Reinforcement Learning with Model Predictive Control for Highway Ramp Metering

Filippo Airaldi, Bart De Schutter, Azita Dabiri

***Abstract*—In the backdrop of an increasingly pressing need for effective urban and highway transportation systems, this work explores the synergy between model-based and learning-based strategies to enhance traffic flow management by use of an innovative approach to the problem of highway ramp metering control that embeds Reinforcement Learning techniques within the Model Predictive Control framework. The control problem is formulated as an RL task by crafting a suitable stage cost function that is representative of the traffic conditions, variability in the control action, and violations of a safety-critical constraint on the maximum number of vehicles in queue. An MPC-based RL approach, which merges the advantages of the two paradigms in order to overcome the shortcomings of each framework, is proposed to learn to efficiently control an on-ramp and to satisfy its constraints despite uncertainties in the system model and variable demands. Finally, simulations are performed on a benchmark from the literature consisting of a small-scale highway network. Results show that, starting from an MPC controller that has an imprecise model and is poorly tuned, the proposed methodology is able to effectively learn to improve the control policy such that congestion in the network is reduced and constraints are satisfied, yielding an improved performance compared to the initial controller.**

***Index Terms*—Ramp Metering, Reinforcement Learning, Model Predictive Control.**

I. Introduction

OVER the past decades, the need for urban and highway transportation has become significantly relevant to our society. Extended travel times (primarily spent in traffic jams), impact on stress levels and pollution are but a few examples of the negative effects due to high demands for mobility that traffic engineers are facing nowadays [1], [2]. In particular, the necessity for advanced intelligent traffic control strategies is arising as the construction of new infrastructure and upgrades to the existing one become increasingly unsustainable, both in economical, environmental, and societal terms [3], [4].

In the context of highway traffic control, ramp metering control has been proven to be an effective tool in regulating the traffic flow entering the network at on-ramps [5]. This methodology consists in modulating the flow at an on-ramp via traffic lights with appropriate timings of red, green, and amber lights. Earlier approaches were fixed-time, i.e., the timings were calculated offline with the help of historical traffic data [6]. More efficient strategies involve real-time measurements of the traffic conditions that enable online computations of the timings in a traffic-responsive way. In these cases, the underlying algorithms range from simpler feedback strategies [7], [8] to more complex model-based architectures, such as Model Predictive Control (MPC) [9], [10], [11].

Indeed, in modern control literature, MPC is an established control paradigm whose appeal can be attributed to its strong theoretical foundations, a diverse array of applications, as well as its remarkable capability to manage multivariate and constrained systems [12]. The versatility of MPC is further evident in its alternative formulations that allow for the systematic handling of disturbances affecting the system, e.g., in a robust [13] or a stochastic [14] manner. Additionally, unlike other black-box or purely data-driven approaches, MPC-based control strategies often maintain a high degree of explainability and interpretability, which is especially favourable in safety-critical settings. These properties are in part thanks to the prediction model that the MPC scheme contains, which allows it to forecast the dynamical evolution of state trajectories along the time horizon, and which is often designed by domain experts. As aforementioned, also the ramp metering control problem has been tackled with MPC [9], [11], [15], [16], [17].

Nonetheless, it is well-known that the performance of such predictive controllers is heavily bound to the quality of the prediction model, which call for a high level of expertise during the system identification phase. In an attempt to counter this limitation, driven by the recent surge in advancements in Machine Learning (ML) algorithms and facilitated by the growing computational and sensing capabilities of digital systems, there is currently a substantial body of research dedicated to learning-based control methodologies that offer ways of leveraging open- or closed-loop data to, e.g., synthesize models and controllers, or enhance existing ones.

Amongst these methodologies, one of the most notable approaches is Reinforcement Learning (RL), a paradigm of ML that has gained substantial attention due to its advancements and successes. It provides a framework that allows to train an agent to interact with an unknown environment and to make decisions to maximise rewards or minimise costs [18]. The popularity in the contemporary ML community tends to favour its model-free variants that rely solely on observed state transitions and stage cost realisations to increase the performance of the control policy. In regard to ramp metering, while MPC has long been established as a viable tool, only more recently the research field has also embraced such RL approaches. Earlier works focused on tabular Q-learning [19], [20], [21] which enjoy simplicity, some degree of interpretability, and convergence guarantees in small and simple environments

This research is part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 101018826 - CLariNet). (*Corresponding author: Filippo Airaldi.*)

The authors are with the Delft Center for Systems and Control, Delft University of Technology, Mekelweg 2, 2628 CD Delft, The Netherlands (e-mail: f.airaldi@tudelft.nl; b.deschutter@tudelft.nl; a.dabiri@tudelft.nl).

[22]. However, they are limited in handling larger state spaces, exhibit slow convergence, and struggle to generalize. As the field of RL progressed, researchers recognized the potential of function approximators and, in particular, of Neural Networks (NNs) and their deep variants (DNNs) [23], with such deep RL methods nowadays surfacing also in traffic control [24] and in ramp metering [25]. However, while DNNs are a predominant tool in function approximation and have demonstrated great efficacy, in general, they lack interpretability, and the integration or embedding of prior knowledge (e.g., from domain experts) within these network is still difficult, even in an approximated way. Thus, providing formal guarantees on properties relevant to a controller, such as stability and safety, in the context of NN-enhanced control architectures is one of today's challenges in the field.

