training curves of our proposal against discrete and continuous benchmarks during C2 scenario based on travel time metric.

6.6. Results and discussion

In this section, we demonstrate the results of the experiments according to the performance evaluation metrics and different flow configurations mentioned earlier. The training performance curves of the agent are shown in Fig. 5. A comparison of training performance of the proposed framework versus the Discrete and Continuous approaches is illustrated in Fig. 6. Notably during the training, the Discrete approach (cyan colored curve) initially learns faster (since it already maintains fixed phase durations and its objective is only to select the most suitable phase) but converges to a sub-optimal solution. Furthermore, the Continuous approach (black colored curve) seems to be less stable and oscillates at the beginning then it reaches a better performance but still worse than the rest. Both the discrete and continuous approaches have been trained longer for 501 episodes as they took more time to converge and stabilize. On the other side, our framework (the parameterized approach curve in red color) is linearly decaying until it outperforms the Discrete and the Continuous curves. This happens due to the fact that the parameterized framework is more flexible and strives to optimize both the phase selection and the timing associated.

6.6.1. Comparison with fixed time and deep learning based approaches

In this comparison we target to statistically verify the following hypothesis H1 stating that "Based on travel time, our hybrid approach is more performant than both fixed time methods and the deep learning based approaches, namely discrete approach (DQN) and continuous approach (DDPG)". Our approach as well as the Fixed-Time, the discrete and the continuous algorithms were run using six simulation scenarios (SC1, …, SC6) and repeated several times. Each run time is an episode of 200 time steps (the value 200 is determined empirically). The average of travel time as well as the standard deviation are recorded and depicted in Table 8 for our approach as well as the aforementioned benchmarks.

Remarkably, we observe from the table that the Fixed Time method scores far behind the competition since its static behavior cannot handle perfectly the dynamic traffic flow situations. However, the deep reinforcement learning based approaches perform clearly better due to the fact that they can effectively handle the dynamic traffic flow conditions. Out of the deep reinforcement learning approaches, our parameterized framework surpasses the benchmarks in all experiments. t-Test is performed to assess the evidence supporting our hypothesis $H1$. The computed p-values recorded in Table 8 show the significant out-performance of our approach over the benchmarks in all scenarios. The null hypothesis stating that there is no difference between our approach and the benchmarks is rejected with a confidence of 99% (p-value < 0.01).

This is an interesting feature of our proposal, since our framework can efficiently control both the phase selections and predict the associated timing of the selected phase which makes the parameterized framework more flexible than others.

In Fig. 7, we present the queue length performance of the Discrete, Continuous and the proposed parameterized approaches during one episode simulation. The illustrated resulting curves show that the performance of our framework outperforms the benchmarks as it by maintaining a lower and more steady queue length during the traffic simulation episode.

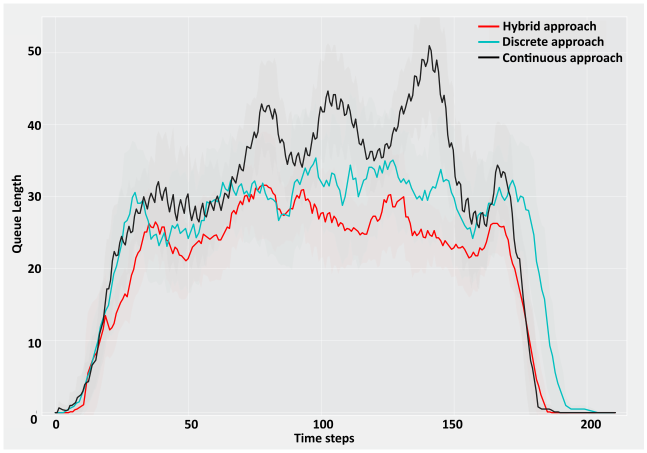

Fig. 7. Queue length performance comparison of our proposal against discrete and continuous approaches.

