Model-Free Perimeter Metering Control for Two-Region Urban Networks Using Deep Reinforcement Learning

Dongqin Zhou[1] and Vikash V. Gayah[1*]

[1]*Department of Civil and Environmental Engineering, The Pennsylvania State University, University Park, PA, 16802*

* Corresponding author.

E-mail addresses: dongqin.zhou@psu.edu (D. Zhou), gayah@engr.psu.edu (V. V. Gayah)

ABSTRACT

Various perimeter metering control strategies have been proposed for urban traffic networks that rely on the existence of well-defined relationships between network productivity and accumulation, known more commonly as network Macroscopic Fundamental Diagrams (MFD). Most existing perimeter metering control strategies require accurate modeling of traffic dynamics with full knowledge of the network MFD and dynamic equations to describe how vehicles move across regions of the network. However, such information is generally difficult to obtain and subject to error. Some model free perimeter metering control schemes have been recently proposed in the literature. However, these existing approaches require estimates of network properties (e.g., the critical accumulation associated with maximum network productivity) in the controller designs. In this paper, a model free deep reinforcement learning perimeter control (MFDRLPC) scheme is proposed for two-region urban networks that features agents with either continuous or discrete action spaces. The proposed agents learn to select control actions through a reinforcement learning process without assuming any information about environment dynamics. Results from extensive numerical experiments demonstrate that the proposed agents: (a) can consistently learn perimeter control strategies under various environment configurations; (b) are comparable in performance to the state-of-the-art, model predictive control (MPC); and, (c) are highly transferable to a wide range of traffic conditions and dynamics in the environment.

Keywords: Macroscopic Fundamental Diagram (MFD); model free deep reinforcement learning (Deep-RL); perimeter control

1. INTRODUCTION

Network Macroscopic Fundamental Diagrams (MFDs) provide a unimodal, low-scatter relationship between the average productivity (e.g., network space-mean flow or trip completion rate) and average use (e.g., network vehicle density or accumulation) for homogeneously loaded urban networks. The proposition of the MFD was initially developed in (Godfrey, 1969) and later studied in (Mahmassani et al., 1987, 1984), but its existence was not established until recent theoretical and empirical findings (Daganzo, 2007; Geroliminis and Daganzo, 2008). Thereafter extensive empirical and simulation investigations have been conducted for the existence and properties of well-defined MFDs (Buisson and Ladier, 2009; Daganzo et al., 2011; Geroliminis and Sun, 2011; Ji et al., 2010; Mazloumian et al., 2010).

MFDs can be used to model urban traffic networks with much less complexity by focusing on the movement of vehicles across regions, and this paradigm can lead to the design of elegant regional level traffic control strategies. Examples include pricing, street network design, and multimodal street allocation (Daganzo and Lehe, 2015; Gayah et al., 2014; Zheng et al., 2012; Zheng and Geroliminis, 2013). The most common application of MFDs for control purposes is perimeter metering control, which is the manipulation of vehicle flows between neighboring regions. Perimeter metering control strategies have been proposed for single-region networks in (Daganzo, 2007; Haddad, 2017a; Haddad and Shraiber, 2014; Keyvan-Ekbatani et al., 2012, 2015a). For two-region urban networks, the perimeter control problem was first formulated in (Haddad and Geroliminis, 2012) and subsequently solved in (Geroliminis et al., 2013), while numerous extensions to multi-region networks can be found in (Aboudolas and Geroliminis, 2013; Haddad et al., 2013; Hajiahmadi et al., 2015; Keyvan-Ekbatani et al., 2015b; Lei et al., 2019; Ren et al., 2020; Sirmatel and Geroliminis, 2018).

Various methods have been proposed to solve MFD-based perimeter control problems. The most promising and extensively used in the literature is model predictive control (MPC) (Geroliminis et al., 2013; Haddad, 2017b; Haddad et al., 2013; Hajiahmadi et al., 2015; Ramezani et al., 2015; Sirmatel and Geroliminis, 2018). However, the success of the MPC approach is heavily dependent on the accuracy of the MFD prediction model. While the estimation of a network's MFD has been studied intensively (Ambühl and Menendez, 2016; Du et al., 2016; Gayah and Dixit, 2013; Nagle and Gayah, 2014; Saberi et al., 2014), the scarcity of empirically observed MFDs in the literature demonstrates the practical difficulty of such MFD estimation efforts. The MPC is a rolling horizon control scheme and may not generalize well to new plants (reality) owing to its sensitivity to horizon parameters and modeling uncertainties (Prabhu and George, 2014; Schrangl et al., 2018). Non-MPC methods for perimeter control have also been proposed and shown to be promising. These include proportional-integral based control (Haddad and Shraiber, 2014; Keyvan-Ekbatani et al., 2015b, 2015a, 2012), adaptive control (Haddad and Mirkin, 2017, 2016), and linear quadratic regulator (Aboudolas and Geroliminis, 2013; Kouvelas et al., 2017; Ni and Cassidy, 2020). However, all these methods are either model-based (i.e., assume full regional traffic dynamics to be known a priori) or require information about the network's MFD, which render them susceptible to the potential mismatch between the prediction model and environment dynamics.

To overcome these challenges, data-driven model free schemes have been proposed for perimeter control problems (Lei et al., 2019; Li and Hou, 2020; Ren et al., 2020, 2018). However, these schemes require the estimation of network properties (e.g., the critical accumulation

associated with maximum network productivity) that are embedded in the controller design. Thus, they may be prone to estimation errors due to multivaluedness, instability, and hysteresis phenomena that are common in real networks (Daganzo et al., 2011; Gayah and Daganzo, 2011; Mahmassani et al., 2013; Mazloumian et al., 2010). Further, it remains uninvestigated whether these methods could transfer to unencountered environment scenarios since the controllers learn from scratch for each case study considered. It is therefore desirable to develop methods for perimeter control that are completely model-free and do not build on pre-collected traffic data for the estimation of MFDs.

Reinforcement learning (RL), which has long been studied for sequential decision-making problems (Sutton and Barto, 2018), might be an appropriate technique for this task. In particular, the authors are interested in the applicability of model-free control methods as they require little to no detailed knowledge on the environment dynamics. Pioneering works in this vein include TD learning (Sutton and Barto, 2018), SARSA (Rummery and Niranjan, 1994), and Q-learning (Watkins and Dayan, 1992). However, these methods store value functions in a tabular form and cannot scale to large, complex problems, such as perimeter control. To overcome this issue, extensive efforts have been invested to integrate RL with deep learning (Lecun et al., 2015). Nevertheless, when nonlinear function approximators such as neural networks are combined with off-policy and bootstrapping, instability and divergence might occur (Tsitsiklis and Roy, 1997), i.e., the deadly triad issue (Sutton and Barto, 2018). Deep Q-Networks (DQN) algorithm (Mnih et al., 2015) was the first work that successfully addressed this using experience replay and target networks. Following its seminal success, an increasing number of DQN variants have been proposed (Hessel et al., 2017; Schaul et al., 2016; van Hasselt et al., 2015; Wang et al., 2015) and deep reinforcement learning (Deep-RL) has attracted substantial research interests in the scientific community.

Deep-RL has been applied by the transportation community for a variety of traffic control purposes, most notably signalized intersection control (Genders and Razavi, 2016; Li et al., 2016; Liang et al., 2018; Wei et al., 2019). An initial attempt to integrate RL into perimeter control problems can be found in (Ni and Cassidy, 2019). However, the RL method in this study was only utilized to redistribute metering rates derived from an MPC-based control framework along the cordon perimeters. In a separate study (Ni and Cassidy, 2020), a model-based RL method was incorporated in the MPC framework to solve the formulated open-loop optimization program as a replacement for the direct sequential method used in (Geroliminis et al., 2013). More recently, a neuro-dynamic programming approach was proposed in (Su et al., 2020) that integrates neural network function approximators and policy iteration. However, these components are only utilized to approximate the analytically found solution, which itself requires full system dynamics to obtain. In addition, the policy iteration technique is heavily model based since it models the transition function explicitly.

In this paper, a model free deep reinforcement learning perimeter control (MFDRLPC) scheme is presented. The proposed scheme learns the long-term values (i.e., Q-values) of state-action pairs iteratively through interactions with the environment, which allows the learning agents to make more sensible future decisions. Specifically, two different agent designs are introduced based on whether the control actions are discretized (referred to as D-RL) or treated as continuous (C-RL). The D-RL agent takes greedy actions based on the learned Q-values, while the C-RL agent features an actor to choose continuous control actions and a critic to evaluate the actions taken by

the actor. Both agent designs do not build on any information about the environment, like the functional form of the MFDs or impacts of vehicle routing decisions. Such strategy is called "model-free", a terminology widely adopted in the literature (Hou and Xiong, 2019; Lei et al., 2019; Li and Hou, 2020; Ren et al., 2020, 2018) that refers exclusively to the agent designs. Note that utilizing a simulation environment to train the agents—as done in this paper—would still require a model of traffic dynamics. Practically speaking, however, the agents can be trained in real life without this model information.

The remainder of this paper is outlined as follows. *Section 2* formulates the control problem for a two-region urban network with MFDs. *Section 3* explains the proposed MFDRLPC scheme in detail. *Section 4* presents the experiment setups and results. A discussion on the implementation prospect of the proposed scheme along with concluding remarks is provided in *Section 5*.

2. PROBLEM FORMULATION

This paper considers an urban network composed of two homogeneous sub-regions, $R_i, i = 1,2$; see Fig. 1(a). Traffic states are expressed by the accumulations $n_{ij}(t), i, j = 1,2$, which represent the number of vehicles within R_i with destinations in R_j at time t. Denote as $f_i(n_i(t))$ the MFD for R_i that defines the regional trip completion rate at accumulation $n_i(t)$ and note that $n_i(t) = \sum_j n_{ij}(t)$. Provided that the trip lengths are similar for all trips within a region, the internal and external trip completion rates for R_i are then respectively calculated by $M_{ii}(t) = n_{ii}(t)/n_i(t) \cdot f_i(n_i(t)),\ M_{ij}(t) = n_{ij}(t)/n_t(t) \cdot f_i(n_i(t)), i \neq j$, where $M_{ii}(t)$ stands for the regional exit flow and $M_{ij}(t)$ the transfer flow. Like accumulations, traffic demands are denoted by $q_{ij}(t), i, j = 1,2$. Note that estimates of traffic demands can be conveniently obtained from historical observations and are thus assumed to be known, but distinctions between these estimates and the ground truth demands might exist, which will be systematically examined in *Section 4.2.3*.

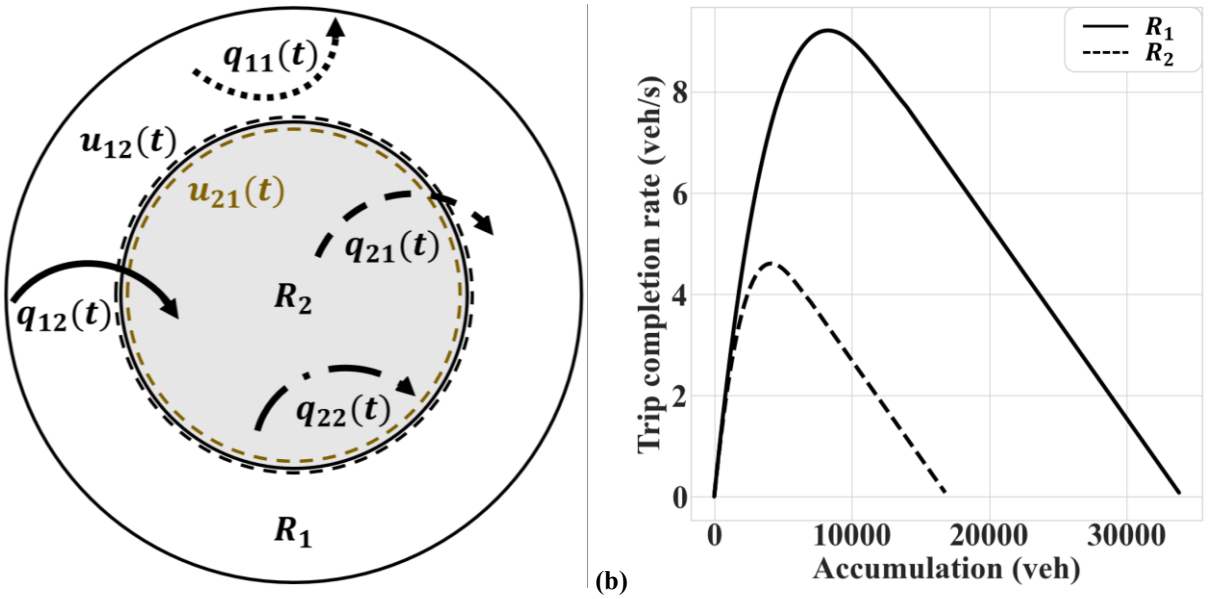


Fig. 1. Schematic diagrams of: (a) a two-region urban network; (b) two-region MFDs

The perimeter controllers are presumed to operate on the boundary between neighboring regions for the purpose of improving network throughput, i.e., the number of trips completed; see also Fig. 1(a). The controllers are denoted by $u_{12}(t)$ and $u_{21}(t)$, which respectively define the allowable portions of transfer flows $M_{12}(t)$ and $M_{21}(t)$. Note that the perimeter controllers cannot constrain internal flows $M_{11}(t)$ and $M_{22}(t)$. Further, it is assumed in this paper that the existence of these controllers will not alter the environment dynamics. Relaxation of this assumption can be found in (Haddad, 2017b; Ni and Cassidy, 2020).

Using this terminology, the two region perimeter control problem with MFDs is formulated as follows (similar to (Haddad et al., 2012)):

$$\max_{u_{12}(t),u_{21}(t)} \int_{t_0}^{t_f} [M_{11}(t) + M_{22}(t)]dt \tag{1}$$

subject to:

$$\frac{dn_{11}(t)}{dt} = q_{11}(t) + u_{21}(t) \cdot M_{21}(t) - M_{11}(t) \tag{2}$$

$$\frac{dn_{12}(t)}{dt} = q_{12}(t) - u_{12}(t) \cdot M_{12}(t) \tag{3}$$

$$\frac{dn_{21}(t)}{dt} = q_{21}(t) - u_{21}(t) \cdot M_{21}(t) \tag{4}$$

$$\frac{dn_{22}(t)}{dt} = q_{22}(t) + u_{12}(t) \cdot M_{12}(t) - M_{22}(t) \tag{5}$$

$$M_{ij}(t) = \frac{n_{ij}(t)}{n_i(t)} f_i\big(n_i(t)\big), i,j = 1,2 \tag{6}$$

$$n_{ij}(t) \geq 0, i,j = 1,2 \tag{7}$$

$$0 \leq n_{11}(t) + n_{12}(t) \leq n_{1,jam} \tag{8}$$

$$0 \leq n_{21}(t) + n_{22}(t) \leq n_{2,jam} \tag{9}$$

$$u_{\min} \leq u_{12}(t) \leq u_{\max} \tag{10}$$

$$u_{\min} \leq u_{21}(t) \leq u_{\max} \tag{11}$$

$$n_{ij}(t_0) = n_{ij,0}, i,j = 1,2 \tag{12}$$

where t_0 is the start time and t_f the final time, $n_{ij,0}$ are the known initial accumulations at t_0, $n_{i,jam}$ is the jam accumulation for R_i, and $u_{\min}$ and $u_{\max}$ are the lower and upper bounds for control actions, $u_{12}(t)$ and $u_{21}(t)$. Equation (1) provides the objective of the perimeter control problem, which is to maximize the cumulative number of trips completed. This objective is also equivalent to minimizing the total travel time in the system. Equation (2)-(6) describe the environment dynamics, which will be internalized by the proposed method and are not known or required for the controller design. Equation (7)-(9) and (10)-(11) are the boundary constraints for accumulations in both regions and control actions, respectively.

3. METHODOLOGY

This section first introduces the formulation of the perimeter control problem in the RL context. This is followed by a detailed explanation of the proposed MFDRLPC scheme that features two RL agents, C-RL and D-RL, with continuous and discrete action spaces, respectively. Finally, this

section provides an overview of the model predictive control method that is to be compared with the proposed scheme.

3.1 RL formulation

The standard RL formulation involves an agent interacting with an environment at discrete time intervals. At each time step t, the agent observes a numerical state of the environment s_t, takes an action a_t according to a policy π, and receives a scalar reward r_{t+1} from the environment. The return is defined as the cumulative discounted future reward from time step t, which might be stochastic. The goal of the RL agent is to learn a policy that maximizes the expected return from the start time. Formally, the optimal perimeter control problem is modeled as a Markov decision process characterized by a tuple $< \boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{R}}, \boldsymbol{\pi}, \boldsymbol{\gamma} >$.