At the same time, on the other side of the coin, the control community is dedicating great efforts in integrating data-driven techniques with control systems in more structured ways, fostering methodologies that aim to learn various components of a controller directly from data [26], [27]. The goals include, e.g., reducing model uncertainties and the impact of disturbances, improving closed-loop performance and battling conservativeness, or promoting safe and robust control policies. Notably, learning-based MPC methodologies [28] have been devised, e.g., to leverage Gaussian Process regression to learn unmodelled, often nonlinear dynamics in the prediction model via state transition observations [29], or to use Bayesian Optimisation to automatically tune and optimise the controller hyperparameters [30].

Recently, [31] proposed and justified the use of MPC as function approximation of the optimal policy in model-based RL. Fig. 1 depicts a schematic overview of this architecture. In such a scheme, the MPC controller's optimisation problem acts both as policy provider and value function approximation of the RL task. Concurrently, the learning algorithm (such as Q-learning and stochastic/deterministic policy gradient) is tasked with adjusting the parametrisation of the controller in an effort to directly or indirectly discover the optimal policy, thus improving closed-loop performance in a data-driven fashion. In this way, despite the presence of mismatches between the prediction model and the real system, the MPC control scheme is able to learn and deliver at convergence the optimal policy and value functions of the underlying RL task, granted the MPC parametrisation is rich enough. In contrast to DNN-based RL, the key advantages of this approach are multiple. MPC-based RL agents contain exclusively white-box components, thus rendering their behaviour more amenable to analysis and certification in terms of safety and stability [32], whereas NNs are notoriously black-box. Moreover, the MPC framework can easily integrate prior information on the system in the form of an expert-based, possibly imperfect prediction model, and it is accompanied by an extensive body of literature regarding its analysis. For example, though it can be challenging, it is possible to craft suitable terminal components to ensure, e.g., stability and recursive feasibility [12]. It can also take constraints on states and actions into account in an explicit and structured way, which NNs are generally incapable of or can do poorly.

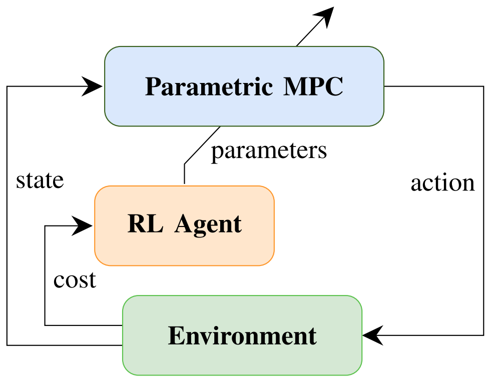

Fig. 1. Diagram of the MPC-based RL architecture

In this paper, inspired by the promising achievements of learning-based control frameworks in various applications [33], [34], [35], a novel approach to tackle the highway ramp metering control problem, combining RL techniques within the MPC framework in accordance to [31], is proposed. This paves the way to a control solution that offers a blend of model-based strategies (leveraging the large amount of information that an expertly-crafted model provides to a control architecture) with learning capabilities (that can exploit the observed data in order to adapt and improve its performance automatically). Likewise, [36] integrates MPC and RL together in a similar traffic task; however, that approach treats the two components as separate (yet collaborating) agents, and it linearly combines their actions. Instead, the method proposed here merges the two into a single agent where the differentiable MPC policy provides actions and is automatically updated by the RL algorithm to increase closed-loop performance.

In summary, this paper contributes to the state-of-the-art by casting the ramp metering control problem as an adaptive control task, and subsequently introducing a promising, Reinforcement Learning-based Model Predictive Control strategy to effectively tackle it, substantiated by numerical simulation results.

The paper is organised as follows. Background on modelling and controlling traffic networks via MPC is provided in Section II. Section III presents the proposed methodology, showing how MPC can be combined with RL, as well as explaining the MPC-based RL algorithm itself. The method is then applied to a numerical case study in Section IV. Finally, Section V concludes the paper with some final remarks and future directions.

A. Notation

The set of indices $\{1, \dots, N\} \subseteq \mathbb{N}$ is denoted as $\mathcal{N}$. Inequalities on vectors are meant to be applied element-wise. To avoid confusion when other subscripts are present, dynamics quantities at time step k can be also referred to as, e.g., $\rho_j(k)$. The i-th index of predicted quantities based on the information available at time step k are denoted as, e.g., $\hat{y}_{i|k}$.

II. Background

Three preliminary components are necessary to illustrate the proposed method. First, a modelling framework must be employed to build a representation of the dynamical behaviour of the highway network, and how ramp metering can influence it to prevents, e.g., bottlenecks and spill-backs. Then, we report a classical formulation of MPC for this task that makes use of such model, and propose some modifications to it that are relevant to this work. Lastly, the essential concepts of RL are briefly introduced.