6.6.2. Comparison with meta-heuristics

In this comparison, we statistically verify the following hypothesis (H2). Table 9 presents the comparison results between PSO, GA and our approach under multiple simulation scenarios (i.e., SC1-SC6). The comparison reveals that while both meta-heuristic approaches, namely, PSO and GA, perform relatively close to each other, they remain significantly behind the proposed parameterized framework across all scenarios. This highlights the limitations of meta-heuristic algorithms in handling the highly dynamic and complex nature of traffic signal control. t-Test is conducted to evaluate the evidence supporting our hypothesis $H2$. When compared with alternatives from meta-heuristics, the computed p-values demonstrate that our approach Substantially outperforms the GA and PSO algorithms in all scenarios. Thus the null

Table 8
Mean and STD of average travel time (s) performance comparison of proposed framework to others.

Scenarios						
Alternative approaches	SC1	SC2	SC3	SC4	SC5	SC6
Fixed-Time: Mean (Std) (*p*-value, One tail t-Test)	164.4 (2.2) (7.27086e−15)	260.6 (16) (1.918e−9)	220.6 (5.4) (2.291e−12)	161.4 (0.5) (1.472e−27)	217.3 (8.7) (2.723e−10)	197.1 (5.0) (1.620e−11)
Discrete approach: Mean (Std) (*p*-value, One tail t-Test)	138.5 (1.2) (2.246e−8)	145.7 (0.9) (1.459e−10)	142.4 (0.7) (3.462e−11)	135.9 (0.5) (1.963e−14)	143.7 (0.8) (6.340e−11)	141.3 (0.6) (1.489e−12)
Continuous approach: Mean (Std) (*p*-value, One tail t-Test)	136.8 (0.9) (6.061e−7)	153.8 (4.2) (2.365e−6)	146.6 (1.9) (5.319e−8)	134.7 (0.6) (2.080e−11)	148.0 (3.1) (3.384e−6)	144.0 (1.9) (1.849e−7)
Parameterized approach: Mean **(Std)**	**133.3** **(1.1)**	**140.6** **(0.8)**	**137.9** **(0.4)**	**130.7** **(0.5)**	**138.3** **(0.75)**	**136.5** **(0.5)**

Table 9
Performance comparison of the proposed framework against meta-heuristic algorithms based on average travel time metric.

Scenarios						
Metaheuristics	SC1	SC2	SC3	SC4	SC5	SC6
Genetic algorithm:Mean (Std) (*p*-value, One tail t-Test)	144.9 (0.74) (1.80e−14)	150.9 (1.46) (1.44e−11)	148.7 (0.67) (1.74e−17)	143.2 (0.52) (5.50e−21)	149.1 (1.12) (6.84e−14)	147.5 (0.66) (1.66e−18)
PSO algorithm: Mean (Std) (*p*-value, One tail t-Test)	143.1 (0.85) (1.15e−13)	149.3 (2.24) (1.20e−7)	146.8 (0.84) (1.10e−13)	141.2 (0.63) (1.50e−18)	147.4 (1.61) (1.05e−9)	145.6 (0.71) (2.22e−16)
Parameterized approach: Mean **(Std)**	**133.3** **(1.1)**	**140.6** **(0.8)**	**137.9** **(0.4)**	**130.7** **(0.5)**	**138.3** **(0.75)**	**136.5** **(0.5)**

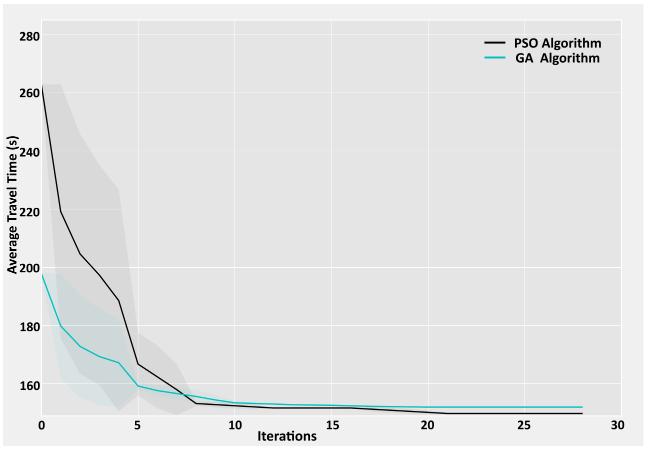

Fig. 8. Training curves of PSO and GA algorithms over generations/iterations.