- **State space, $\boldsymbol{\mathcal{S}}$**. For the C-RL agent, the state consists of four accumulations $n_{ij}, i,j = 1,2$ and four estimated traffic demands $q_{ij}, i,j = 1,2$ averaged over the next time step. Each element of the state is scaled to [0,1] through a division by its maximum value. For the D-RL agent, the implemented controller values at the previous time step are also incorporated in the state design.

- **Action space**, $\boldsymbol{\mathcal{A}}$. For the C-RL agent, the action is comprised of two real values from the allowable range $[u_{\min}, u_{\max}]$ for the perimeter controllers u_{12} and u_{21}. For the D-RL agent, three actions are defined for each of the two perimeter controllers: 1) increase its value by Δu; 2) keep its value unchanged; or, 3) decrease its value by Δu, where Δu is a predetermined amount indicating the allowable change in the perimeter controller values. Changing controller values by a set amount allows for gradual change in control over time. In total, the D-RL agent has 9 actions to choose from (three options for each of the two controllers). After an action is chosen by the agents, it is implemented in the environment for the duration of a time step, Δt.

- **Transition dynamics**, $\boldsymbol{\mathcal{P}}$. The environment receives an action from the agent and arrives at a new state according to the transition dynamics $\mathcal{P}(s_{t+1}|s_t, a_t): \mathcal{S} \times \mathcal{A} \to \mathcal{S}$. In this work, the transition dynamics are expressed by the traffic dynamics described in *Section 2* and are not explicitly modeled.

- **Reward function**, $\boldsymbol{\mathcal{R}}$. After taking an action, the agent receives an immediate reward from the environment as an assessment of the action just taken, according to the reward function $r(s_t, a_t): \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. The reward is defined as $(M_{11}(t) + M_{22}(t))/C$, where C is a large constant utilized to normalize the reward. In this paper, we purposefully normalize the reward to $[0, 1]$, as suggested by (Henderson et al., 2017). The agent is trained to maximize the cumulative number of trips completed, which is identical to the objective function presented in *Section 2*. Further, a large negative quantity is added to the reward to penalize unreasonable control strategies that precipitate gridlock or invalid accumulations.

- **Policy**, $\boldsymbol{\pi}$, and **discount factor**, $\boldsymbol{\gamma}$. At each time step, the agent chooses an action based on a policy that maps states to actions, i.e., $\pi: \mathcal{S} \to \mathcal{A}$. The return G_t measures the long-term quality of the policy and is defined as the total discounted reward from time step t

$$G_t = \sum_{\tau=t}^{T} \gamma^{\tau-t} r_{\tau+1} \tag{13}$$

where T is the total time steps of an episode and $\gamma \in [0,1]$ is a discount factor utilized to impose more values on immediate reward above delayed reward. Owing to the potential stochasticity in the policy, the purpose of the RL agent is to maximize the expected value of the return, $\mathbb{E}[G_t]$, from the start time, which in this paper represents the expected total number of trips completed from t_0.

3.2 Algorithms

3.2.1 Discrete agent (D-RL)

Following the convention of model free control methods, the action value function $Q(s,a)$ is used, which implies the expected return after taking action a at state s and then following policy π. The action value function follows the recursive relationship knowns as the Bellman Equation (Sutton and Barto, 2018):

$$Q(s_t, a_t) = \mathbb{E}_\pi\left[r_{t+1} + \gamma Q\left(s_{t+1}, \pi(s_{t+1})\right)\right] \tag{14}$$

where $\pi(\cdot)$ represents the policy to be learnt.

Q-learning (Watkins and Dayan, 1992), a popular off-policy model free algorithm, uses the greedy policy (i.e., $\pi(s) = \arg\max_a Q(s,a)$) and updates the action value function in a tabular manner as per (14). However, this algorithm cannot be scaled up for complex problems with large state and/or action spaces since explicit storage of all possible state-action pairs is likely intractable. Therefore, function approximation has been proposed as a method for generalization to directly map state-action pairs to action values (also known as Q-values). Nonlinear function approximators have long been avoided due to the potential instability and divergence (Sutton and Barto, 2018; Tsitsiklis and Roy, 1997). The Deep Q-Networks (DQN) algorithm (Mnih et al., 2015) marks the first success to have achieved consistent convergence while using large neural networks as function approximators. A schematic diagram of the DQN algorithm is presented in Fig. 2. Particularly, two major components have contributed to its ability to consistently converge: experience replay (Lin, 1992) and target network. The principal idea behind experience replay is to store the collected experiences in a replay buffer and apply Q-learning update rule on sampled batches of experiences. In this manner, the collected experiences could be used multiple times, improving sample efficiency significantly. Further, the sampling of experiences helps reduce correlations between the transitions utilized to update the Q-network, thus enhancing stability in the learning process. The target network shares the same structure as the Q-network, but its weights are periodically updated from the Q-network weights. Therefore, the learning targets are roughly static throughout the training process. Adjusting the Q-network towards static learning targets mimics supervised learning and has been shown to produce more stable learning processes.

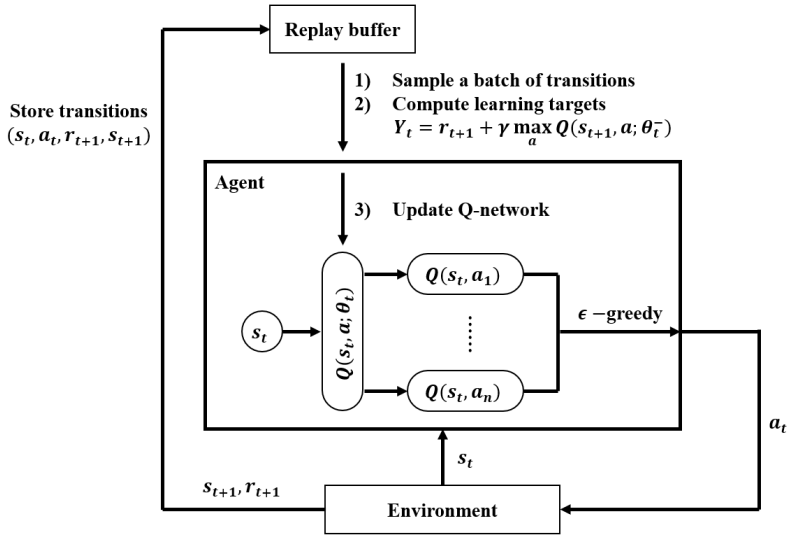


Fig. 2. Schematic diagram of the DQN algorithm. $Q(:,:;\theta_t)$ and $Q(:,:;\theta_t^-)$ are the Q- and target networks, respectively. $\epsilon-$ greedy is the exploration strategy. n is the number of actions ($n = 9$ in this work).

Note that the greedy policy is adopted by DQN similarly to Q-learning. Hence, the same value (i.e., $\max_a Q(s,a)$) is used to both select an action and evaluate it (see the learning targets in Fig. 2), which is likely to result in an overestimation of action values. This overestimation problem was first observed in (Thrun and Schwartz, 1993) and later affirmatively studied in (van Hasselt et al., 2015). In the latter reference, the authors proposed to decouple the max operation in the greedy policy into action selection and evaluation. Combining Double Q-learning (van Hasselt, 2010) with advances of the DQN, a new algorithm named Double DQN was developed, where the Q-network is used for action selection and the target network for evaluation. In mathematical term, the learning target used by Double DQN is as follows

$$Y_t = r_{t+1} + \gamma Q\left(s_{t+1}, \arg\max_a Q(s_{t+1}, a; \boldsymbol{\theta}_t); \boldsymbol{\theta}_t^-\right) \tag{15}$$

where $Q(:,:;\boldsymbol{\theta}_t)$ and $Q(:,:;\boldsymbol{\theta}_t^-)$ stand for the parameterized Q- and target networks, respectively. Substituting the learning targets in Fig. 2 with (15), one obtains the Double DQN algorithm, which is adopted as the learning algorithm for the D-RL agent in the present paper.

Value-based methods such as DQN and Double DQN can only cope with discrete or low-dimensional action spaces. Hence, the perimeter controller values are discretized for the D-RL agent as described in *Section 3.1*. The $\epsilon-$ greedy strategy is utilized for the D-RL agent to explore the state and action spaces. Specifically, with predicted Q-values from the Q-network, the agent takes the greedy action (i.e., $\arg\max_a Q(s,a)$) with probability $1-\epsilon$ and takes a random action with probability ϵ. To achieve sublinear total opportunity loss during exploration, the ϵ value is decayed throughout the learning process. The specific decay schedule will be presented in due course. The Q-network (hence the target network) is constructed as a multilayer perceptron that takes as input the state and outputs a 9-dimensional vector representing the Q-values for each state-action pair (the action space of the D-RL agent is 9-dimensional). More concretely, the Q-network has two 64-unit dense layers with ReLU activations. The outputs of these two layers are then connected to a 9-unit dense linear layer. All dense layers initialize their weights based on a normal distribution

with default parameters. The optimization algorithm employed is Adam (Kingma and Ba, 2015) with epsilon set to 10^{-8}.

3.2.2 Continuous agent (C-RL)

While the discretization strategy provides a straightforward approach to handle continuous action spaces, it still possesses numerous limitations, most notably the curse of dimensionality. Specifically, the number of discrete actions increases considerably with the granularity of discretization and finer grained control is likely intractable as the action space is too large to be explored efficiently. Moreover, such discretization cannot be scaled up to multi-regions since the dimension of the action space increases exponentially with the number of regions. In contrast, continuous control schemes can more flexibly choose perimeter controller values, making fine control possible. Note that, though perimeter controllers are typically implemented by traffic signals with discrete timing plans, continuous action designs are still an unspoken convention in the literature. In fact, continuous perimeter control actions render its implementation by traffic signal more feasible as the signal timing plans do not need to be adjusted drastically across time steps. Moreover, continuous control schemes are also scalable to multi-region networks, as shown in (Lei et al., 2019; Ramezani et al., 2015; Ren et al., 2020; Sirmatel and Geroliminis, 2018). The authors hence present the C-RL agent that assumes a continuous action space. Also note that the proposed agent has the same number of control variables as any other continuous control method and thus in theory can be applied to multi-region networks as well. However, showing so is beyond the scope of the present work.

Unlike value-based methods that adopt a greedy policy based on Q-values, policy gradient methods (Sutton et al., 2000) explicitly parameterize the policy so that continuous control actions can be conveniently obtained. Optimization methods such as gradient ascent are then utilized to directly optimize the policy to achieve higher long-term value (e.g., return). Specifically, a policy objective function can be defined to evaluate the parameterized policy, $\pi_\theta(a|s): \mathcal{S} \to P(\mathcal{A})$, which represents a stochastic distribution over actions. Since the policy is dependent only on the parameters θ, the policy objective function is expressed as a function of the parameters, denoted as $J(\theta)$. The parameters are then updated iteratively in the direction of the policy objective function gradient $\nabla_\theta J(\theta)$ to maximize $J(\theta)$. The fundamental result behind these methods is the (stochastic) policy gradient theorem (Sutton et al., 2000)

$$\nabla_\theta J(\theta) = E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s)\, Q^{\pi_\theta}(s,a)] \tag{16}$$

which states the policy gradient irrespective of the policy objective function used. However, the estimation of the stochastic policy gradient involves an integral over both state and action spaces, thus requiring a large number of samples. For this reason, (Silver et al., 2014) modeled the policy as a deterministic decision $a = \mu_\theta(s)$ and proposed the deterministic policy gradient,

$$\nabla_\theta J(\theta) = E_{\mu_\theta}\left[\nabla_\theta \mu_\theta(s) \nabla_a Q^{\mu_\theta}(s,a)|_{a=\mu_\theta(s)}\right] \tag{17}$$

which is the expected gradient of the action value function with respect to parameters θ. Since the deterministic policy gradient only integrates over the state space, it can be estimated much more efficiently than its stochastic counterpart (16), especially with high-dimensional action spaces. To enhance exploration of the state and action space, an off-policy deterministic actor-critic algorithm was proposed in (Silver et al., 2014). This algorithm was subsequently extended in (Lillicrap et al., 2016) where the actor and critic are modeled with neural networks. Integrating advances of the

DQN, the latter paper proposed the model-free, off-policy actor-critic Deep Deterministic Policy Gradient (DDPG) algorithm, which is the learning algorithm adopted for the C-RL agent in this paper. The DDPG algorithm is schematically presented in Fig. 3.

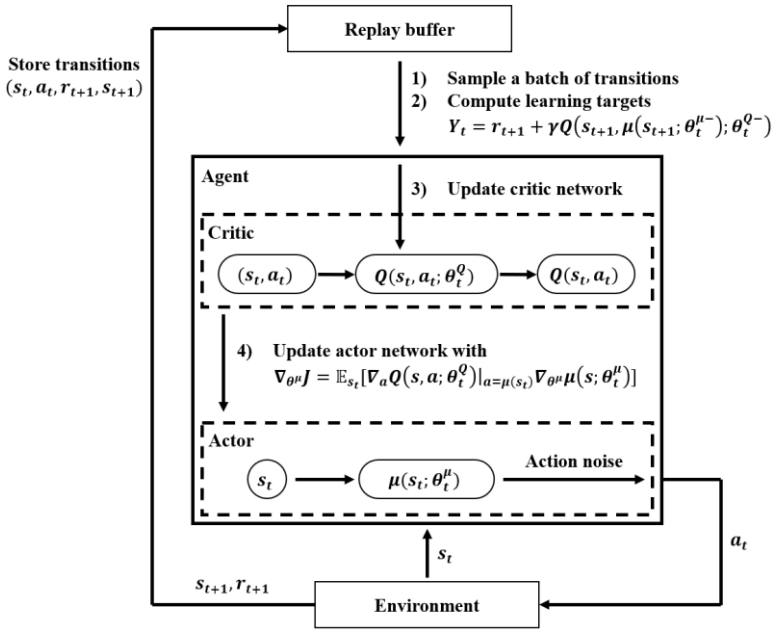


Fig. 3. Schematic diagram of the DDPG algorithm. $\mu(:;\boldsymbol{\theta}_t^{\mu})$ and $Q(:,:;\boldsymbol{\theta}_t^{Q})$ are respectively the actor and critic networks, while $\mu(:;\boldsymbol{\theta}_t^{\mu^-})$ and $Q(:,:;\boldsymbol{\theta}_t^{Q^-})$ are the target actor and critic networks.

Actor-critic methods (Bhatnagar et al., 2009; Degris et al., 2012; Grondman et al., 2012) combine the advantages of both value-based and policy gradient methods. These methods have an actor to take actions (i.e., policy) and a critic to evaluate the actions taken by the actor (i.e., value functions). Two sets of parameters are separately maintained for the actor and critic. The critic parameters are updated in a similar manner to Q-learning (Watkins and Dayan, 1992) while the actor parameters are updated in the direction suggested by the critic. Correspondingly, DDPG has an actor, which maps states to actions, and a critic, which maps state-action pairs to Q-values. The learning target for the critic is

$$Y_t = r_{t+1} + \gamma Q\left(s_{t+1}, \mu\left(s_{t+1}; \boldsymbol{\theta}_t^{\mu^-}\right); \boldsymbol{\theta}_t^{Q^-}\right) \tag{18}$$

where $\mu(:;\boldsymbol{\theta}_t^{\mu^-})$ and $Q(:,:;\boldsymbol{\theta}_t^{Q^-})$ are target networks for the actor and critic, respectively. After updating the critic parameters, the deterministic policy gradient is then computed (as per step 4) in Fig. 3), which is utilized to update the actor parameters with gradient ascent.