A. METANET Modelling Framework

Based on the traffic phenomena to investigate, different approaches to modelling a highway network exist [37]. In this paper, the macroscopic second-order METANET framework is employed to obtain a discrete-time dynamical representation of the behaviour of highway traffic under ramp metering control. Proposed as a simulation tool in [38], [39], it made one of its first appearances in combination with MPC in [15], [11], and has since been extended to cover several phenomena and traffic measures [40], [41].

In the remainder of this section, the basics of METANET are introduced. In this framework, a network is represented as a directed graph where each segment (a continuous stretch of highway with homogeneous properties) is modelled as an arc, and origins are represented as nodes. Virtual nodes are also employed to separate segments with heterogeneous properties (e.g., due to a lane drop). We assume such network consists of $|\mathcal{S}|$ segments and $|\mathcal{O}|$ origins, where $\mathcal{S}$ and $\mathcal{O}$ are the sets of indices of segments and origins, respectively. At time step k, segment $j \in \mathcal{S}$ is characterized by its traffic density $\rho_j(k)$ and a mean speed $v_j(k)$, which form its state. Conversely, origin $o \in \mathcal{O}$ is characterized by the queue length $w_o(k)$. In the destination-independent variant, the evolution in time of segment states is described by the following equations:

$$\rho_j(k+1) = \rho_j(k) + \frac{T}{L_j \lambda_j}\big(q_{j-1}(k) - q_j(k) + r_o(k)\big), \quad (1)$$

$$\begin{aligned} v_j(k+1) = v_j(k) &+ \frac{T}{\tau}\Big(V_\text{e}\big(\rho_j(k)\big) - v_j(k)\Big) \\ &+ \frac{T}{L_j} v_j(k)\big(v_{j-1}(k) - v_j(k)\big) \\ &- \frac{\eta T}{\tau L_j}\frac{\rho_{j+1}(k) - \rho_j(k)}{\rho_j(k) + \kappa} \\ &- \frac{\mu T r_o(k) v_j(k)}{L_j \lambda_j \big(\rho_j(k) + \kappa\big)}, \end{aligned} \quad (2)$$

$$q_j(k) = \lambda_j \rho_j(k) v_j(k), \quad (3)$$

$$V_\text{e}\big(\rho(k)\big) \coloneqq v_\text{free} \exp\left(-\frac{1}{a}\left(\frac{\rho(k)}{\rho_\text{crit}}\right)^a\right), \quad (4)$$

where T is the sampling time, L_j and λ_j are respectively the length of the segment and its number of lanes, (4) is the well-known equilibrium speed formula, and $\tau, \eta, \kappa, \mu, a, \rho_\text{crit}, v_\text{free}$ are model parameters describing different quantities and phenomena. In particular, ρ_crit indicates the traffic density at which traffic flow transitions from free-flow conditions to congested conditions. At this density, the flow rate of vehicles through the network is at its maximum, and any increase in the number of vehicles beyond this point results in reduced speeds and increased congestion. The free-flow speed v_free describes instead the speed at which vehicles can travel when the road is not congested or disrupted, i.e., it is the maximum speed that vehicles can achieve on a particular segment under low traffic volumes. Lastly, $r_o(k)$ is the incoming flow generated by origin o connected to segment j; if none is connected, it is null. In these equations, the index $j-1$ refers to the segment upstream of j, and $j+1$ to the segment downstream. For origins, the queue length evolves as

$$w_o(k+1) = w_o(k) + T\big(d_o^w(k) - r_o(k)\big), \quad (5)$$

where the origin's demand $d_o^w(k)$ acts as an uncontrollable external input affecting the queue dynamics. This work is mainly concerned with on-ramps; however, different types of models for origins exist, and interested readers are referred to [40]. Having defined the queue length as in (5), the on-ramp flow can be computed as

$$r_o(k) = \tilde{r}_o(k) \min\Bigg\{ d_o^w(k) + \frac{w_o(k)}{T}, C_o, C_o\left(\frac{\rho_\text{max} - \rho_j(k)}{\rho_\text{max} - \rho_\text{crit}}\right)\Bigg\}, \quad (6)$$

where C_o is the capacity of the origin, ρ_max is another model parameter, $\rho_j(k)$ is the density of the segment j that is connected to this ramp. Lastly, $\tilde{r}_o(k) \in [0,1]$ is the metering rate, which is regarded as the control action.

Finally, the downstream boundary conditions are here considered to be affected by congestion, i.e., the highway segment j that ends in destination $i \in \mathcal{D}$ is subject to the (virtual) downstream density modelled in [10] as

$$\rho_{j+1}(k) = \max\Big\{\min\big\{\rho_j(k), \rho_\text{crit}\big\}, d_i^\rho(k)\Big\}, \quad (7)$$

where $\mathcal{D}$ is the set of destination indices, $d_i^\rho(k)$ models the congestion density of destination i, and is regarded as another external input.