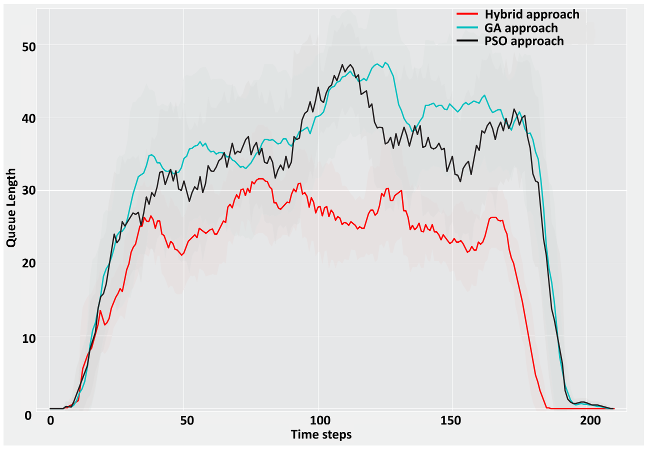

Fig. 9. Queue length performance comparison of parameterized approach against meta-heuristic-based algorithms.

hypothesis stating that there is no difference between our approach and the PSO and GA algorithms is rejected with a confidence of 99% (p-value < 0.01). Further evidence of this performance gap can be observed in Fig. 9, which illustrates the queue length performance during a single episode of simulation. The results clearly demonstrate the superiority of the proposed framework, as it consistently minimizes vehicle queue lengths compared to the PSO and GA approaches.

Fig. 8 depicts the training curves of the meta-heuristic algorithms, GA and PSO, based on average travel time over iterations. Both algorithms exhibit similar convergence behavior except that the PSO algorithm converges to a slightly lower average travel time value.

7. Threats to validity

Similar to any experimental study, the results of our study might have been impacted by a number of factors. In this section, we discuss the main threats to validity and how we mitigated them.

7.1. Construct threats to validity

Construct threats to validity are concerned with the relation between theory and observation. A key point in this regard is related to the suitability of our evaluation measures. We used three different performance evaluation measures that are commonly used for evaluating the traffic signal control approaches, namely the average travel time (ATT), the waiting time (AWT), and the queue length (QL). Thus, we believe that by using multiple performance measures that reflect different perspectives, there is negligible threat to construct validity. Other possible threats with the measurement of the exploited traffic data, as well as the independent variables, have been mitigated by generating various dynamic traffic flows that mimic real-world scenarios, as described in Section 6. These traffic scenarios are simulated using tools well established in the TSC field, such as SUMO (Behrisch et al., 2011), which is accurate enough for conducting our study.

7.2. Internal threats to validity

Threats to internal validity concern the possible errors in our experiments. We used reliable traffic datasets that are supported by SUMO (Behrisch et al., 2011), a widely-used open source traffic simulator. Another possible threat to internal validity could be related to bias in the replication of the benchmark approaches. We used the standard implementation of the Multi-Pass Parameterized Deep Q-networks (MP-DQN) based on the agent architecture implementation available online[3] to fit in our TSC framework. Furthermore, we used the standard experiment setting for the benchmark approaches (fixed time, discrete, and continuous approaches). We also double checked all the experimental setup. Thus, we believe there is negligible threat to internal validity. To reduce errors in our code, we have double checked and fully tested our code, still there could be errors that we did not notice.