It can be observed from the comparison between (15) and (18) that, the DDPG algorithm directly parameterizes the policy whereas the Double DQN algorithm obtains its policy from predicted Q-values. For problems with high-dimensional or even continuous action spaces, direct parameterization will be particularly advantageous as the argmax operation in (15) becomes an optimization problem that is too computationally expensive to solve at each time step. Discretization of the action space could help alleviate some of the computational burden. However, structure of the action space will be lost to some extent. Therefore, it can be expected that the C-RL agent will significantly outperform the D-RL agent for the perimeter control problem under study, which will be demonstrated in *Section 4.*

For the C-RL agent, both the actor and critic networks are constructed as multilayer perceptron, a structure commonly seen in DDPG-related studies (Horgan et al., 2018; Lillicrap et al., 2016) that is suitable for low-dimensional input-output relationships considered in this work. The actor network takes as input an 8-dimensional state vector that contains the set of four accumulations and the averages of four estimated traffic demands. These inputs are then connected to two 64-unit dense layers with ReLU activations. Outputs of these dense layers are projected to a 2-dimensional space with a dense tanh layer. Finally, a Lambda layer is used to bound the outputs between u_{min} and u_{max}, thus producing a 2-dimensional action vector containing the perimeter controller values. Moreover, unbiased noise drawn from a normal distribution is directly injected into the actions to facilitate exploration of the state and action spaces. The scale parameter of the normal distribution is decayed in the learning process since less exploration is needed as the agent gathers more information about the environment. The critic network receives the state and action vector and passes these inputs through two 64-unit dense layers with ReLU activations. Outputs of these layers are subsequently connected to a dense linear layer that has only 1 unit representing the Q-value for the input state-action pair. The weights of all dense layers are initialized from a normal distribution with default parameters. Target networks share the same structure as the corresponding actor and critic networks. Adam (Kingma and Ba, 2015) is utilized to optimize the loss for the critic network with epsilon set to 10^{-8}.

3.2.3 Formalization of the MFDRLPC scheme

In this work, the proposed model free C-RL and D-RL agents learn entirely from interactions with the environment and a significant amount of experiences are needed for efficient learning. For this purpose, the distributed architecture Ape-X (Horgan et al., 2018) is combined with the learning algorithms to collect more experiences. More specifically, the Ape-X architecture maintains numerous experience generators and a single centralized learner. Each generator has its own instance of the environment and is assigned different exploration strategies to expand the amount of experiences they jointly encounter. These experiences are then stored in a fixed-size replay buffer in a first in first out order, i.e., prioritization of experiences is performed based only on recency as opposed to TD errors (Horgan et al., 2018; Schaul et al., 2016). In this manner, the experiences where the agents are making more educated decisions are valued more than outdated ones, which are removed from the buffer once the buffer size is exceeded by the amount of collected experiences. The centralized learner uniformly samples experiences from the shared buffer, which are utilized to update the neural networks of the agents. For the C-RL agent, all generators assume unbiased Gaussian noise with decaying parameter for the normal distribution. The sampled experiences are used to update the actor and critic networks. At convergence, the learned policy is explicitly represented by the fully trained actor network. For the D-RL agent, all generators use decaying ϵ-greedy strategy for exploration and the sampled experiences are used to update the Q-network. At convergence, the learned policy can be derived by taking greedy actions with respect to the learned Q-values. The proposed C-RL and D-RL agents of the model free deep reinforcement learning perimeter control (MFDRLPC) scheme are formalized in Algorithm 1 and 2, respectively.

Algorithm 1. Model Free Deep Reinforcement Learning Perimeter Control: C-RL Agent

Randomly initialize actor network $\boldsymbol{\theta}_0^{\mu}$ and critic network $\boldsymbol{\theta}_0^{Q}$
Initialize target actor and critic networks $\boldsymbol{\theta}_0^{\mu-} = \boldsymbol{\theta}_0^{\mu}, \boldsymbol{\theta}_0^{Q-} = \boldsymbol{\theta}_0^{Q}$
Initailize replay buffer **B**, buffer size *M*, sample size *m*, iteration number *I,* genetaor number *G*
for *iter* = 1 **to** *I* **do**
    Initialize a normal distribution *N* for action exploration with decayed scale parameter
    for generator = 1 **to** *G* **do**
        Load actor network $\boldsymbol{\theta}_{iter}^{\mu} = \boldsymbol{\theta}_{iter-1}^{\mu}$
        $s_0 \leftarrow$ Environment.Reset()
        for $t = 1$ **to** *T* **do**
            $a_{t-1} = \mu(s_{t-1};\ \boldsymbol{\theta}_{iter}^{\mu}) + N_{t-1}$
            $(r_t, s_t) \leftarrow$ Environment.Step(s_{t-1}, a_{t-1})
            B.add$((s_{t-1}, a_{t-1}, r_t, s_t))$
        end for
    end for
    if B.size() > *M* **then**
        B.remove()
    end if
    Training samples ← **B**.sample(*m*)
    Periodically load target actor and critic networks $\boldsymbol{\theta}_{iter}^{\mu-} = \boldsymbol{\theta}_{iter-1}^{\mu}, \boldsymbol{\theta}_{iter}^{Q-} = \boldsymbol{\theta}_{iter-1}^{Q}$
    $\boldsymbol{\theta}_{iter}^{Q} \leftarrow$ Update the critic network towards learning target (18)
    $\boldsymbol{\theta}_{iter}^{\mu} \leftarrow$ Update the actor network using the sampled policy gradient (17)
end for

Algorithm 2: Model Free Deep Reinforcement Learning Perimeter Control: D-RL Agent

Randomly initialize Q-network $\boldsymbol{\theta}_0$ and target Q-network $\boldsymbol{\theta}_0^- = \boldsymbol{\theta}_0$
Initialize replay buffer **B**, buffer size M, sample size m, iteration number I, generator number G
for *iter* = 1 **to** I **do**
    Decay the ϵ value for $\epsilon-$ greedy exploration
    for generator = 1 **to** G **do**
        Load Q-network $\boldsymbol{\theta}_{iter} = \boldsymbol{\theta}_{iter-1}$
        $s_0 \leftarrow$ Environment.Reset()
        for $t = 1$ **to** T **do**
            $a_{t-1} = \arg\max_a Q(s_{t-1}, a; \boldsymbol{\theta}_{iter})$ with probability $1-\epsilon$
                a random action with probability ϵ
            $(r_t, s_t) \leftarrow$ Environment.Step(s_{t-1}, a_{t-1})
            B.add$((s_{t-1}, a_{t-1}, r_t, s_t))$
        end for
    end for
    if B.size() $>$ M **then**
        B.remove()
    end if
    Training sample $\leftarrow$ **B**.sample(m)
    Periodically load target network $\boldsymbol{\theta}_{iter}^- = \boldsymbol{\theta}_{iter-1}$
    $\boldsymbol{\theta}_{iter} \leftarrow$ Update Q-network towards learning target (15)
end for

The list of important hyperparameters for the C-RL and D-RL agents is presented in Table 1. Note that, a systematic grid search of hyperparameter values is not conducted due to the high computational burden. However, as will be shown, the proposed scheme is able to compete with the state-of-the-art MPC method even with randomly chosen hyperparameters. This demonstrates the significant potential of Deep-RL methods on MFD-based network traffic control. On the other hand, with systematically tuned hyperparameters, the proposed agents will be trained in a more specialized manner, which will increase performance but might be more prone to overfitting and reduced transferability.

Table 1. List of hyperparameters and their values

Hyperparameter	Value C-RL	Value D-RL	Description
Iteration number (I)	250	250	The number of iterations to train the agents
Generator number (G)	32	32	The number of generators used to collect experiences
Replay buffer size (M)	10000	10000	Storage capacity of the replay buffer
Sample size (m)	1000	1000	The number of transitions sampled for network updates
Initial scale parameter	0.3	-	Initial scale of normal distribution for action noise
Action Noise scale decay	0.001	-	Linear exploration decay in each iteration
Final scale parameter	0.05	-	Final scale of normal distribution for action noise
Initial ϵ	-	0.8	Initial value of ϵ in $\epsilon-$ greedy exploration
ϵ decay	-	0.98	Exponential exploration decay factor in each iteration
Final ϵ	-	0.01	Final value of ϵ in $\epsilon-$ greedy exploration
Critic epoch	128	-	Number of forward and backward pass for sampled transitions
Initial critic learning rate	0.001	-	The initial learning rate used by Adam for the critic
Critic learning rate decay	0.98	-	Exponential learning rate decay factor in each iteration
Actor epoch	2	-	The times gradient ascent is executed for the actor
Initial actor learning rate	0.0025	-	The initial learning rate used by gradient ascent for the actor
Actor learning rate decay	0.93	-	Exponential learning rate decay factor in each iteration
Q-network epoch	-	128	Number of forward and backward pass for sampled transitions
Initial learning rate	-	0.001	The initial learning rate used by Adam for the Q-network
Learning rate decay	-	0.95	Exponential learning rate decay factor in each iteration
Minimum learning rate	0.0001	0.0001	The minimum learning rate for the actor, critic, and Q-network
Batch size	256	128	The number of transitions to update the networks once
Discount factor	0.95	0.8	Discount factor γ used to compute the learning targets
Target networks lifetime	5	5	The number of iterations to periodically update target networks
Early stopping patience	20	20	The number of epochs with no improvement to stop training

3.3 Model Predictive Control

The proposed scheme is compared against the state-of-the-art method, i.e., model predictive control (MPC) (Geroliminis et al., 2013). MPC is an advanced model-based control method which assumes sufficient knowledge of environment dynamics. Using a closed-loop framework, the MPC can accommodate discrepancy between the prediction model and plant (reality). At each time step, an open-loop of the control problem is formulated as a nonlinear program (NLP). Controller values are obtained by solving the formulated NLP and only the first controller is implemented into the plant. This procedure is iteratively carried out until termination of the control period.

To formulate the open-loop control problem into an NLP problem, a prediction horizon N_p and a control horizon N_c are assumed. The MPC controller assumes piecewise constant controls for each step in the prediction horizon. To reduce computational complexity, the perimeter controllers are only allowed to change for N_c steps and are kept fixed at the last control values thereafter. Specifically, in the context of perimeter control, at each time step, the MPC controller predicts the evolution of accumulations for the next N_p steps based on the initial accumulations, theorized MFDs prediction model, and control variables. An objective function is then constructed to maximize the throughput of the network depending on how the accumulations change. The objective function and corresponding constraints (e.g., path constraints, vehicle conservation) are then transformed to the standard form of an NLP. The formulated NLP could be solved using popular NLP solvers such as IPOPT (Wächter and Biegler, 2006). In the present work, the NLP is

solved using the Sequential Least SQuares Programming (SLSQP) method (Kraft, 1988), a wrapper of the Sequential Quadratic Programming method (Nocedal and Wright, 2006) that can handle any combination of bounds, inequality, and equality constraints. The obtained solutions provide values for the perimeter controllers and the first one is carried out in the plant. Note that, the MFDs used in the prediction model are accurate while the MFDs in the plant might exhibit uncertainty and scatter, especially as the network becomes more congested.

In this paper, the MPC is implemented according to (Geroliminis et al., 2013) without adding smoothing control constraints to provide best-case performance for comparison. Both the prediction and control horizons are set to 20. The selection of the prediction horizon is in line with previous research studies that apply the MPC to perimeter control problems (Geroliminis et al., 2013; Hajiahmadi et al., 2015; Ramezani et al., 2015). The control horizon is selected to maximize the performance of the MPC.

4. EXPERIMENTS

In this section, the proposed MFDRLPC scheme is compared with the MPC to perform perimeter control in a two-region urban network as in Fig. 1(a). Note that the major benefit of the proposed method is that it does not build on any information about the environment dynamics, whereas the MPC assumes full knowledge of regional traffic dynamics. Comparison between the C-RL and D-RL agents is also provided, which aims to verify the advantage of continuous control scheme over discretization of actions in high-dimensional control problems. The discretization is conducted by setting the predetermined allowable change in perimeter controller values to $\Delta u = 0.1$. In addition, the no control (NC) strategy is also included for comparison. Under the NC strategy, transfer flows will not be restricted and the perimeter controllers are set to their maximum values for the whole control period, i.e., $u_{12} = u_{21} = u_{\max}$. This strategy provides a baseline for other methods whose performances can be expressed as relative improvements over it.

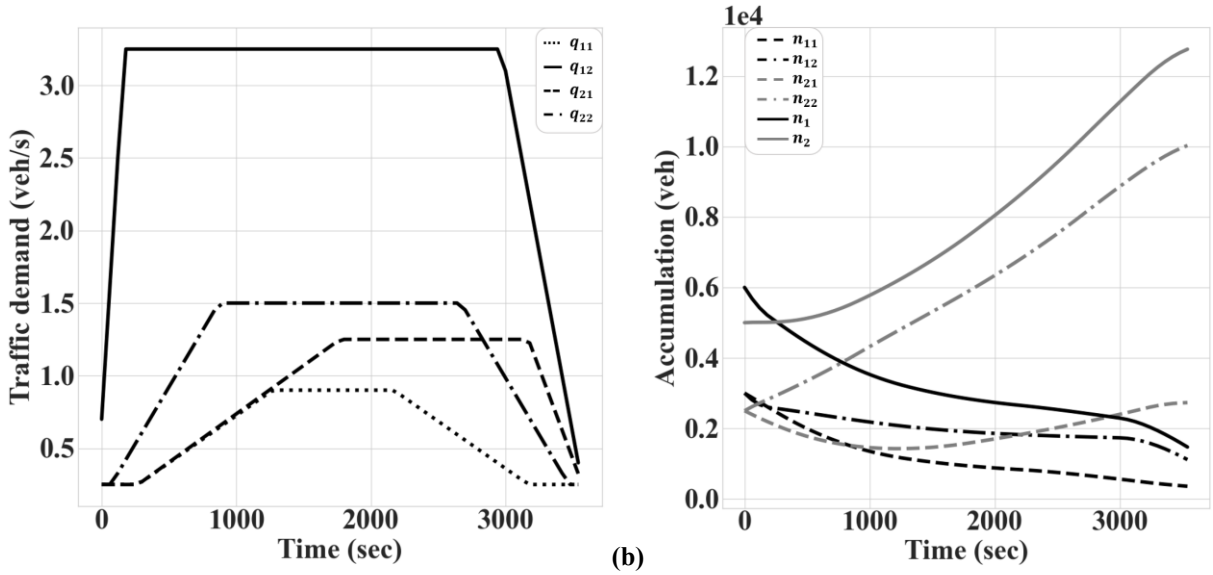
4.1 Experiment setup

For the two-region network considered in this paper, the MFD for R_1 is the same as the one observed in Yokohama, Japan[1] (Gao and Gayah, 2018; Geroliminis and Daganzo, 2008), whereas a scaled-down version of it is adopted for R_2 that corresponds to a smaller region (e.g., a city center); see Fig. 1(b). The critical accumulations associated with maximum productivity are $n_{1,cr} = 8241$ veh and $n_{2,cr} = 4120$ veh for R_1 and R_2, respectively. Note that the MFDs do not need to be exact and the impact of modeling errors in the MFDs will be explicitly examined in *Section 4.2.3*.

The estimated traffic demand profile is illustrated in Fig. 4(a), which exhibits higher demand to R_2 (the simulated city center) than to R_1 (the periphery of a city) during the 1-hour period. The demand profile presented here represents traffic conditions during morning peak. In

[1] The functional form of the MFD selected here is piecewise: a third-order polynomial when the region is not extremely congested and linear for extreme congestion. Tests using strictly third-order polynomial for the entire range of densities reveal that this choice does not negatively impact the performance of the MPC method. The piecewise MFD is adopted since it more accurately represents macroscopic traffic dynamics.

subsequent experiments, more extreme demand patterns will be considered as well. The initial traffic states in R_1 and R_2 are respectively assumed to be uncongested and congested with $n_1(t_0) = 6000$ veh and $n_2(t_0) = 5000$ veh.



(a) **(b)**

Fig. 4. (a) estimated traffic demands; (b) accumulations under the NC strategy

Perimeter controllers are often implemented by signalized intersections along the perimeter borders. Due to the loss time of signal timing plans, transfer flows may not be fully accommodated by the controllers, hence the maximum controller value is fixed at $u_{\max} = 0.9$. On the other hand, a complete prohibition of transfer flows can hardly be enforced in a realistic manner, thus the minimum control value is set to $u_{\min} = 0.1$. Further, the signalized intersections are assumed to have a uniform cycle length of 60s, so the duration of a time step is $\Delta t = 60s$.

With these settings, the evolution of accumulations under the NC strategy is presented in Fig. 4(b), which shows a steadily decreasing accumulation in R_1 and increasing accumulation in R_2. Note that R_1 is initially uncongested. Thus, reduced accumulation decreases its regional production. On the other hand, R_2 is initially congested and the increasing accumulation leads to consistently reduced production. As a consequence, the network throughput is relatively low under the NC strategy. In subsequent sections, it will be shown how the network throughput can be effectively improved by properly implementing perimeter control in the system to mitigate the unbalanced traffic condition.