In a more concise formulation, the states, actions and external factors at time k can be lumped together in vectors as

$$\begin{aligned} x_k &= \big[\rho_1(k) \ \ \dots \ \ \rho_{|\mathcal{S}|}(k) \ \ v_1(k) \ \ \dots \ \ v_{|\mathcal{S}|}(k) \\ &\qquad\qquad w_1(k) \ \ \dots \ \ w_{|\mathcal{O}|}(k)\big]^\top, \end{aligned} \quad (8)$$

$$u_k = \big[\tilde{r}_1(k) \ \ \dots \ \ \tilde{r}_{|\mathcal{O}|}(k)\big]^\top, \quad (9)$$

$$d_k = \Big[d_1^w(k) \ \ \dots \ \ d_{|\mathcal{O}|}^w(k) \ \ d_1^\rho(k) \ \ \dots \ \ d_{|\mathcal{D}|}^\rho(k)\Big]^\top, \quad (10)$$

respectively. Then, the METANET framework can be exploited to build the nonlinear dynamics

$$x_+ = f(x, u, d), \quad (11)$$

where $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_d} \to \mathbb{R}^{n_x}$, with $n_x = 2|\mathcal{S}| + |\mathcal{O}|$, $n_u = |\mathcal{O}|$, $n_d = |\mathcal{O}| + |\mathcal{D}|$, and x_+ indicates the state at the next time step. Having the dynamics described as in (11), they can be used for designing a model-based controller.

B. MPC for Ramp Metering Control

Model Predictive Control (MPC) [12] is a well-known control framework for dynamical systems that is based on the formulation of a finite-horizon optimal control problem, consisting of three fundamental pillars: the prediction of the evolution of the system's dynamics over a finite window, a suitable objective function to be minimised along this prediction window, and the inclusion of constraints on state and/or action variables. The first is typically achieved by means of a (often approximate) model of the system's behaviour, which is used to simulate the evolution of the states along the window. The second is a function that quantifies the performance of the controller, and is often a trade-off between conflicting objectives. Thirdly, MPC allows to systematically integrate into its structure constraints on the states and/or actions, thus yielding policies that are able to take these constraints into account in an optimal way. On the whole, the framework offers quite a degree of flexibility, and it is unsurprising that it lends itself also to the ramp metering control problem.

We define the prediction horizon N_p and the control horizon $N_\text{c} \leq N_\text{p}/M$, where M is a natural number to distinguish the process timescale T from the controller timescale $T_\text{c} = T/M$. In other words, the process runs M times faster than the controller, with the MPC being solved only every M time steps (instead of at each time step). Once solved, the first optimal action is then applied for the next M time steps, in a receding horizon fashion. Therefore, in the MPC controller timescale, control action indices are indicated as i_c and can be related to indices i of the dynamics timescale as $i_c(i) = \min\{\lfloor i/M \rfloor, N_\text{c} - 1\}$.[1]

Given the current state x_k of the network, the controller is given by

$$\begin{aligned}
\min_{\hat{u}_k, \hat{x}_k} \quad & \sum_{i=0}^{N_\text{p}} L_\text{T}(\hat{x}_{i|k}) + \xi \sum_{i=0}^{N_\text{c}-1} L_\text{V}(\hat{u}_{i|k}) && \text{(12a)} \\
\text{s.t.} \quad & \hat{x}_{0|k} = x_k && \text{(12b)} \\
& \hat{x}_{i+1|k} = f(\hat{x}_{i|k}, \hat{u}_{i_c(i)|k}, d_k), \; i = 0, \dots, N_\text{p} - 1 && \text{(12c)} \\
& g(\hat{x}_{i|k}, \hat{u}_{i_c(i)|k}) \leq 0 \qquad i = 0, \dots, N_\text{p} && \text{(12d)}
\end{aligned}$$

where the optimisation variables are the actions and states along the MPC control and prediction horizons, respectively

$$\hat{u}_k = \begin{bmatrix} \hat{u}_{0|k}, & \dots, & \hat{u}_{N_\text{c}-1|k} \end{bmatrix} \in \mathbb{R}^{n_u \times N_\text{c}}, \tag{13}$$

$$\hat{x}_k = \begin{bmatrix} \hat{x}_{0|k}, & \dots, & \hat{x}_{N_\text{p}|k} \end{bmatrix} \in \mathbb{R}^{n_x \times (N_\text{p}+1)}, \tag{14}$$

As aforementioned, once this optimisation problem is solved, we apply the optimal control action $\hat{u}^\star_{0|k}$ from time step k to $k + M - 1$, as per the receding horizon approach.

The objective (12a) comprises two terms. The former is often the total time spent (TTS), a metric frequently used to quantify efficiency of traffic control, where the function $L_\text{T} : \mathbb{R}^{n_x} \to \mathbb{R}$ is defined as

$$L_\text{T}(x_k) \coloneqq T \left(\sum_{j \in \mathcal{S}} L_j \lambda_j \rho_j(k) + \sum_{o \in \mathcal{O}} w_o(k) \right). \tag{15}$$

The latter term of (12a) is usually a cost proportional to the variability of the inputs along the control horizon, where $L_\text{V} : \mathbb{R}^{n_u} \to \mathbb{R}$ is

$$L_\text{V}(u_k) \coloneqq \sum_{o \in \mathcal{O}} \Big(\tilde{r}_o(k) - \tilde{r}_o(k-1) \Big)^2. \tag{16}$$

The scalar ξ governs the trade-off between these two terms. Note that in (12b) we assume that the current process state x_k is fully measurable. If this is not the case, an observer is required to provide an estimate of it. Likewise, in (12c) we assume the external inputs d_k to be fully known. If this is not the case, a forecast of the evolution of these quantities along the MPC horizon is required. Lastly, (12d) can be used to include any constraint to be imposed on the state-action pair, if necessary.