7.3. Conclusion threats to validity

Conclusion threats to validity concern the relationship between the treatment and the outcome. Indeed, in our empirical evaluation, we statistically analyzed the obtained results using the t-Test statistical analysis which provided strong evidence for validating our assumptions and our experimental study. Hence, we believe that there is negligible threat to the validity of our conclusions.

7.4. External threats to validity

Threats to external validity relate to the quality of our datasets and generalizability of our findings. While our experiments are based on a widely used open source tool, Simulation of Urban MObility (SUMO) simulator (Behrisch et al., 2011) and synthetic traffic datasets, to efficiently simulate the intersection environment, we are planning to conduct further experiments on real world traffic data. Indeed, some constraints may apply when implementing our approach in real intersections such as the limitations posed by maximum queue length and the calculated average travel time, which rely on the available sensors on the road.

8. Conclusion and future work

In this paper, we addressed the challenge of simultaneously controlling traffic signal control (TSC) phase selection and predicting the corresponding phase duration. To achieve this, we leveraged a state-of-the-art parameterized deep reinforcement learning architecture, namely, Multi-Pass Parameterized Deep Q-networks (MP-DQN). We tailored and adapted the MP-DQN architecture in the TSC environment to effectively select the appropriate phase and while simultaneously predicting its duration. To evaluate our framework and demonstrate its performance, we run multiple simulated experiments with different traffic scenarios and configurations. Our proposal is further compared with a set of baselines including traditional and learning-based approaches such as discrete, continuous, and meta-heuristic (GA and PSO) approaches. The performance of these approaches is evaluated based on average travel time (ATT), average waiting time (AWT), and queue length (QL) evaluation metrics. The results clearly demonstrate that our framework outperforms all benchmarks across the simulated experiments. This superior performance is attributed to the framework's capability to efficiently handle phase selection while predicting the corresponding phase timing. Specifically, our approach improves the travel time performance by 33%, 3.5%, and 5.3% over Fixed-Time, Discrete, and Continuous approaches respectively, and by 7.5% and 6.4% over GA and PSO approaches respectively.

[3] https://github.com/cycraig/MP-DQN.

One advantage of the proposed approach is that it integrates phase selection and phase timing prediction into a single framework, leading to more adaptive and efficient traffic signal control. This provides flexibility for dynamic traffic scenarios, contributing to its superior performance across various evaluation metrics. However, one disadvantage of our approach is the increased computational complexity due to the parameterized action space, which could affect real-time deployment in large-scale or multi-intersection scenarios. The limitations of this work include its reliance on simulated environments, which may not fully represent real-world traffic conditions. Additionally, the computational demands of the MP-DQN architecture may pose challenges for scalability in larger networks.

Future works could be dedicated to conducting further experiments on multi-intersection scenarios and incorporating real-world data from signalized intersections to improve realism and applicability. Efforts could also focus on optimizing the computational efficiency of the architecture, potentially through lightweight neural network designs, to facilitate real-time deployment in large-scale traffic networks.

CRediT authorship contribution statement

Salah Bouktif: Writing – review & editing, Writing – original draft, Validation, Supervision, Project administration, Methodology, Funding acquisition, Conceptualization. **Abderraouf Cheniki:** Writing – review & editing, Writing – original draft, Software, Methodology, Formal analysis, Data curation, Conceptualization. **Ali Ouni:** Writing – review & editing, Writing – original draft, Validation, Funding acquisition. **Hesham El-Sayed:** Writing – review & editing, Supervision, Funding acquisition.

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.

Acknowledgments

This work was supported by the Emirates Center for Mobility Research (ECMR) of the United Arab Emirates University (grant number 31R225) and by Abu Dhabi Department of Education and Knowledge (ADEK award number AARE18-114 grant 21T052).