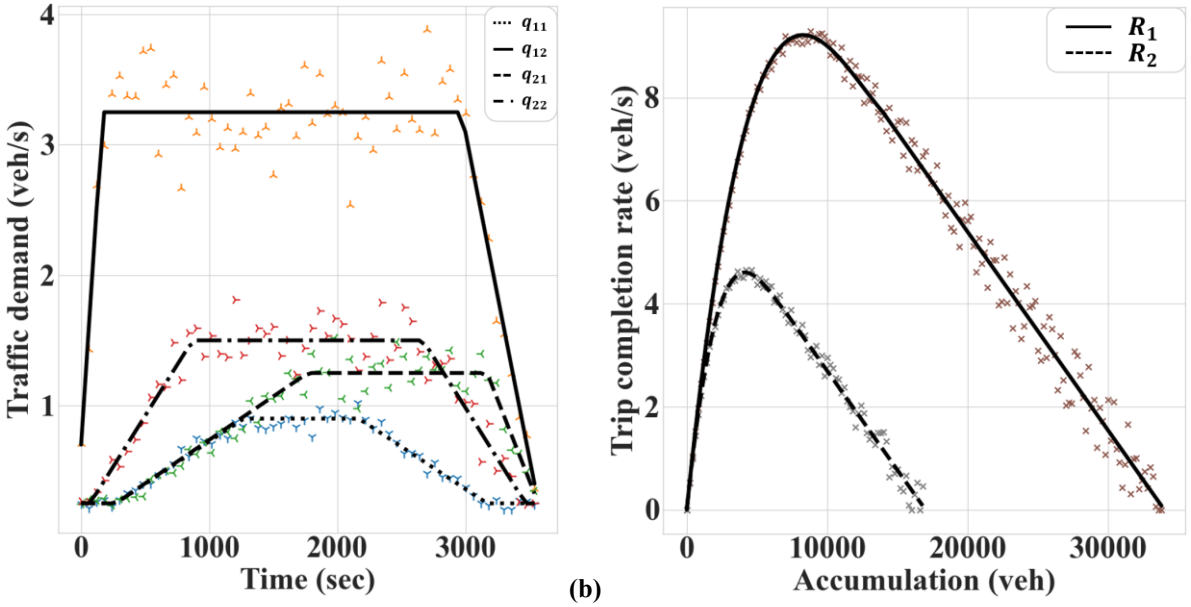
The simulation environment used in the present paper is a realistic two-region MFDs plant with potential uncertainty in the MFDs and/or traffic demand, as described in (Geroliminis et al., 2013). Specifically, the MFDs and traffic demands in the environment are expressed by

$$\tilde{f}_i\big(n_i(t)\big) = f_i\big(n_i(t)\big) + \varsigma(t) \cdot n_i(t), i = 1,2 \tag{19}$$

$$\tilde{q}_{ij}(t) = \max\left(q_{ij}(t) \cdot (1 + \varepsilon(t)), 0\right), i, j = 1,2 \tag{20}$$

where $\varsigma(t) \sim U(-\alpha, \alpha)$ and $\varepsilon(t) \sim N(0, \sigma^2)$. The predefined parameters α and σ are separately the level of uncertainty in the MFDs and traffic demands. Under this definition, the uncertainty in the MFDs is proportional to current accumulations. As a region becomes more congested, the errors between the expected and realized trip completion rates grow, which is consistent with empirical

and analytical findings (Buisson and Ladier, 2009; Daganzo et al., 2011). The uncertainty in traffic demands represents temporal fluctuations and is proportional to the estimated traffic demands. For the same value of σ, the magnitude of potential errors in demand would increase with the estimated traffic demands. The level of uncertainty in the environment with $\sigma = 0.1, \alpha = 0.1$ is illustrated in Fig. 5(a) and (b) for traffic demands and MFDs, respectively. The simulation environment is then obtained by replacing MFDs and demand values in Equation (2)-(6) with terms (19)-(20). The proposed C-RL and D-RL agents learn the long-term values (i.e., Q-values) for various actions taken at each state via interaction with the simulation environment. The agents also internalize the environment dynamics via this interaction. Specifically, the agents receive states and rewards from the environment while the environment implements actions taken by the agents.



(a) (b)

Fig. 5. Environment uncertainty: (a) estimated traffic demands; (b) two-region MFDs

It is worth reiterating that, while the agent designs of the proposed MFDRLPC scheme are model free, the environment still involves a model, i.e., the MFD-based traffic dynamics. However, the use of the MFD model to construct the simulation environment or I/O data generator is not atypical when applying model free methods for perimeter control; for example, see (Lei et al., 2019; Li and Hou, 2020; Ren et al., 2020, 2018).

The objective of perimeter control is to improve network production, i.e., cumulative trip completion (CTC), which equivalently is to minimize the total travel time (TTT) in the system. Therefore, these two quantities, CTC and TTT, are used to evaluate the performances of the four perimeter control strategies: NC, MPC, and the proposed C-RL and D-RL agents. All strategies are applied for perimeter control under various environment configurations. The environment configurations considered in this paper are arranged according to the level of environment uncertainty, with more details provided in Table 2. Note that while the initial accumulations are the same among different environment configurations, their impacts will be thoroughly investigated in *Section 4.2.3*.

Table 2. Environment configurations under consideration

Env. No.	σ	α	Description
1	0	0	Deterministic scenario as a benchmark
2	0.1	0	Medium uncertainty in traffic demands
3	0.2	0	High uncertainty in traffic demands
4	0	0.1	Medium uncertainty in MFDs
5	0.1	0.1	Medium uncertainty in traffic demands and MFDs
6	0.2	0.1	High uncertainty in traffic demands and medium uncertainty in MFDs
7	0	0.2	High uncertainty in MFDs
8	0.1	0.2	Medium uncertainty in traffic demands and high uncertainty in MFDs
9	0.2	0.2	High uncertainty in traffic demands and MFDs

4.2 Experiment results

4.2.1 Convergence consistency of the MFDRLPC scheme

Performance curves of the four methods under all environment configurations are presented in Fig. 6. Each curve provides the evolution of CTC across consecutive training iterations. The C-RL and D-RL agents were trained with numerous random seeds to fully gauge their ability to consistently converge, as suggested in (Henderson et al., 2017). The darker line in Fig. 6 represents the median training performance over random seeds while the shaded areas are obtained by plotting the two extreme values in each iteration and filling the areas in between. Reporting the mean results does not affect the conclusions to be presented. However, since the mean is more sensitive to extreme values than the median, they are not included. Results for the NC and MPC in Fig. 6 are obtained by running both strategies multiple times and reporting the median and extreme values. The NC and MPC are not learning-based methods, hence their performances are relatively fixed across different runs and environments, as indicated by the flat darker line and narrow shaded band.

The shaded areas in Fig. 6 are neither representative of the level of uncertainty in the environment nor the fluctuations in the training process. Instead, they represent the randomness experienced by the agents. As an example, suppose the agents collected disastrous experiences in the deterministic environment (i.e., Env. No. 1), then their performance curves are likely to exhibit a rather poor lower bound. Contrarily, if the agents happen to follow a large quantity of desirable trajectories while exploring in the high uncertainty environment (i.e., Env. No. 9), then they will likely achieve decent lower-bound performances. This would occur even though the level of uncertainty in the deterministic environment is the lower. In general, training fluctuations are better captured by the median curves as they are illustrative of the normalized performances and are comparable across different environments. The seemingly fluctuating median performance curves are not atypical; for example, see (Horgan et al., 2018; Lillicrap et al., 2016). As shown in Fig. 6, the median performance curves of the proposed C-RL and D-RL agents appear to be noisier with increasing uncertainty in the traffic demands (column-wise comparison) but remain relatively invariant to uncertainty in the MFDs (row-wise comparison). This is reasonable since the magnitude of uncertainty in the demands, which is proportional to flows, is much larger than that of uncertainty in the MFDs, which is proportional to accumulations. Comparing across the median

performance curves of the proposed agents, one could observe that the training processes of the C-RL agent exhibit greater fluctuations than the D-RL agent before convergence.

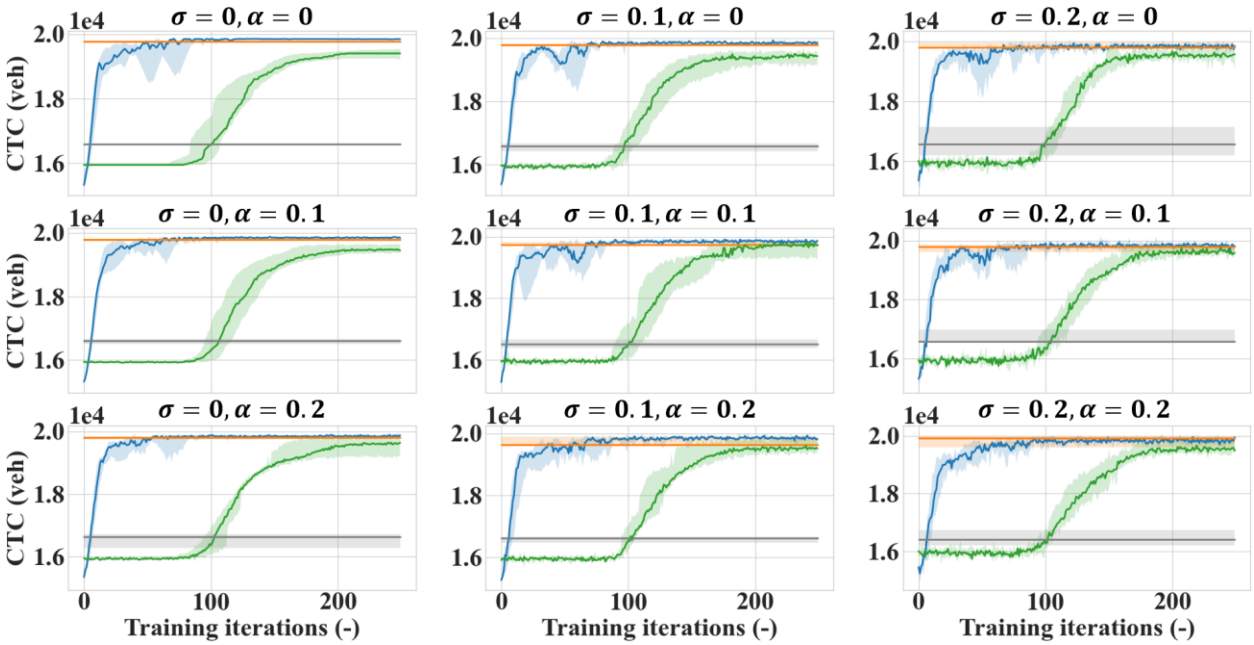


Fig. 6. Performance curves of the four methods under all environment configurations.
Blue: C-RL; Orange: MPC; Green: D-RL; Gray: NC[2]

As shown in Fig. 6, performances of the MFDRLPC scheme (the C-RL and D-RL agents) steadily improve as training proceeds and become stable after some time under all environment configurations. This demonstrates the ability to consistently learn and converge regardless of experiment scenarios. Both agents outperform the NC strategy well before convergence, with the C-RL agent taking around 5 iterations while D-RL agent around 100 iterations. In addition, at the end of the training process, all three perimeter control methods achieve much higher CTC than the NC, indicating that the congestion can be effectively mitigated and network throughput increased using perimeter control. A further comparison between the C-RL and D-RL agents reveals the significantly superior convergence property of the former. On one hand, the C-RL agent can learn much more efficiently than the D-RL agent. Specifically, it takes the C-RL agent around 3 iterations to exceed the D-RL agent though it underperforms the latter initially. Additionally, the C-RL agent achieves its best performance within 100 iterations for all environment configurations. In contrast, the D-RL agent is just about to conduct effective learning after 100 iteration. The C-RL agent also realizes much better asymptotic performances than the D-RL agent. For one, performance curves of the C-RL agent exhibit variations of CTC that are far smaller than the D-RL agent towards the end of the training process. For another, the final achieved CTC of the C-RL agent are always better than the D-RL agent. These observations demonstrate the advantage of

[2] The figures here adopt colors to differentiate between control methods and the reader is encouraged to refer to the electronic version of the figures when possible.

continuous control scheme over discretized control strategy in high-dimensional problems, as expected.

More importantly, the performance of the C-RL agent is comparable with and may even exceed the MPC method for all scenarios considered. In cases where there is relatively large uncertainty in the environment (Env. No. 3, 5-9), the performance of the D-RL agent also approaches the MPC, though consistently inferior. These results showcase the considerable potential of Deep-RL on MFD-based network traffic control. However, they are not surprising. In fact, it has been shown in the literature that model free methods have the potential to achieve comparable or even better performances to the MPC; see (Lei et al., 2019; Ren et al., 2020) for example. The reasons are twofold. For one, the MPC relies heavily on an accurate prediction model. When there is discrepancy in the traffic dynamics between the prediction model and plant, the performance of the MPC might deteriorate as it receives wrong information from the environment. On the contrary, the proposed agents internalize such dynamics via interaction with the environment. Consequently, the agents become roughly as knowledgeable about the environment as the MPC. For another, the MPC is sensitive to horizon parameters (Prabhu and George, 2014; Schrangl et al., 2018). While the MPC considers only part of future conditions (i.e., the prediction horizon), the proposed agents consider the entire simulation period with discount. In addition, careful tuning of the horizon parameters is nontrivial yet time consuming, and it is uncertain whether a particular parameter setting works consistently well across different environments. These two reasons also justify the development of model free methods for perimeter control.

4.2.2 Effectiveness of the MFDRLPC scheme

The effectiveness of the proposed scheme is examined by visualizing its control outcomes and comparing those with the MPC approach. The deterministic environment (i.e., Env. No. 1) is considered here for illustration. Similar considerations can be conducted for other environments, but the conclusions are the same. The evolution of accumulations and controller values for u_{12} in the 1-hour period are provided in Fig. 7 and Fig. 8, respectively. Note that the demand profile adopted in this paper exhibits a smaller overall demand to R_1, which has a larger production function (i.e., a larger MFD). Transfer flows to R_1 are thus not constrained by any control strategy during the whole study period, i.e., $u_{21} = u_{\max}$. This is excluded from the control actions in Fig. 8 for clarity of presentation.

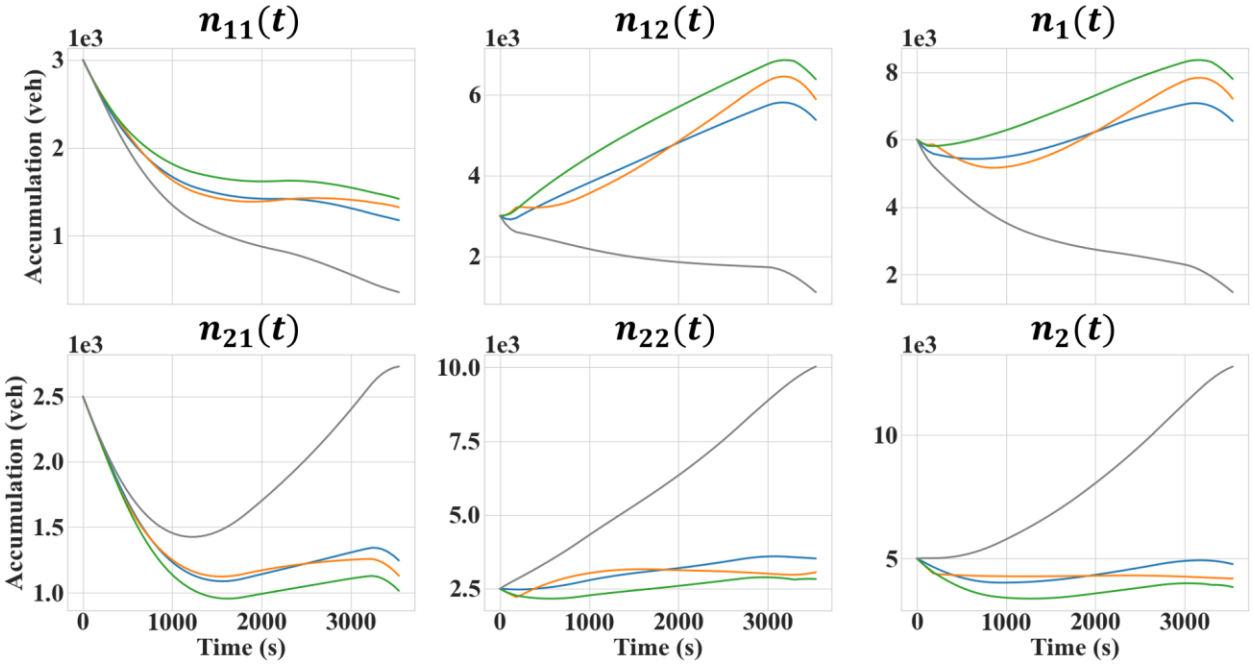


Fig. 7. Evolution of accumulations in the deterministic environment.
Blue: C-RL; Orange: MPC; Green: D-RL; Gray: NC