C. Modifications to the MPC Formulation

In this work, the two following modifications are applied to the MPC formulation (12).

Firstly, in order to emulate some sort of safety-critical scenario, we seek control policies that prevent the queue length $w_{\tilde{o}}$ within a particular ramp $\tilde{o} \in \mathcal{O}$ from exceeding a safe threshold w_max. A similarly constrained scenario can be found in, e.g., [11]. Given the MPC setting, it is straightforward to incorporate such a requirement via the soft constraint

$$h_0(x_k, \sigma_k) \coloneqq w_{\tilde{o}}(k) - w_\text{max} - \sigma_k, \tag{17}$$

where $\sigma_k \geq 0$ is a slack variable (an additional optimisation variable) whose purpose is to avoid feasibility issues due to this additional constraint. The value of this variable must be then penalised in the objective as in (21). We opt for a soft constraint because 1) it is assumed that small violations to this constraint, despite undesired, are tolerable, and 2) we need a mechanism to both relax constraints during the learning process and penalise violations, in order to allow the agent to learn discerning between safe and unsafe policies.

Secondly, we slightly modify the MPC optimisation problem in an effort to ease the numerical complexity of solving such a nonlinear problem. As it turns out, some numerical issues can arise when solving (12) with a gradient-based solver and with the flow at on-ramp given by (6). This is due to the $\min$ operator, which can cause the gradient to be zero over a vast region of the state-action space. This is especially prone to problems when the MPC scheme is embedded within an RL algorithm (as described in the next section). Therefore, instead of controlling the metering rate $\tilde{r}_o(k)$ to influence the flow $r_o(k)$ at on-ramp o, we make the choice to control $r_o(k)$ directly. The implications are the following:

1) the action is now defined as $u_k = \begin{bmatrix} r_1(k) & \dots & r_O(k) \end{bmatrix}^\top$;
2) to make sure that for any on-ramp flow that the MPC controller could output as control action there exists a

[1]This definition of $i_c(i)$ entails that the last action is kept constant for the remainder of the prediction horizon, a common choice in literature, though more complex solutions could be employed.

rate that can achieve it, the following hard constraints must be taken into account

$$h_1(x_k, u_k) := r_o(k) - d_o^w(k) - \frac{w_o(k)}{T}, \tag{18}$$

$$h_2(x_k, u_k) := r_o(k) - C_o \left(\frac{\rho_{\text{max}} - \rho_j(k)}{\rho_{\text{max}} - \rho_{\text{crit}}} \right); \tag{19}$$

3) the dynamics (11) remain mostly unchanged, but (6) becomes irrelevant as its role is already taken care of by (18) and (19);
4) the cost term (16) is also updated to

$$L_{\text{V}}(u_k) := \sum_{o \in \mathcal{O}} \left(\frac{r_o(k) - r_o(k-1)}{C_o} \right)^2. \tag{20}$$

All together, these measures ensure that, after properly re-scaling the weight ξ, the new MPC formulation is equivalent to the previous, yet is easier to solve numerically.

For convenience, let the constraints be grouped as $h(x_k, u_k, \sigma_k) = \begin{bmatrix} h_0(x_k, \sigma_k) & h_1(x_k, u_k) & h_2(x_k, u_k) \end{bmatrix}^\top$. Overall, the final MPC formulation becomes

$$\begin{aligned} \min_{\hat{u}_k, \hat{x}_k, \sigma} \quad & \sum_{i=0}^{N_{\text{p}}} L_{\text{T}}(\hat{x}_{i|k}) + \xi \sum_{i=0}^{N_{\text{c}}-1} L_{\text{V}}(\hat{u}_{i|k}) + \zeta^\top \sigma \\ \text{s.t.} \quad & \hat{x}_{0|k} = x_k \\ & \hat{x}_{i+1|k} = f(\hat{x}_{i|k}, \hat{u}_{i_c(i)|k}, d_k), \ i = 0, \dots, N_{\text{p}} - 1 \\ & h(\hat{x}_{i|k}, \hat{u}_{i_c(i)|k}, \sigma_k) \leq 0 \qquad i = 0, \dots, N_{\text{p}} \\ & \sigma \geq 0, \end{aligned} \tag{21}$$

where $\sigma = \begin{bmatrix} \sigma_0 & \dots & \sigma_{N_{\text{p}}} \end{bmatrix}^\top$ is the collection of slack variables along the prediction horizon, and $\zeta \in \mathbb{R}^{N_{\text{p}}+1}$ is the corresponding vector of positive weights for slack penalty.