Data availability

Synthetic Data and Codes that support the findings of this study are openly available at https://github.com/abderraouf2che/Hybrid-Deep-RL-Traffic-Signal-Control.

References

Behrisch, M., Bieker, L., Erdmann, J., Krajzewicz, D., 2011. SUMO - simulation of urban mobility: An overview. In: In SIMUL 2011, the Third International Conference on Advances in System Simulation. pp. 63–68.

Bester, C.J., James, S.D., Konidaris, G.D., 2019. Multi-pass Q-networks for deep reinforcement learning with parameterised action spaces. arXiv:1905.04388.

Bouktif, S., Cheniki, A., Ouni, A., El-Sayed, H., 2023. Deep reinforcement learning for traffic signal control with consistent state and reward design approach. Knowl.-Based Syst. 267, 110440. http://dx.doi.org/10.1016/j.knosys.2023.110440, URL: https://www.sciencedirect.com/science/article/pii/S0950705123001909.

Casas, N., 2017. Deep deterministic policy gradient for urban traffic light control. arXiv:1703.09035.

Dell'Orco, M., Başkan, Ö., Marinelli, M., 2014. Artificial bee colony-based algorithm for optimising traffic signal timings. In: Snáˇ sel, V., Krömer, P., Köppen, M., Schaefer, G. (Eds.), Soft Computing in Industrial Applications. Springer International Publishing, Cham, pp. 327–337.

Eberhart, R., Kennedy, J., 1995. A new optimizer using particle swarm theory. In: MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science. pp. 39–43. http://dx.doi.org/10.1109/MHS.1995.494215.

García-Nieto, J., Alba, E., Carolina Olivera, A., 2012. Swarm intelligence for traffic light scheduling: Application to real urban areas. Eng. Appl. Artif. Intell. 25 (2), 274–283. http://dx.doi.org/10.1016/j.engappai.2011.04.011, URL: http://www.sciencedirect.com/science/article/pii/S0952197611000777 Special Section: Local Search Algorithms for Real-World Scheduling and Planning.

Genders, W., Razavi, S., 2016. Using a deep reinforcement learning agent for traffic signal control. arXiv:1611.01142.

Gordon, R., Tighe, W., 2005. Traffic control systems handbook (2005 edition).

Gu, S., Lillicrap, T., Sutskever, I., Levine, S., 2016. Continuous deep Q-learning with model-based acceleration. arXiv:1603.00748.

Haarnoja, T., Zhou, A., Abbeel, P., Levine, S., 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. arXiv:1801.01290.

van Hasselt, H., Guez, A., Silver, D., 2015. Deep reinforcement learning with double Q-learning. arXiv:1509.06461.

Hausknecht, M., Stone, P., 2016. Deep reinforcement learning in parameterized action space. arXiv:1511.04143.

Haydari, A., Yilmaz, Y., 2022. Deep reinforcement learning for intelligent transportation systems: A survey. IEEE Trans. Intell. Transp. Syst. 23 (1), 11–32. http://dx.doi.org/10.1109/TITS.2020.2999526.

Holland, J.H., 1992. Genetic algorithms. Sci. Am. 267 (1), 66–73, URL: http://www.jstor.org/stable/24939139.

Hussain, K., Mohd Salleh, M.N., Cheng, S., Shi, Y., 2019. Metaheuristic research: a comprehensive survey. Artif. Intell. Rev. 52 (4), 2191–2233. http://dx.doi.org/10.1007/s10462-017-9605-z.

INRIX Scoreboard, P.R., 2022. URL: https://inrix.com/scorecard/.

Karaboga, D., Basturk, B., 2007. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J. Global Optim. 39 (3), 459–471. http://dx.doi.org/10.1007/s10898-007-9149-x.

Khamis, M.A., Gomaa, W., 2014. Adaptive multi-objective reinforcement learning with hybrid exploration for traffic signal control based on cooperative multi-agent framework. Eng. Appl. Artif. Intell. 29, 134–151. http://dx.doi.org/10.1016/j.engappai.2014.01.007.

Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., 1983. Optimization by simulated annealing. Sci. 220, 671–680.

Kolat, M., Kővári, B., Bécsi, T., Aradi, S., 2023. Multi-agent reinforcement learning for traffic signal control: A cooperative approach. Sustain. 15 (4), http://dx.doi.org/10.3390/su15043479, URL: https://www.mdpi.com/2071-1050/15/4/3479.

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D., 2019. Continuous control with deep reinforcement learning. arXiv:1509.02971.

Liu, X.-Y., Zhu, M., Borst, S., Walid, A., 2023. Deep reinforcement learning for traffic light control in intelligent transportation systems. arXiv:2302.03669 URL: https://arxiv.org/abs/2302.03669.

Masson, W., Ranchod, P., Konidaris, G., 2015. Reinforcement learning with parameterized actions. arXiv:1509.01644.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M., 2013. Playing atari with deep reinforcement learning. arXiv:1312.5602.

Oda, T., Otokita, T., Tsugui, T., Mashiyama, Y., 1997. Application of simulated annealing to optimization of traffic signal timings. IFAC Proc. Vol. 30 (8), 733–736. http://dx.doi.org/10.1016/S1474-6670(17)43908-5, URL: http://www.sciencedirect.com/science/article/pii/S1474667017439085 8th IFAC/IFIP/IFORS Symposium on Transportation Systems 1997 (TS '97), Chania, Greece, 16-18 June.

Osman, I.H., Kelly, J.P., 1996. Meta-heuristics: An overview. In: Osman, I.H., Kelly, J.P. (Eds.), Meta-Heuristics: Theory and Applications. Springer US, Boston, MA, pp. 1–21. http://dx.doi.org/10.1007/978-1-4613-1361-8_1.

Rasheed, F., lim Alvin Yau, K., Noor, R.M., Wu, C., Low, Y.C., 2020. Deep reinforcement learning for traffic signal control: A review. IEEE Access 8, 208016–208044, URL: https://api.semanticscholar.org/CorpusID:227221242.

Shi, Y., Qi, Y., Lv, L., Liang, D., 2021. A particle swarm optimisation with linearly decreasing weight for real-time traffic signal control. Mach. 9 (11), URL: https://www.mdpi.com/2075-1702/9/11/280.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., Riedmiller, M., 2014. Deterministic policy gradient algorithms. In: Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32. ICML '14, JMLR.org, pp. I–387–I–395.

Tan, M.K., Chuo, H.S.E., Chin, R.K.Y., Yeo, K.B., Teo, K.T.K., 2016. Genetic algorithm based signal optimizer for oversaturated urban signalized intersection. In: 2016 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia). pp. 1–4. http://dx.doi.org/10.1109/ICCE-Asia.2016.7804762.

Tieleman, T., Hinton, G., 2012. Lecture 6.5 - rmsprop: Divide the gradient by a running average of its recent magnitude.

Vidali, A., Crociani, L., Vizzari, G., Bandini, S., 2019. A deep reinforcement learning approach to adaptive traffic lights management. In: WOA.

Watkins, C.J.C.H., Dayan, P., 1992. Q-learning. Mach. Learn. 8 (3), 279–292. http://dx.doi.org/10.1007/BF00992698.

Wei, H., Chen, C., Zheng, G., Wu, K., Gayah, V., Xu, K., Li, Z., 2019. 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. KDD '19, Association for Computing Machinery, New York, NY, USA, pp. 1290–1298. http://dx.doi.org/10.1145/3292500.3330949.

Xiong, J., Wang, Q., Yang, Z., Sun, P., Han, L., Zheng, Y., Fu, H., Zhang, T., Liu, J., Liu, H., 2018. Parametrized deep Q-networks learning: Reinforcement learning with discrete-continuous hybrid action space. arXiv:1810.06394.