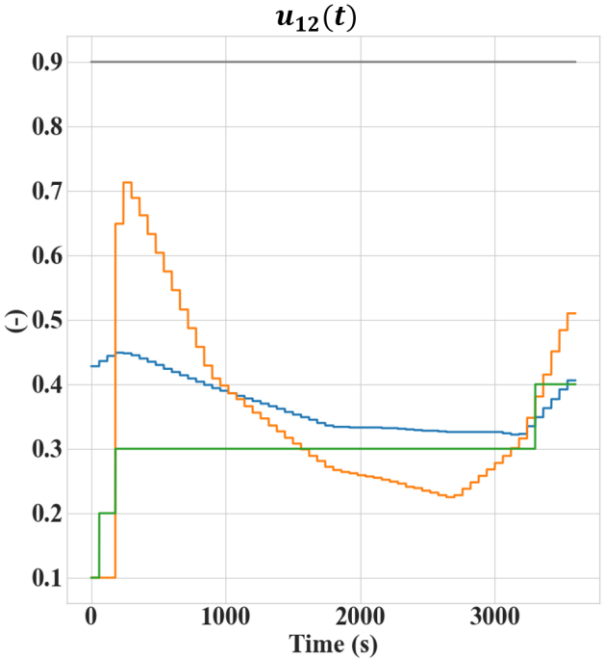


Fig. 8. Control action (u_{12}) over time in the deterministic environment.
Blue: C-RL; Orange: MPC; Green: D-RL; Gray: NC

The evolution of accumulations for the NC strategy shown in Fig. 7 corresponds to that in Fig. 4(b). The total accumulation in R_2 keeps increasing as a consequence of the high demand to R_2. Note that n_{21} first decreases as initially the demand q_{21} is rather small. Later, as q_{21} increases and R_2 becomes sufficiently congested, the accumulation n_{21} increases even though no limitation is imposed on the transfer flow $M_{21}(t)$. Contrarily, accumulations in R_1 steadily decrease since

internal flows are not restricted while the inter-region traffic demand is offset by the transfer flow. When perimeter control is applied, a fraction of transfer vehicles to R_2 are blocked and instead remain within R_1, resulting in an initial increase of n_{12}. Approaching the end of the 1-hour period, n_{12} decreases following the reduction of transfer demand q_{12} and relaxed controller u_{12}. Owing to perimeter control, accumulations in R_1 end up higher than realized by the NC strategy. However, R_1 is still uncongested and a higher regional production can be achieved. Unlike the NC method that allows n_2 to increase in an unconstrained manner, perimeter control strategies effectively maintain n_2 around the critical value such that throughput in R_2 is improved as well. Overall, the progression of accumulations depicted in Fig. 7 clearly illustrates how the unbalanced traffic condition by the NC approach can be productively mitigated by perimeter control and how network throughput can be increased.

A closer examination of Fig. 7 reveals that the accumulations realized by the proposed C-RL and D-RL agents are similar to those by the MPC, mostly due to the resemblance of control actions between them, as shown in Fig. 8. At the beginning of the study period, all three perimeter control methods tend to allow more transfer flows by increasing u_{12}, with the MPC having the sharpest increase. Around the same time when the MPC begins to decrease u_{12} in response to the high transfer demand q_{12}, the C-RL agent also starts to impose stricter controls, whereas the D-RL agent keeps the control inputs static. For the D-RL agent that takes discrete actions with fixed jumps, this is a sensible decision since adjusting the control downwards or upwards might result in drastic changes in the accumulations and hurt the trip completion. However, a better decision could be readily obtained. For example, the D-RL agent could increase u_{12} by 0.1 and maintain the value until decreasing it around 1000s. The D-RL agent fails to learn such strategy, which explains why its realized trip completion is lower than the C-RL agent or the MPC. Later, as overall demand to R_2 reaches its maximum, the MPC enforces even stricter control. When internal demand q_{22} decreases and traffic conditions in the two regions become slightly unbalanced, the MPC lessens its restriction on the transfer flow. During this period, however, both proposed agents apply a roughly constant control value for u_{12}. Finally, as all traffic demands diminish at the end of the 1-hour period and no severe congestion is present in the system, all three perimeter control methods are inclined to increase u_{12} so that more transfers can be made and more trips completed. Note that, the C-RL agent generally selects smaller controller values for u_{12} than the MPC in the early period (before 1000s). However, such smaller values are compensated by its looser control in the later period (around 1000~3000s), which explains its competitiveness to the MPC method. In addition, the control policy learned by the C-RL agent is more implementable than the MPC since its actions change more smoothly. Such smoothness is built in the design of the actor network, which uses the tanh activation for action outputs. Given that perimeter control is typically implemented by traffic signals in practice, a smooth control policy could thus avoid the need to adjust the signal timing plans abruptly across consecutive time steps. In summary, these results indicate that the proposed scheme, especially the C-RL agent, is capable of deriving similar control policies to the MPC while in the meantime preserving the easiness of practical implementations.

To more comprehensively demonstrate the effectiveness of the proposed scheme, three more tests are conducted: (1) the environment exhibits atypical traffic demand patterns; (2) the network becomes empty at the end of an extended control period; and (3) the agent only has limited access to accumulations in the environment. Specifications of these tests will be provided shortly. Note that, since the C-RL agent has been shown much superior to the D-RL agent, the latter is not considered in the following experiments.

4.2.2.1 Atypical traffic demand profiles in the environment

For all experiments presented above, the environment assumes approximately realistic traffic demand profiles that simulate traffic conditions during morning peaks. The objective of this test is thus to investigate whether the proposed C-RL agent could still learn reasonable control strategies with atypical demand profiles in the environment. The deterministic environment is considered in this test for illustration.

The demand profile in Fig. 9(a) is scaled down from that in Fig. 4(a) with a coefficient of 0.3, which simulates extremely low traffic demands in the environment. In Fig. 9(b), internal demand in R_2 is increased drastically while the demands to R_1 are kept unchanged to simulate a busy city center. Intuitively, when traffic demands are extremely low in the environment (first demand profile), congestion should rarely exist and little to no perimeter control should be implemented, i.e., $u_{12} \approx 0.9, u_{21} \approx 0.9$. On the contrary, when one region is always congested due to high internal demand and the other barely congested (second demand profile), perimeter controllers should be activated to allow as many outbound vehicles as possible and as few inbound vehicles as possible for the congested region, i.e., $u_{12} \approx 0.1, u_{21} \approx 0.9$. This intuition is confirmed by solutions to the control problem using the MPC; see Fig. 10.

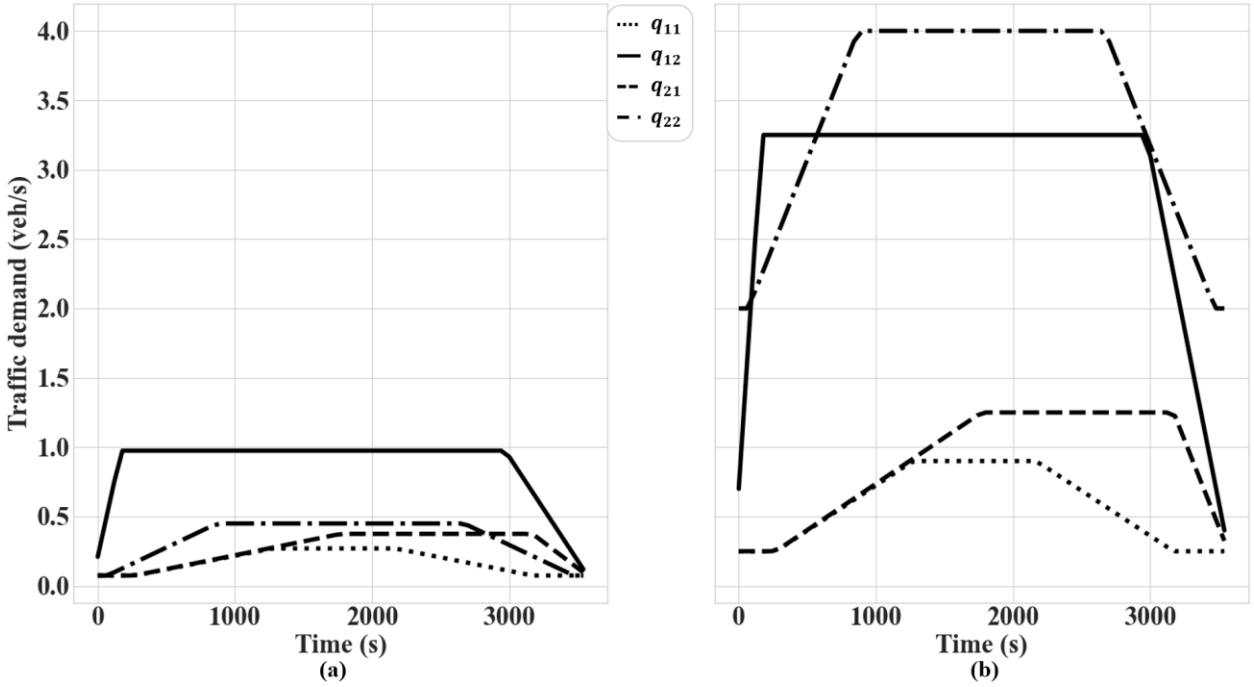


Fig. 9. Atypical demand profiles: (a) low demands; (b) high internal demand q_{22}

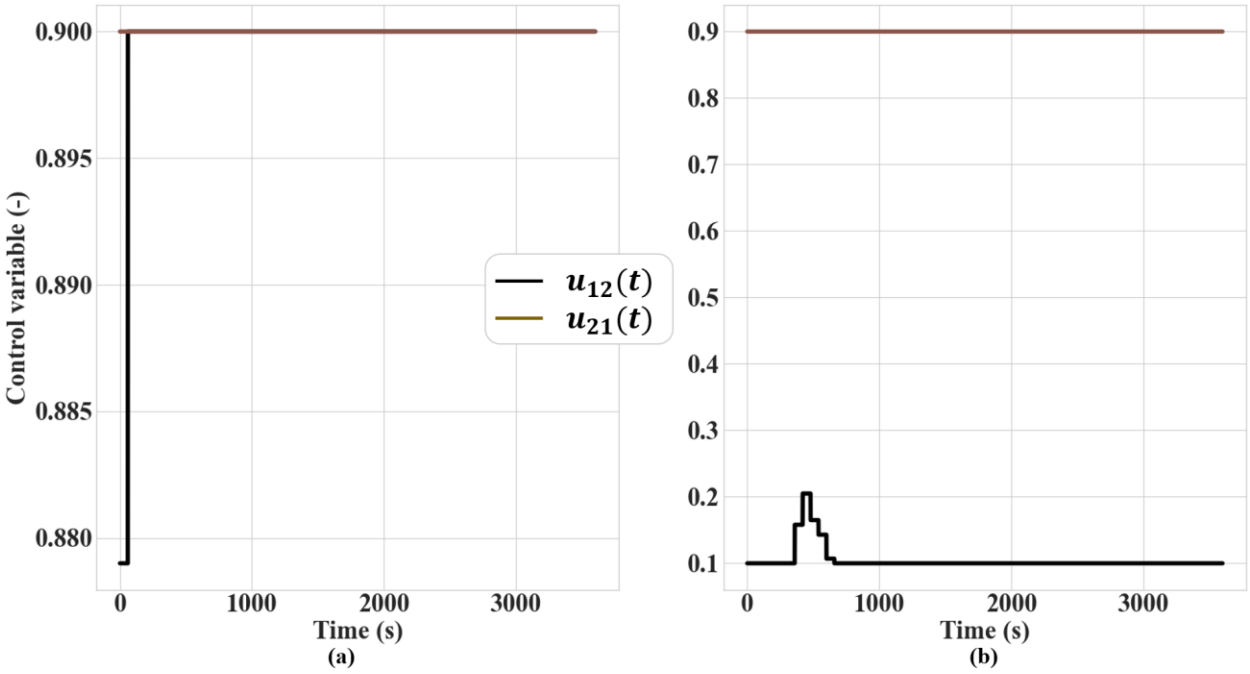


Fig. 10. MPC control variables for: (a) low demands; (b) high internal demand q_{22}

Similarly, control actions chosen by the C-RL agent are presented in Fig. 11, which are also consistent with the above intuition. The control actions by the MPC and the C-RL agent are not exactly identical, thus resulting in slightly different performance curves, as reflected in Fig. 12. As can be observed from Fig. 12, the C-RL agent can learn the optimal policy fairly quickly (within 50 iteration) and achieve comparable performances with the MPC. Note that since the policies for these extreme scenarios are adequately simple to learn, only one performance curve is presented for each case, but the C-RL agent was tested using numerous random seeds.

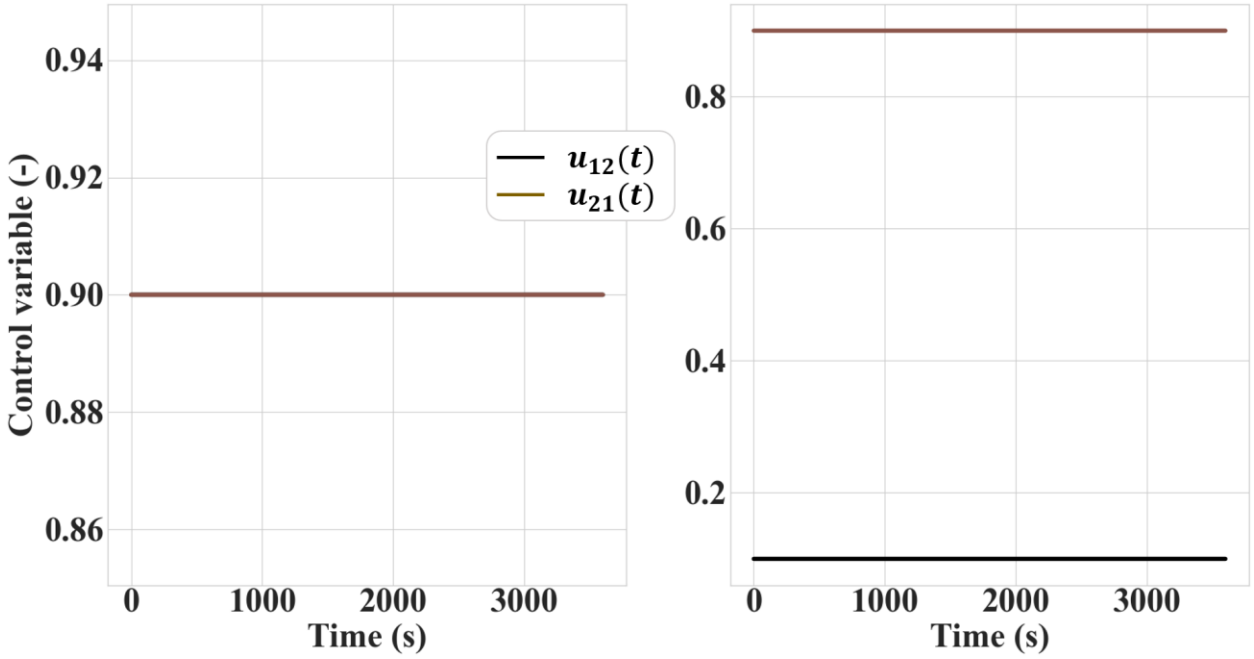


Fig. 11. C-RL control variables for: (a) low demands; (b) high internal demand q_{22}