D. Reinforcement Learning

To better understand how MPC can be integrated with RL, a brief introduction to the latter is here provided. In what follows, we abide by the canonical RL approach [18]. Given the continuous state s and action a, a discrete-time Markov Decision Process with state transitions $s \xrightarrow{a} s_+$ and underlying conditional probability density

$$\mathbb{P}[s_+ | s, a] : \mathbb{R}^{n_s} \times \mathbb{R}^{n_s} \times \mathbb{R}^{n_a} \to [0, 1] \tag{22}$$

is used to model the discrete-time system dynamics. The performance of a given deterministic policy $\pi_\theta : \mathbb{R}^{n_s} \to \mathbb{R}^{n_a}$ parametrized in $\theta \in \mathbb{R}^{n_\theta}$ is defined as

$$J(\pi_\theta) := \mathbb{E} \left[\sum_{k=0}^{\infty} \gamma^k L\big(s_k, \pi_\theta(s_k)\big) \right], \tag{23}$$

where $L : \mathbb{R}^{n_s} \times \mathbb{R}^{n_a} \to \mathbb{R}$ is a stage cost function and $\gamma \in (0, 1]$ the discount factor. The goal of the RL algorithm is then to find the optimal policy $\pi_\theta^\star$ as

$$\pi_\theta^\star = \underset{\theta}{\arg\min} \quad J(\pi_\theta). \tag{24}$$

Since it is in general impossible to find and characterise the true unknown optimal value functions and policy $V_\star$, $Q_\star$, $\pi_\star$, function approximation techniques (such as DNN and, as in this paper, MPC) have been employed as a powerful alternative for tackling problem (24) [23]. Depending on the algorithm, these allow to formulate the approximations V_θ, Q_θ, and π_θ, which are then used to solve (24) either directly or indirectly via iterative gradient updates of the parametrisation

$$\theta \leftarrow \theta - \alpha \nabla_\theta \sum_{i=1}^{m} \psi\left(s_i, a_i, s_{i+1}, \theta\right), \tag{25}$$

where $\alpha \in \mathbb{R}_+$ is the learning rate, m the size of the batch of observations used in the update, and ψ captures the controller's performance and varies from algorithm to algorithm. Direct (or policy-based) methods explicitly parametrise and learn the approximation π_θ of the optimal policy itself. For instance, policy gradient methods attempt to solve (24) directly by moving along the gradient of the policy, i.e.,

$$\mathbb{E}\left[\psi\left(s_i, a_i, s_{i+1}, \theta\right)\right] = J(\pi_\theta). \tag{26}$$

On the other hand, indirect (or value-based) methods opt to indirectly derive the optimal policy by estimating and learning the value functions V_θ, Q_θ from the underlying RL task, and implicitly derive the policy from these. A member of this category, Q-learning learns to approximate the action-value by solving

$$\min_\theta \quad \mathbb{E}\left[\|Q_\star(s, a) - Q_\theta(s, a)\|^2 \right], \tag{27}$$

with the hope of indirectly recovering the optimal policy from it. In its first-order, recursive (i.e., with $m = 1$) formulation, θ is updated via (25) where

$$\psi(s_i, a_i, s_{i+1}, \theta) = \big(L(s_i, a_i) + \gamma V_\theta(s_{i+1}) - Q_\theta(s_i, a_i)\big)^2. \tag{28}$$

III. MPC-based RL for Ramp Metering

As discussed in Section I, the closed-loop performance of the MPC controller (12) or (21) is dependent on its components, especially the prediction model. Mismatches in the dynamics can undermine the ability of MPC to reliably predict the system behaviour, and thus impacting the controller performance and its ability to avoid constraint violations (especially relevant of safety-critical applications). The highly nonlinear nature of the METANET model contributes to making this issue even more relevant, since the model calibration requires the application of some nonlinear technique, e.g., nonlinear least-squares [42]. In what follows, we show how these shortcomings can be addressed by leveraging online closed-loop data to learn most of the MPC parameters automatically via RL.

A. Ramp Metering as an RL Task

To appropriately apply an RL algorithm to the ramp metering task, we first need to define a proper stage cost to quantify the performance of a policy in controlling the metering, as per (23). Consider the stage cost

$$L(x_k, u_k) := c_{\text{T}} L_{\text{T}}(x_k) + c_{\text{V}} L_{\text{V}}(u_k) + c_{\text{C}} L_{\text{C}}(x_k), \tag{29}$$

where $L_{\text{C}} : \mathbb{R}^{n_x} \to \mathbb{R}$ is

$$L_{\text{C}}(x_k) \coloneqq \max\big\{0, w_{\tilde{o}}(k) - w_{\text{max}}\big\}. \tag{30}$$

The first term in (29) penalises the time spent travelling through the network and waiting in queues, at the current time step k. This term is representative of the TTS criterion, which is often the primary target of ramp metering control strategies in literature. The second term penalises variations of the current control action $r_o(k)$ compared to the previous instant $r_o(k-1)$, and attempts to punish policies that are too jerky in the input signal. Lastly, the third contribution penalises violations in the queue constraint, so that the agent is encouraged to come up with policies that are able to satisfy it. Scalars c_{T}, c_{V}, and c_{C} are weights that balance each term differently. Next, we delve into how MPC can be used as function approximation to solve the ramp metering RL task.