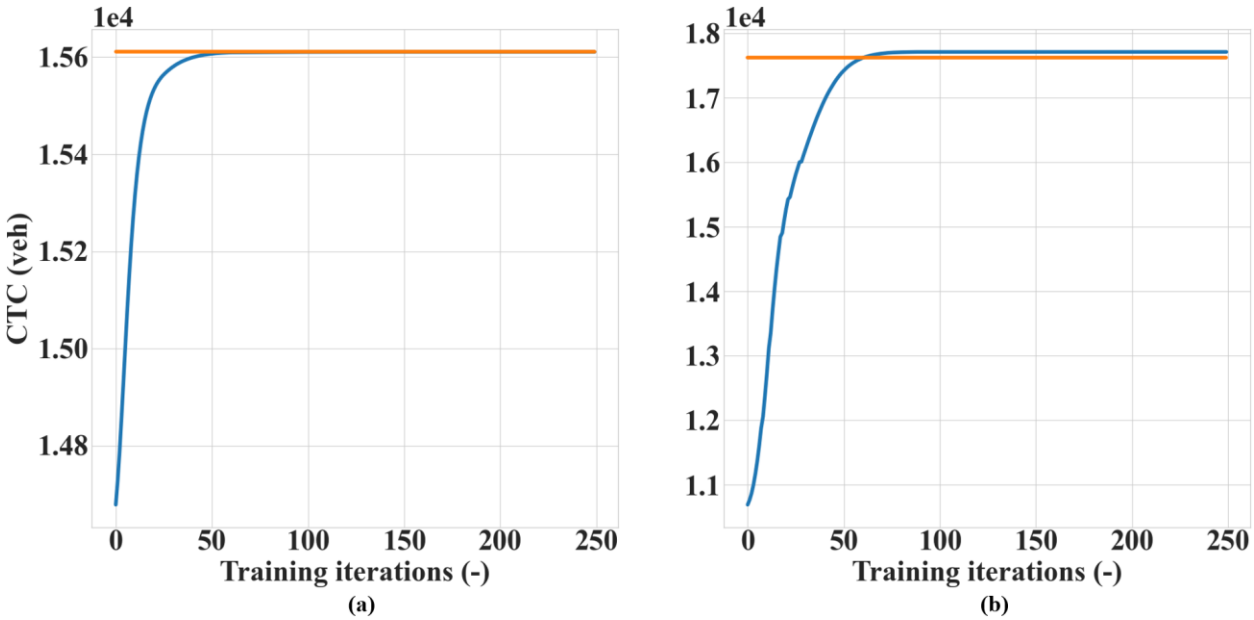
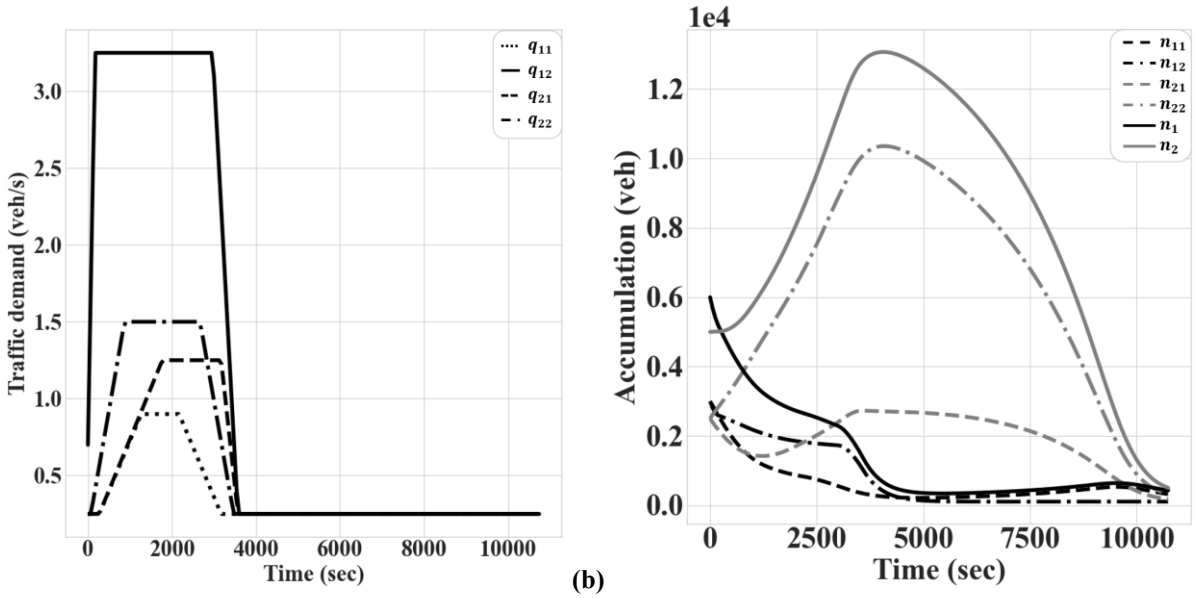


Fig. 12. Performance curves for: (a) low demands; (b) high internal demand q_{22}
Blue: C-RL; Orange: MPC

4.2.2.2 Extended control period with network being empty in the end

The demand profile in Fig. 4(a) simulates a morning peak. At the end of the simulated morning peak, the two regions both operate around their respective critical value under perimeter control; see Fig. 7. However, it remains uncertain whether subsequent operations of the two regions would alter the conclusions presented above. Hence, this test considers an experiment scenario where the

network becomes nearly empty at the end of the control period. To do so, the control period is extended to 3 hours with a demand profile shown in Fig. 13(a). The extended portion from Fig. 4(a) represents the traffic situations after the morning peak with a constant small value for all four demands. Again, the deterministic environment is considered here as an illustration. The evolution of accumulations under the NC strategy is presented in Fig. 13(b). As can be observed, the network becomes close to empty at the end of the simulation even when no perimeter control is applied. Therefore, a reasonable comparison could be established between the C-RL agent and the MPC as the accumulation trajectories beyond the control period will not affect the comparison.



(a) **(b)**

Fig. 13. (a) 3-hour demand profile; (b) accumulations under the NC strategy

To compare with the MPC, a combined control strategy is presented. Specifically, during the first hour of the simulation, the C-RL agent is utilized for perimeter control. For the subsequent control period, the NC strategy is adopted. Reasons for choosing such a combination of control methods are twofold. For one, as demonstrated in the test above with atypical demand profiles, no control should be implemented with extremely low traffic demands in the environment. Though nontrivial accumulations are present in the network, they are near the critical values and do not cause congestion. In practice, it is natural to lift any restriction on transfer flows after the morning peak. For another, when the traffic demands are close to zero, four input signals to the C-RL agent will be close to zero, which hurts its training process.

The evolution of trip completion over time is provided in Fig. 14 for the NC, the MPC, and the proposed control method (i.e., C-RL + NC). Note that, all control methods should have the same trip completion when the network becomes empty, which is consistent with Fig. 14. As can be observed, the performance of the proposed control strategy is almost identical to the MPC, which is indicative of an extremely high level of comparability between the two methods. In addition, the proposed method and the MPC realize trip completions much more efficiently than the NC, resulting in a substantial area between the completion curves. This area represents the saved total travel time (TTT) from the NC strategy, a term to be revisited shortly. This observation again indicates the notable advantage of perimeter control over the NC method.

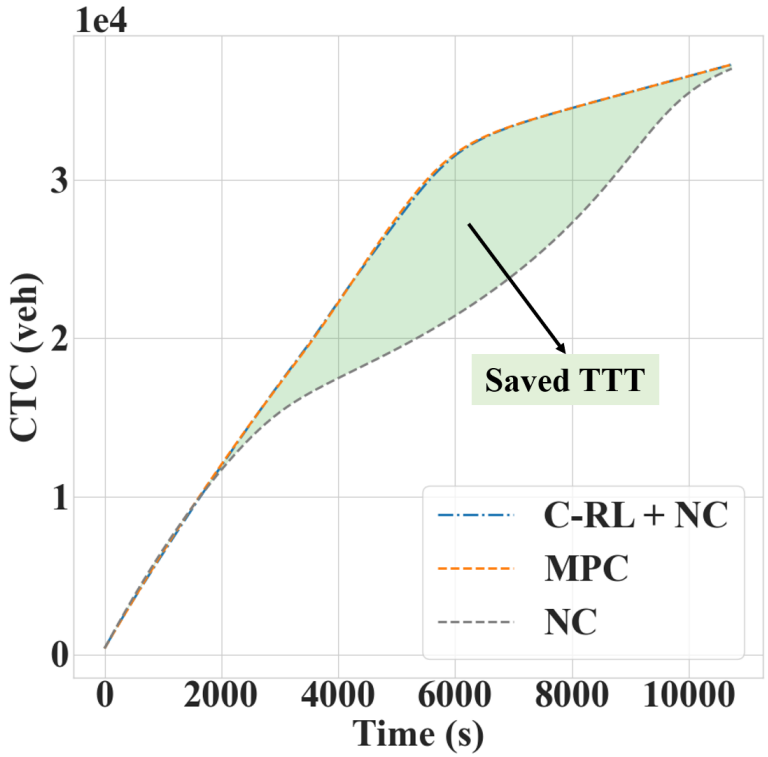


Fig. 14. Evolution of trip completion over time

4.2.2.3 Limited access to environment accumulations for the agent

As described in *Section 3.1*, the proposed C-RL agent receives accumulations and demands from the environment as its state and acts based on this information. While estimates of traffic demands can be readily obtained from historical observations, the acquisition of accumulations n_{ij} might not be straightforward, though they can be estimated using loop detectors and probe vehicles. Therefore, this test investigates whether the proposed agent works in the absence of such detailed accumulation when it only has access to total regional accumulations $n_i, i = 1,2$. The evolution of trip completion in the deterministic environment is presented in Fig. 15, where the original C-RL agent trained with accumulations n_{ij} is denoted by *C-RL (ij)* while the one trained with n_i by *C-RL (i)*. As can be observed, training the C-RL agent with total regional accumulations n_i is almost identical to training with detailed accumulations n_{ij}, which indicates that the proposed agent works with only regional accumulations n_i. This result also manifests the implementation potential of the proposed agent since total regional accumulations are comparatively easy to obtain.

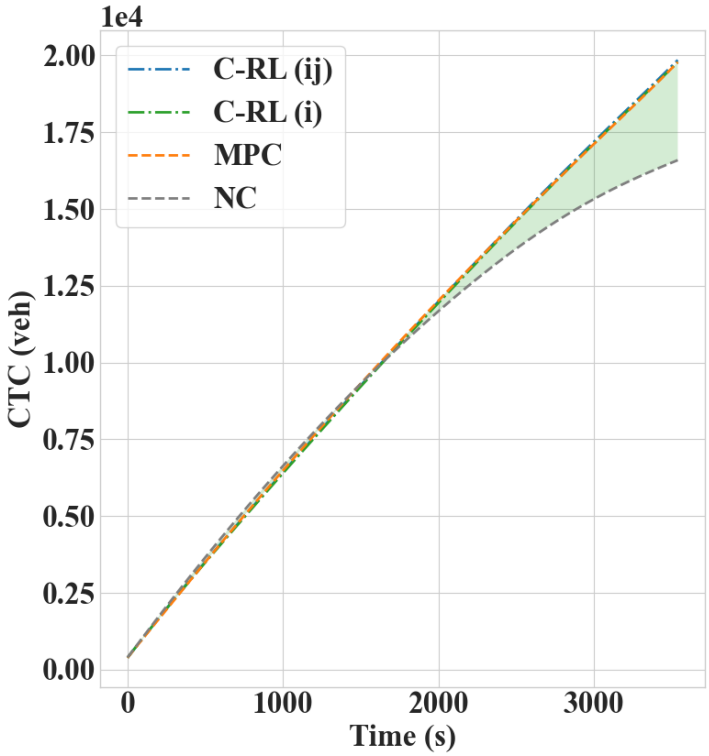


Fig. 15. Evolution of trip completion over time for the NC, the MPC, the C-RL agent trained with n_i, and the one trained with n_{ij}.

To sum up, the results presented in this section suggest that the proposed method can effectively learn sensible perimeter control strategies to maximize the number of trips completed. For realistic experiment scenarios, the proposed scheme (the C-RL agent in particular) obtains similar but more implementable control policies to the MPC. The proposed scheme could also learn the optimal control policy with relative ease for scenarios with irregular traffic demands. Additionally, the proposed scheme is highly comparable to the MPC when the network becomes nearly empty towards the end of the control period. Moreover, the proposed scheme works even without detailed accumulations n_{ij} from the environment. These results have fully demonstrated the effectiveness of the proposed scheme and imply that it might be applicable on all types of traffic conditions in the environment. Finally, these results also show the promising application prospect of Deep-RL methods on MFD-based traffic control.

4.2.3 Transferability of the MFDRLPC scheme

The transferability of the proposed scheme is examined in this section by applying the pretrained C-RL agents on a variety of unseen environments. To the best of our knowledge, this is the first examination of whether perimeter control methods can generalize to unencountered environments. For all case studies reported in (Ren et al., 2020), it is uninvestigated whether the learned controller for one case could transfer to another without performing the learning process all over again. The model based MPC formulates an optimization problem at each time step for each environment and it remains in question whether a particular parameter setting could work in another environment. Note that the term “transferability” is employed here instead of “robustness” as the latter has a

broader meaning which does not fit in well with the context. Moreover, since the proposed C-RL agent has clear advantages over the D-RL agent, the latter is excluded from following experiments.

The performances of the C-RL agent are compared with the MPC and NC strategies in terms of total travel time (TTT) in the system. The new environments considered here have different traffic conditions (i.e., initial accumulations and traffic demands) and traffic dynamics (i.e., MFDs) from those used to train the agents. The initial accumulations and traffic demands in the unseen environments are obtained by:

$$n_{i,new} = n_{i,0} \cdot (1 + \phi) \tag{21}$$

$$\tilde{q}_{ij}(t) = q_{ij}(t) \cdot (1 + \eta) \tag{22}$$

where ϕ, η denote the level of variation. Values of ϕ, η considered here range from -0.10 to 0.10. Thus, higher values indicate higher initial accumulations or traffic demands in the environment than originally seen by the C-RL agent. Note that, the MPC has full knowledge of the new traffic conditions. The MFDs in the unseen environment are defined as:

$$\tilde{f}_i\big(\tilde{n}_i(t)\big) = f_i\big(\tilde{n}_i(t)\big) + \varphi \cdot \tilde{n}_i(t) \tag{23}$$

where φ stands for the MFD modeling errors. Negative values of φ are indicative of over-representation of environment dynamics and the new environment is not as productive as perceived by the C-RL agent or the MPC. On the contrary, positive values of φ suggest underestimation of environment production. Values of φ considered here range from -0.50 to 0.50. The experiments in this section are conducted on a large number of different environments, and only a subset of the results is presented here for brevity.

The first set of experiments test the transferability of the proposed method with respect to different initial accumulations in the environment. Thus, the traffic demands and MFDs are kept constant. Intuitively, higher initial accumulations will aggravate the unbalanced traffic condition under the NC strategy, making perimeter control increasingly necessary. Contrarily, perimeter control might not be needed if the initial accumulations are low enough such that congestion rarely exists. The achieved TTT of three control methods with respect to the initial accumulations is presented in Fig. 16, where each subplot represents a C-RL agent trained in the corresponding environment (see Table 2). Heights of the green shaded areas represent saved TTT by the C-RL agent from the NC strategy. Similarly, heights of the pink shaded areas represent the saved TTT by the MPC from the C-RL agent. The new environments considered here assume $\eta = 0, \varphi = 0$ for illustration. Note that, performances of C-RL agent trained on individual environments are less informative due to randomness, hence more analytical focus is placed on the general trend.

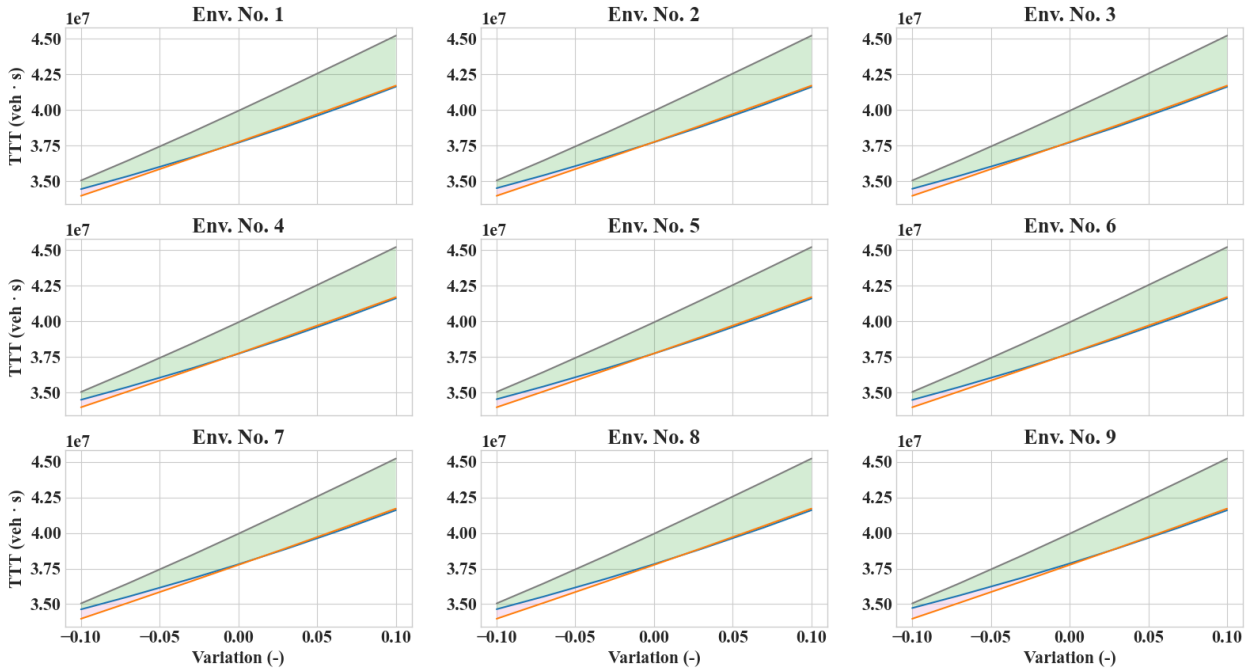


Fig. 16. TTT achieved by three methods with respect to initial accumulations.
Blue: C-RL; Orange: MPC; Gray: NC

As can be observed from Fig. 16, the saved TTT from the NC increase monotonically with initial accumulations, which suggests a higher level of necessity to implement perimeter control as the network becomes more congested initially. It can also be inferred that, if the network starts with extremely small initial accumulations ($\phi = -0.3$, for example), the NC might be the optimal control strategy. More importantly, Fig. 16 shows that the realized TTT of the C-RL agent is about the same as the MPC when initial accumulations are moderately larger than those perceived by the agents during training (i.e., $0 \leq \phi \leq 0.10$). This suggests the C-RL agent is highly transferable to environments with larger initial accumulations, even more so considering that the MPC has full access of the new initial accumulations and system dynamics whereas the C-RL agent has never encountered these test environments. When the initial accumulations are smaller than perceived by the C-RL agent, it fails to compete with the MPC, which might result from its strict control policy as more relaxed controls are favorable in an environment that is less congested. Additionally, Fig. 16 reveals that the C-RL agent performs approximately similarly when transferring to unseen environments regardless of the environment it was trained on. This might suggest the C-RL agent has decent generic property, i.e., it can cope with any traffic condition in the environment.