B. MPC as Function Approximation in RL

In Section I it was discussed how [31] suggests the employment of a parametric MPC scheme as function approximation in the RL problem, and lies the foundations of its theory. In this paper, we follow this paradigm. In particular, we choose to parameterise the cost function, the prediction model as well as some of the constraints in (21). Then, the RL algorithm is applied to adjust the parametrisation based on observed state transitions and rewards, in order to achieve better closed-loop performance in terms of (23).

Consider the MPC controller with parametrisation θ

$$\begin{aligned}
\min_{\hat{u}_k, \hat{x}_k, \sigma} \quad & \sum_{i=0}^{N_{\text{p}}} \gamma^i \Big(\theta_{\text{T}} L_{\text{T}}(\hat{x}_{i|k}) + \theta_{\text{C},i}\sigma_i\Big) \\
& + \theta_{\text{V}} \sum_{i=0}^{N_{\text{c}}-1} \gamma^{Mi} L_{\text{V}}(\hat{u}_{i|k}) + \lambda_\theta(\hat{x}_{0|k}) \\
& + \sum_{i=1}^{N_{\text{p}}-1} \gamma^i \ell_\theta(x_k) + \gamma^{N_{\text{p}}} \Gamma_\theta(x_{N_{\text{p}}}) && \text{(31a)} \\
\text{s.t.} \quad & \hat{x}_{0|k} = x_k && \text{(31b)} \\
& \hat{x}_{i+1|k} = f_\theta(\hat{x}_{i|k}, \hat{u}_{i_c(i)|k}, d_k),\ i = 0, \dots, N_{\text{p}} - 1, && \text{(31c)} \\
& h_\theta(\hat{x}_{i|k}, \hat{u}_{i_c(i)|k}, \sigma_k) \leq 0, \qquad i = 0, \dots, N_{\text{p}}, && \text{(31d)} \\
& \sigma \geq 0. && \text{(31e)}
\end{aligned}$$

Note that the objective has the additional parameterised terms $\lambda_\theta, \ell_\theta, \Gamma_\theta : \mathbb{R}^{n_x} \to \mathbb{R}$. These are the learnable initial, stage, and terminal costs, respectively. The initial cost is a linear combination of the state

$$\lambda_\theta(x_k) \coloneqq \sum_{j \in \mathcal{S}} \left(\theta_{\lambda,j}^{\rho} \frac{\rho_j(k)}{\rho_{\text{max}}} + \theta_{\lambda,j}^{v} \frac{v_j(k)}{v_{\text{max}}}\right) + \sum_{o \in \mathcal{O}} \theta_{\lambda,o}^{w} \frac{w_o(k)}{w_{\text{max}}}, \tag{32}$$

where ρ_{max}, v_{max}, w_{max} are here used as fixed, non-learnable normalization constants in order to improve the stability of the learning process. As explained in [31], since our objective is economic, this initial cost is required in order for the MPC scheme to recover the optimal policy. Conversely, stage and terminal costs are quadratic in nature

$$\begin{aligned}
\ell_\theta(x_k) \coloneqq & \sum_{j \in \mathcal{S}} \theta_{\ell,j}^{\rho} \left(\frac{\rho_j(k) - \rho_{\text{sp}}}{\rho_{\text{max}}}\right)^2 \\
& + \sum_{j \in \mathcal{S}} \theta_{\ell,j}^{v} \left(\frac{v_j(k) - v_{\text{sp}}}{v_{\text{max}}}\right)^2 \\
& + \sum_{o \in \mathcal{O}} \theta_{\ell,o}^{w} \left(\frac{w_o(k)}{w_{\text{max}}}\right)^2,
\end{aligned} \tag{33}$$

$$\begin{aligned}
\Gamma_\theta(x_k) \coloneqq & \sum_{j \in \mathcal{S}} \theta_{\Gamma,j}^{\rho} \left(\frac{\rho_j(k) - \rho_{\text{sp}}}{\rho_{\text{max}}}\right)^2 \\
& + \sum_{j \in \mathcal{S}} \theta_{\Gamma,j}^{v} \left(\frac{v_j(k) - v_{\text{sp}}}{v_{\text{max}}}\right)^2 \\
& + \sum_{o \in \mathcal{O}} \theta_{\Gamma,o}^{w} \left(\frac{w_o(k)}{w_{\text{max}}}\right)^2,
\end{aligned} \tag{34}$$

where densities and speeds are encouraged to track the fixed, non-learnable setpoints ρ_{sp} and v_{sp}, respectively, and the queues to be as small as possible. Regarding the METANET model f_θ in (31c), we allow the RL algorithm to adjust two fundamental parameters in the dynamics, i.e., $\tilde{\rho}_{\text{crit}}$ and $\tilde{a}$ (which in general can differ, during the learning process, from their counterparts ρ_{crit}, a of the real dynamics). In this way, the agent is able to partially tune the prediction model based on performance (23) rather than on system identification, implying that these learning parameters might grow to different values compared to the ones belonging to the true underlying dynamics. Finally, also constraints h_θ (31d) get parameterised, since $\tilde{\rho}_{\text{crit}}$ appears in it (see (19)).