The second set of experiments test the transferability of the proposed method to different demands patterns, so initial accumulations and the MFDs are held constant. The performance of the C-RL agent with respect to traffic demands is presented in Fig. 17, where all new environments assume $\phi = 0, \varphi = 0$. As can be observed, the pretrained C-RL agent is remarkably comparable to the MPC when applied to environments with different levels of traffic demands, irrespective of its training environments. In particular, the C-RL agent appears notably transferable to large demand patterns. On the other hand, as traffic demands in the environment become much smaller than those known to the agent, the MPC is more advantageous, which is also likely the result of the stricter controls of the C-RL agent. Overall these two sets of tests suggest that the proposed method

is resilient to slightly more congested traffic conditions but might be insufficient if the test environment is much less congested than perceived by the pretrained agents.

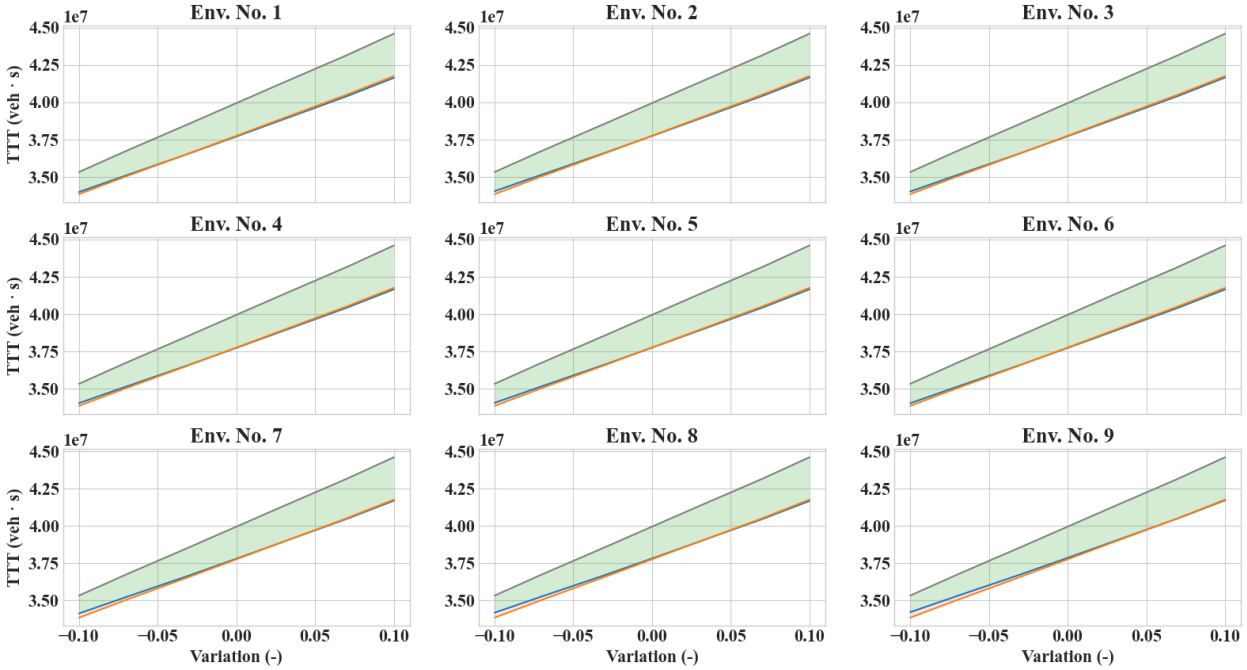


Fig. 17. TTT achieved by three methods with respect to traffic demands.
Blue: C-RL; Orange: MPC; Gray: NC

The transferability of the proposed scheme with respect to the MFD modeling errors is investigated in this final set of experiments. Intuitively, when large distinctions of traffic dynamics exist between the training environments and the new environments, the pretrained agents may cease to achieve consistently satisfactory performances. Assume for example that the proposed method is trained to control a relatively small urban network; the learnt policy should not be expected to perform well on a much larger urban network as the dynamics of the small network would likely not be representative of the larger one.

The performance of the C-RL agent with respect to MFD modeling errors is shown in Fig. 18, where the new environments assume $\phi = 0, \eta = 0$. Initially when the environment dynamics are severely over-represented ($\varphi \leq -0.4$), perimeter control methods significantly outperform the NC. This is reasonable as when the environment is not as productive as perceived, more serious congestion is likely to arise under the NC strategy. As the environment dynamics becomes less overrated or even slightly underestimated ($-0.3 \leq \varphi \leq 0.1$), the advantage of perimeter control diminishes, but the C-RL agent is still comparable to the MPC. However, when the environment production is highly underestimated ($\varphi \geq 0.3$), both perimeter control methods fail to compete with the NC, suggesting that vehicle movements should not be restricted. This again is sensible since congestion might not exist with higher production in the environment and the NC can allow more vehicles to complete their trips, thus reducing the TTT in the system.

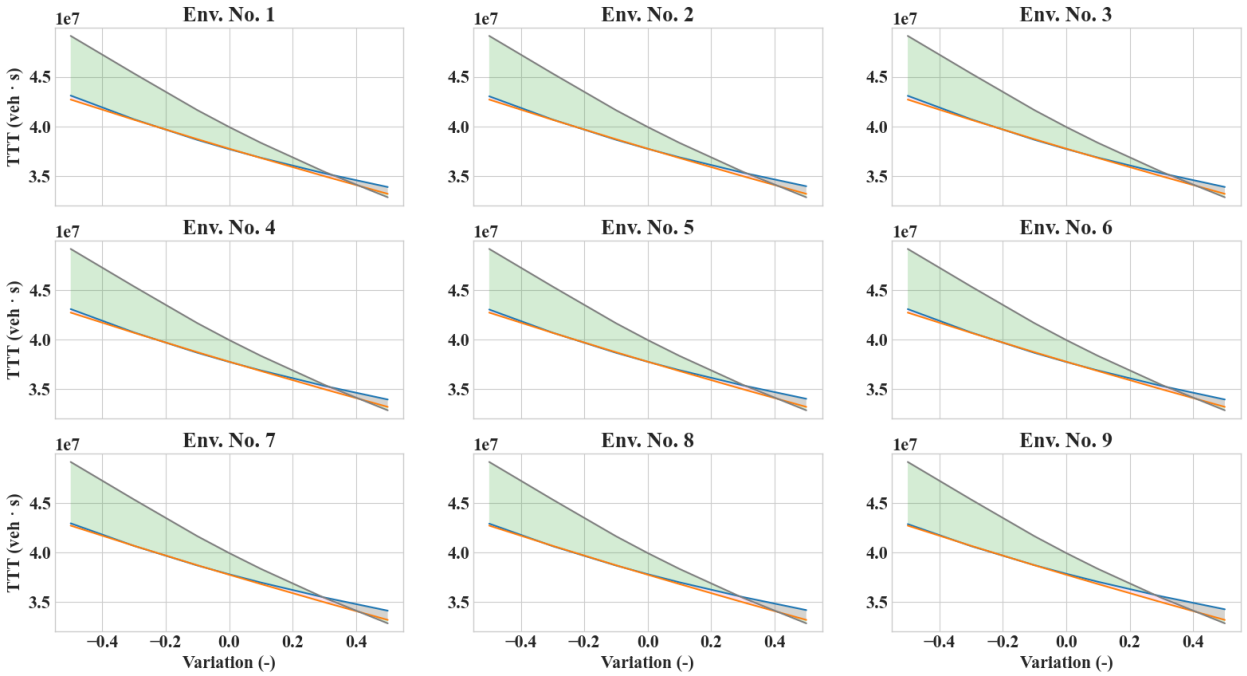


Fig. 18. TTT achieved by three methods with respect to MFD modeling errors.
Blue: C-RL; Orange: MPC; Gray: NC

To conclude, the experiment results presented in this section have demonstrated that the proposed C-RL agent is comparable to the MPC when applied to test environments with slightly more congested traffic conditions even though the MPC has full knowledge of the traffic dynamics in the test environments. In addition, the C-RL agent has been shown transferable to environments with moderate MFD modeling errors. Results also suggest the proposed scheme and the MPC may not adapt well to environments with considerably underestimated traffic dynamics. However, this should not be an issue in practice. The reasons are twofold. First, one could train a C-RL agent for each network to be controlled. This pretrained agent will then be resilient to changeable traffic conditions as shown above. Second, traffic dynamics in a network are roughly reproducible (Geroliminis and Daganzo, 2008) with the upper bounds largely confined by the network topology. Therefore, the pretrained agent is unlikely to severely underestimate the environment production and thus could consistently work for the network. Additionally, as with the convention of applying pretrained RL agents, the test scenario should not be greatly different from the training ones.

5. DISCUSSION AND CONCLUDING REMARKS

In summary, this paper proposes a novel Deep-RL based scheme (i.e., the MFDRLPC) for the canonical two-region perimeter control problems. The proposed scheme features two agents that respectively assume continuous and discrete action spaces. The distributed Ape-X architecture is combined with the learning algorithms of the agents to facilitate efficient learning. Extensive numerical experiments have been conducted to assess the convergence consistency of the proposed approach and to compare its performance against the state-of-the-art MPC method. Results show that the proposed scheme could consistently converge under a wide range of environments even with significant uncertainty in realized traffic demands and network MFDs. Moreover, the

proposed agents are highly comparable to the MPC method when applied to the environments that they were specifically trained in, even with atypical demand profiles in the environment, extended control period, or limited access to accumulation information. In addition, the proposed scheme demonstrates a high level of transferability in application to new unseen environments, which indicates great potential for practical implementations. Concretely, the agents generally achieve comparable performances to the MPC approach when applied to environments with slightly more congested traffic conditions and moderate MFD modeling errors.

The proposed MFDRLPC scheme has some clear advantages over existing model based or model free perimeter control methods (Geroliminis et al., 2013; Lei et al., 2019; Ren et al., 2020, 2018; Sirmatel and Geroliminis, 2018; Su et al., 2020). The principal benefit is that the proposed scheme does not build on any information about environment dynamics, thus avoiding the need to estimate the MFD or critical accumulation. In addition, the proposed scheme does not require pre-collected external traffic data but instead learns control strategies entirely from direct interaction with the environment. Though the learning process could potentially benefit from additional network-wide data (such as the critical accumulation, maximum trip completion rate or even the entire region's MFD), the results show that the latter are not needed since the proposed scheme learns from the overall network outcomes. In this manner, the proposed scheme can internalize the traffic dynamics and achieve comparable performances to the state-of-the-art MPC approach. There are, however, potential obstacles that may hinder its real-life application.

First, the proposed scheme could be data intensive and computationally expensive, particularly in the learning portion. In this paper, it takes around 43 to 52 minutes to train the C-RL and D-RL agents respectively whereas the computation time of the MPC is negligible in comparison. However, such straightforward comparison of computational time does not fairly assess the two methods. Model free Deep-RL methods require numerous samples to produce a reasonable control policy as the samples are utilized to learn the environment dynamics. On the contrary, the MPC requires that such dynamics be known and incorporated into the framework. Therefore, though the MPC appears computationally cheap, it is considerably data intensive as it depends on detailed traffic dynamics and the MFD, which are generally difficult to obtain. Thus, by nature of the model free design, the proposed scheme will take longer computation time to train than the MPC. However, while it does take more time to train the agents from scratch, the time required to implement the pretrained Deep-RL agents is negligible. In fact, the proposed scheme is much less computationally intensive in the application process than the MPC as the latter always formulates and solves a high-dimensional nonlinear program every time it is implemented. In addition to this benefit, the proposed scheme exhibits decent potential for real-life application. For one, the proposed scheme could be trained offline in simulation and then applied to a more realistic environment. As the transferability tests indicate, the proposed scheme could generalize to environments with slightly more congested traffic conditions and moderate MFD modeling errors. Hence, the agent could first be trained with slightly uncongested traffic condition and high environment production such that it can achieve promising performances at the time of adaptation. For another, the offline pretrained agent could keep on training at the time of application, thus improving its control outcomes. This process could be conducted in real time with newly collected experiences as the online adaptation is neither data nor computation intensive. In summary, the proposed scheme shows promising implementation prospect since it could be trained offline and applied (and trained) online with performance guarantees. Moreover, from a practical standpoint,

the proposed scheme does not need to be trained from scratch in real world as this may cause some underperformance in the early stage of training.

Additionally, the proposed scheme may encounter hindrance from existing infrastructure. For example, the state designs of the proposed agents include detailed accumulations n_{ij}, whose acquisition may be nontrivial. Fortunately, this can be accommodated with ease in practice. For one, n_{ij} can be estimated via large scale use of loop detectors and probe vehicles. The former can provide reasonable estimates of regional accumulations (i.e., n_i), while the latter can provide regional destination splits. This information can then be used to partition the regional accumulations into destination-specific ones (i.e., convert n_i into n_{ij}). The accuracy of these methods is a subject of open research, which is left as future work of this study. For another, as *Section 4.2.2.3* shows, the proposed scheme could perform well even with only total regional accumulations n_i, which are easily attainable from the environment. Similarly, the reward design features the use of exit flows (i.e., trip completion rates), which might be difficult to obtain directly. However, exit flows have been shown to be linearly related to network average flows (Daganzo, 2007; Geroliminis and Daganzo, 2008); thus, the latter could readily be adopted as surrogates for the reward.

Finally, the environment is assumed to be fully observable, as widely done in the literature. Hence, data transmission is not restricted, and loss of information (such as states and rewards) is not considered. While this might be a moderately restricting assumption, it should not be an impediment to related research works. Moreover, this assumption will cease to be restricting with advances in the technological frontier, such as high penetration of connected automated vehicles.

Considering the potential limitations above, possible future efforts are pointed out here that might enhance real-life applications of the proposed scheme. This includes training the agents in a more realistic environment (as opposed to the numerical simulations performed herein) and perhaps using field experience. To this end, one should consider collecting interactive traffic data in real-time to calibrate the simulation environment and train the proposed agents continuously during application, which would be necessary for real-world scenarios. In this manner, more information about the traffic dynamics might be internalized by the proposed agents such that the simulation environment can do away with explicit modeling using MFDs. Future work could also consider developing a general perimeter controller that learns how to control for a new network without retraining or how to improve convergence rate of the proposed agents (for example via more principled exploration or experience utilization).

ACKNOWLEDGEMENTS

This research was supported by NSF Grant CMMI-1749200 and a seed grant through the Penn State Institute of CyberScience.

REFERENCES

Aboudolas, K., Geroliminis, N., 2013. Perimeter and boundary flow control in multi-reservoir heterogeneous networks. Transp. Res. Part B Methodol. 55, 265–281. https://doi.org/10.1016/j.trb.2013.07.003

Ambühl, L., Menendez, M., 2016. Data fusion algorithm for macroscopic fundamental diagram estimation. Transp. Res. Part C Emerg. Technol. 71, 184–197. https://doi.org/10.1016/J.TRC.2016.07.013

Bhatnagar, S., Sutton, R.S., Ghavamzadeh, M., Lee, M., 2009. Natural actor-critic algorithms. Automatica 45, 2471–2482. https://doi.org/10.1016/j.automatica.2009.07.008

Buisson, C., Ladier, C., 2009. Exploring the Impact of Homogeneity of Traffic Measurements on the Existence of Macroscopic Fundamental Diagrams. Transp. Res. Rec. J. Transp. Res. Board 2124, 127–136. https://doi.org/10.3141/2124-12

Daganzo, C.F., 2007. Urban gridlock: Macroscopic modeling and mitigation approaches. Transp. Res. Part B Methodol. 41, 49–62. https://doi.org/10.1016/j.trb.2006.03.001

Daganzo, C.F., Gayah, V. V., Gonzales, E.J., 2011. Macroscopic relations of urban traffic variables: Bifurcations, multivaluedness and instability. Transp. Res. Part B Methodol. 45, 278–288. https://doi.org/10.1016/j.trb.2010.06.006

Daganzo, C.F., Lehe, L.J., 2015. Distance-dependent congestion pricing for downtown zones. Transp. Res. Part B Methodol. 75, 89–99. https://doi.org/10.1016/j.trb.2015.02.010

Degris, T., White, M., Sutton, R.S., 2012. Off-Policy Actor-Critic. Proc. 6th Int. Symp. Adapt. Motion Anim. Mach.

Du, J., Rakha, H., Gayah, V. V, 2016. Deriving macroscopic fundamental diagrams from probe data: Issues and proposed solutions. Transp. Res. Part C Emerg. Technol. 66, 136–149.

Gao, X. (Shirley), Gayah, V. V., 2018. An analytical framework to model uncertainty in urban network dynamics using Macroscopic Fundamental Diagrams. Transp. Res. Part B Methodol. 117, 660–675. https://doi.org/10.1016/j.trb.2017.08.015

Gayah, V. V., Daganzo, C.F., 2011. Clockwise hysteresis loops in the Macroscopic Fundamental Diagram: An effect of network instability. Transp. Res. Part B Methodol. 45, 643–655. https://doi.org/10.1016/j.trb.2010.11.006

Gayah, V. V., Dixit, V. V., 2013. Using Mobile Probe Data and the Macroscopic Fundamental Diagram to Estimate Network Densities. Transp. Res. Rec. J. Transp. Res. Board 2390, 76–86. https://doi.org/10.3141/2390-09

Gayah, V. V, Gao, X.S., Nagle, A.S., 2014. On the impacts of locally adaptive signal control on urban network stability and the macroscopic fundamental diagram. Transp. Res. Part B Methodol. 70, 255–268.

Genders, W., Razavi, S., 2016. Using a Deep Reinforcement Learning Agent for Traffic Signal Control.

Geroliminis, N., Daganzo, C.F., 2008. Existence of urban-scale macroscopic fundamental diagrams: Some experimental findings. Transp. Res. Part B Methodol. 42, 759–770.

Geroliminis, N., Haddad, J., Ramezani, M., 2013. Optimal perimeter control for two urban regions with macroscopic fundamental diagrams: A model predictive approach. IEEE Trans. Intell. Transp. Syst. 14, 348–359. https://doi.org/10.1109/TITS.2012.2216877

Geroliminis, N., Sun, J., 2011. Properties of a well-defined macroscopic fundamental diagram for urban traffic. Transp. Res. Part B Methodol. 45, 605–617. https://doi.org/10.1016/j.trb.2010.11.004

Godfrey, J.W., 1969. The mechanism of a road network. Traffic Eng. Control 11, 323–327.

Grondman, I., Busoniu, L., Lopes, G.A.D., Babuška, R., 2012. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. https://doi.org/10.1109/TSMCC.2012.2218595

Haddad, J., 2017a. Optimal coupled and decoupled perimeter control in one-region cities. Control

Eng. Pract. 61, 134–148. https://doi.org/10.1016/j.conengprac.2017.01.010
Haddad, J., 2017b. Optimal perimeter control synthesis for two urban regions with aggregate boundary queue dynamics. Transp. Res. Part B Methodol. 96, 1–25. https://doi.org/10.1016/j.trb.2016.10.016
Haddad, J., Geroliminis, N., 2012. On the stability of traffic perimeter control in two-region urban cities. Transp. Res. Part B Methodol. 46, 1159–1176. https://doi.org/10.1016/j.trb.2012.04.004
Haddad, J., Mirkin, B., 2017. Coordinated distributed adaptive perimeter control for large-scale urban road networks. Transp. Res. Part C Emerg. Technol. 77, 495–515. https://doi.org/10.1016/j.trc.2016.12.002
Haddad, J., Mirkin, B., 2016. Adaptive perimeter traffic control of urban road networks based on MFD model with time delays. Int. J. Robust Nonlinear Control 26, 1267–1285. https://doi.org/10.1002/rnc.3502
Haddad, J., Ramezani, M., Geroliminis, N., 2013. Cooperative traffic control of a mixed network with two urban regions and a freeway. Transp. Res. Part B Methodol. 54, 17–36. https://doi.org/10.1016/j.trb.2013.03.007
Haddad, J., Ramezani, M., Geroliminis, N., 2012. Model predictive perimeter control for urban areas with macroscopic fundamental diagrams, in: Proceedings of the American Control Conference. pp. 5757–5762. https://doi.org/10.1109/acc.2012.6314693
Haddad, J., Shraiber, A., 2014. Robust perimeter control design for an urban region. Transp. Res. Part B Methodol. 68, 315–332. https://doi.org/10.1016/j.trb.2014.06.010
Hajiahmadi, M., Haddad, J., De Schutter, B., Geroliminis, N., 2015. Optimal hybrid perimeter and switching plans control for urban traffic networks. IEEE Trans. Control Syst. Technol. 23, 464–478. https://doi.org/10.1109/TCST.2014.2330997
Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., Meger, D., 2017. Deep Reinforcement Learning that Matters. 32nd AAAI Conf. Artif. Intell. AAAI 2018 3207–3214.
Hessel, M., Modayil, J., van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., Silver, D., 2017. Rainbow: Combining Improvements in Deep Reinforcement Learning. 32nd AAAI Conf. Artif. Intell. AAAI 2018 3215–3222.
Horgan, D., Quan, J., Budden, D., Barth-Maron, G., Hessel, M., van Hasselt, H., Silver, D., 2018. Distributed Prioritized Experience Replay.
Hou, Z., Xiong, S., 2019. On Model-Free Adaptive Control and Its Stability Analysis. IEEE Trans. Automat. Contr. 64, 4555–4569. https://doi.org/10.1109/TAC.2019.2894586
Ji, Y., Daamen, W., Hoogendoorn, S., Hoogendoorn-Lanser, S., Qian, X., 2010. Investigating the Shape of the Macroscopic Fundamental Diagram Using Simulation Data. Transp. Res. Rec. J. Transp. Res. Board 2161, 40–48. https://doi.org/10.3141/2161-05
Keyvan-Ekbatani, M., Kouvelas, A., Papamichail, I., Papageorgiou, M., 2012. Exploiting the fundamental diagram of urban networks for feedback-based gating. Transp. Res. Part B Methodol. 46, 1393–1403. https://doi.org/10.1016/j.trb.2012.06.008
Keyvan-Ekbatani, M., Papageorgiou, M., Knoop, V.L., 2015a. Controller design for gating traffic control in presence of time-delay in urban road networks. Transp. Res. Part C Emerg. Technol. 59, 308–322. https://doi.org/10.1016/j.trc.2015.04.031
Keyvan-Ekbatani, M., Yildirimoglu, M., Geroliminis, N., Papageorgiou, M., 2015b. Multiple concentric gating traffic control in large-scale urban networks. IEEE Trans. Intell. Transp. Syst. 16, 2141–2154. https://doi.org/10.1109/TITS.2015.2399303
Kingma, D.P., Ba, J.L., 2015. Adam: A method for stochastic optimization, in: 3rd International

Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings. International Conference on Learning Representations, ICLR.
Kouvelas, A., Saeedmanesh, M., Geroliminis, N., 2017. Enhancing model-based feedback perimeter control with data-driven online adaptive optimization. Transp. Res. Part B Methodol. 96, 26–45. https://doi.org/10.1016/j.trb.2016.10.011
Kraft, D., 1988. A software package for sequential quadratic programming. Tech. Rep. DFVLR-FB 88-28.
Lecun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature. https://doi.org/10.1038/nature14539
Lei, T., Hou, Z., Ren, Y., 2019. Data-Driven Model Free Adaptive Perimeter Control for Multi-Region Urban Traffic Networks With Route Choice. IEEE Trans. Intell. Transp. Syst. 1–12. https://doi.org/10.1109/tits.2019.2921381
Li, D., Hou, Z., 2020. Perimeter Control of Urban Traffic Networks Based on Model-Free Adaptive Control. IEEE Trans. Intell. Transp. Syst. 1–13. https://doi.org/10.1109/tits.2020.2992337
Li, L., Lv, Y., Wang, F.Y., 2016. Traffic signal timing via deep reinforcement learning. IEEE/CAA J. Autom. Sin. 3, 247–254. https://doi.org/10.1109/JAS.2016.7508798
Liang, X., Du, X., Wang, G., Fellow, Z.H., 2018. Deep Reinforcement Learning for Traffic Light Control in Vehicular Networks, IEEE Transactions on Automatic Control.
Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D., 2016. Continuous control with deep reinforcement learning, in: 4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings. International Conference on Learning Representations, ICLR.
Lin, L.-J., 1992. Self-improving reactive agents based on reinforcement learning, planning and teaching. Mach. Learn. 8, 293–321. https://doi.org/10.1007/bf00992699
Mahmassani, H., Williams, J.C., Herman, R., 1987. Performance of urban traffic networks, in: 10th International Symposium on Transportation and Traffic Theory.
Mahmassani, H., Williams, J.C., Herman, R., 1984. Investigation of network-level traffic flow relationships: Some simulation results. Transp. Res. Rec. J. Transp. Res. Board 971, 121–130.
Mahmassani, H.S., Saberi, M., Zockaie, A., 2013. Urban network gridlock: Theory, characteristics, and dynamics. Transp. Res. Part C Emerg. Technol. 36, 480–497. https://doi.org/10.1016/j.trc.2013.07.002
Mazloumian, A., Geroliminis, N., Helbing, D., 2010. The spatial variability of vehicle densities as determinant of urban network capacity 368, 4627–4647. https://doi.org/10.1098/rsta.2010.0099
Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D., 2015. Human-level control through deep reinforcement learning. Nature 518, 529–533. https://doi.org/10.1038/nature14236
Nagle, A.S., Gayah, V. V., 2014. Accuracy of Networkwide Traffic States Estimated from Mobile Probe Data. Transp. Res. Rec. J. Transp. Res. Board 1–11. https://doi.org/10.3141/2421-01
Ni, W., Cassidy, M., 2020. City-wide traffic control: Modeling impacts of cordon queues. Transp. Res. Part C Emerg. Technol. 113, 164–175. https://doi.org/10.1016/j.trc.2019.04.024
Ni, W., Cassidy, M.J., 2019. Cordon control with spatially-varying metering rates: A Reinforcement Learning approach. Transp. Res. Part C Emerg. Technol. 98, 358–369.

https://doi.org/10.1016/j.trc.2018.12.007
Nocedal, J., Wright, S., 2006. Numerical Optimization. Springer Science & Business Media.
Prabhu, S., George, K., 2014. Performance improvement in MPC with time-varying horizon via switching, in: IEEE International Conference on Control and Automation, ICCA. IEEE Computer Society, pp. 168–173. https://doi.org/10.1109/ICCA.2014.6870915
Ramezani, M., Haddad, J., Geroliminis, N., 2015. Dynamics of heterogeneity in urban networks: Aggregated traffic modeling and hierarchical control. Transp. Res. Part B Methodol. 74, 1–19. https://doi.org/10.1016/j.trb.2014.12.010
Ren, Y., Hou, Z., Lei, T., 2018. Two-region macroscopic traffic network perimeter control via model free adaptive control based strategy, in: 2017 Asian Control Conference, ASCC 2017. Institute of Electrical and Electronics Engineers Inc., pp. 899–904. https://doi.org/10.1109/ASCC.2017.8287290
Ren, Y., Hou, Z., Sirmatel, I.I., Geroliminis, N., 2020. Data driven model free adaptive iterative learning perimeter control for large-scale urban road networks. Transp. Res. Part C Emerg. Technol. 115, 102618. https://doi.org/10.1016/j.trc.2020.102618
Rummery, G.A., Niranjan, M., 1994. On-Line Q-Learning Using Connectionist Systems.
Saberi, M., Mahmassani, H.S., Hou, T., Zockaie, A., 2014. Estimating Network Fundamental Diagram Using Three-Dimensional Vehicle Trajectories. Transp. Res. Rec. J. Transp. Res. Board 2422, 12–20. https://doi.org/10.3141/2422-02
Schaul, T., Quan, J., Antonoglou, I., Silver, D., 2016. Prioritized experience replay, in: 4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings. International Conference on Learning Representations, ICLR.
Schrangl, P., Ohtsuka, T., Del Re, L., 2018. Parameter sensitivity reduction of nonlinear model predictive control for discrete-time systems, in: 2017 Asian Control Conference, ASCC 2017. Institute of Electrical and Electronics Engineers Inc., pp. 2131–2136. https://doi.org/10.1109/ASCC.2017.8287504
Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., Riedmiller, M., 2014. Deterministic Policy Gradient Algorithms, in: 31st International Conference on Machine Learning. Beijing, China.
Sirmatel, I.I., Geroliminis, N., 2018. Economic Model Predictive Control of Large-Scale Urban Road Networks via Perimeter Control and Regional Route Guidance. IEEE Trans. Intell. Transp. Syst. 19, 1112–1121. https://doi.org/10.1109/TITS.2017.2716541
Su, Z.C., Chow, A.H.F., Zheng, N., Huang, Y.P., Liang, E.M., Zhong, R.X., 2020. Neuro-dynamic programming for optimal control of macroscopic fundamental diagram systems. Transp. Res. Part C Emerg. Technol. 116, 102628. https://doi.org/10.1016/j.trc.2020.102628
Sutton, R.S., Barto, A.G., 2018. Reinforcement learning: An introduction. MIT Press.
Sutton, R.S., Mcallester, D., Singh, S., Mansour, Y., 2000. Policy Gradient Methods for Reinforcement Learning with Function Approximation, Advances in Neural Information Processing Systems. MIT Press.
Thrun, S., Schwartz, A., 1993. Issues in Using Function Approximation for Reinforcement Learning.
Tsitsiklis, J.N., Roy, B. Van, 1997. An Analysis of Temporal-Difference Learning with Function Approximation, IEEE Transactions on Automatic Control.
van Hasselt, H., 2010. Double Q-learning, in: Advances in Neural Information Processing Systems. pp. 2613–2621.
van Hasselt, H., Guez, A., Silver, D., 2015. Deep Reinforcement Learning with Double Q-learning.

30th AAAI Conf. Artif. Intell. AAAI 2016 2094–2100.
Wächter, A., Biegler, L.T., 2006. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math. Program. 106, 25–57. https://doi.org/10.1007/s10107-004-0559-y
Wang, Z., Schaul, T., Hessel, M., van Hasselt, H., Lanctot, M., de Freitas, N., 2015. Dueling Network Architectures for Deep Reinforcement Learning. 33rd Int. Conf. Mach. Learn. ICML 2016 4, 2939–2947.
Watkins, C.J.C.H., Dayan, P., 1992. Q-learning. Mach. Learn. 8, 279–292. https://doi.org/10.1007/bf00992698
Wei, H., Xu, N., Zhang, H., Zheng, G., Zang, X., Chen, C., Zhang, W., Zhu, Y., Xu, K., Li, Z., 2019. CoLight: Learning Network-level Cooperation for Traffic Signal Control 1913–1922. https://doi.org/10.1145/3357384.3357902
Zheng, N., Geroliminis, N., 2013. On the Distribution of Urban Road Space for Multimodal Congested Networks. Procedia - Soc. Behav. Sci. 80, 119–138. https://doi.org/10.1016/j.sbspro.2013.05.009
Zheng, N., Waraich, R.A., Axhausen, K.W., Geroliminis, N., 2012. A dynamic cordon pricing scheme combining the Macroscopic Fundamental Diagram and an agent-based traffic model. Transp. Res. Part A Policy Pract. 46, 1291–1303. https://doi.org/10.1016/j.tra.2012.05.006