Remark 1: In general, a parametrisation that makes use of quadratic terms with fixed setpoints, as in (33) and (34), can lead to sub-optimal MPC policies in low-demand, free-flow regimes because it penalises low densities and low speeds as much as it penalises high densities and high speeds. At the same time, the theory at the foundation of MPC-based RL assumes the controller's parametrisation to be rich enough to adequately capture the real value function [31]. For this reason, in this work, we opt for the aforementioned adjustable quadratic cost terms. That been said, one could envision more involved and more performing learnable cost terms, especially in more complex and difficult traffic settings. For instance, asymmetric penalty functions are a valid choice, e.g., a left-sided Huber loss that quadratically penalises high traffic quantities, but penalises low ones only linearly.

The whole parametrisation is

$$\theta = \begin{bmatrix} \tilde{\rho}_{\text{crit}} & \tilde{a} & \theta_{\{\text{T,V,C}\}} & \theta_{\{\lambda,\ell,\Gamma\}}^{\{\rho,v,w\}} \end{bmatrix}^\top. \tag{35}$$

Table I shows a summary of this parametrisation, reporting the scope and the space of each term. As reported in Table II in the next section, these real spaces are often bounded in order to avoid undesired consequences, e.g., to preserve convexity of the parametric cost terms, or to prevent parameters that have some physical meaning from taking unrealistic values. Moreover, variations in the proposed parametrisation and

TABLE I
PARAMETRISATION θ OF THE MPC FUNCTION APPROXIMATION (31)

Symbol	Scope	Space
$\tilde{\rho}_{\text{crit}}$	model, cost, constraint	$\mathbb{R}$
$\tilde{a}$	model	$\mathbb{R}$
θ_{T}	cost - TTS weight	$\mathbb{R}$
θ_{V}	cost - control variability weight	$\mathbb{R}$
θ_{C}	cost - slack weights	$\mathbb{R}^{N_{\text{p}}}$
$\theta^{\{\rho,v\}}_{\{\lambda,\ell,\Gamma\}}$	{initial, stage, terminal} cost - $\{\rho, v\}$ weights	$\mathbb{R}^{\lvert\mathcal{S}\rvert}$
$\theta^{w}_{\{\lambda,\ell,\Gamma\}}$	{initial, stage, terminal} cost - w weights	$\mathbb{R}^{\lvert\mathcal{O}\rvert}$

their influence on the learning outcome are discussed in the appendix.

Scheme (31) yields the approximation $V_\theta : \mathbb{R}^{n_x} \to \mathbb{R}$ of the value function as

$$V_\theta(x_k) = \min_{\hat{u}_k, \hat{x}_k, \sigma} \quad (31\text{a}) \qquad \text{s.t.} \quad (31\text{b}) - (31\text{e}). \tag{36}$$

The value function (36) satisfies the fundamental equalities of the Bellman equations [31], so that

$$\begin{aligned} Q_\theta(x_k, u_k) = \min_{\hat{u}_k, \hat{x}_k, \sigma} \quad & (31\text{a}) \\ \text{s.t.} \quad & (31\text{b}) - (31\text{e}), \\ & \hat{u}_{0|k} = u_k. \end{aligned} \tag{37}$$

$$\pi_\theta(x_k) = \arg\min_u \quad Q_\theta(x_k, u). \tag{38}$$

In practice, policy (38) is found as the first optimal action $\hat{u}^\star_{0|k}$ from the $\arg\min$ of (36). The goal of this parametrisation is to give the MPC scheme (31) enough degrees of freedom to sufficiently adapt to the RL task at hand. The cardinal point is that θ must be rich enough to allow the scheme to capture the optimal policy $\pi^\star$ [31]. Of course, the choice of parametrisation is not unique, and other solutions may be viable.

C. Second-Order LSTD Q-learning

In order to adjust the parametrisation θ of (31), a second-order least-squares temporal difference (LSTD) Q-learning algorithm [43] is employed to find the policy π_θ that minimises the closed-loop performance. Q-learning is one of the most well-known indirect, temporal difference algorithms available in RL. In essence, it searches for the parametrisation that best fits the action-value function Q_θ to the observed data, with the hope of indirectly recovering the optimal policy from this approximation. It has also shown very promising results in the context of MPC [44]. The second-order Newton's method, coupled with an experience replay buffer of the past observed transitions [45] and a re-formulation of the Q-fitting problem as a least-squares, ensures faster convergence and higher sample efficiency compared to traditional first-order methods. For details, see [44].

Since Q-learning requires differentiation with respect to θ of the function approximation, i.e., the MPC scheme (37), we follow the canonical approach by leveraging nonlinear programming sensitivity techniques [46] to compute such sensitivities. Prior to any update, the computed Hessian matrix is always modified to be positive-definite by addition of a multiple of the identity matrix [47]. Furthermore, to impose lower and upper bounds on the parameters (as reported in Table II), the vanilla update rule is cast as the following optimisation problem

$$\begin{aligned} \Delta\theta^\star = \arg\min_{\Delta\theta} \quad & \frac{1}{2}\Delta\theta^\top H \Delta\theta + \alpha p^\top \Delta\theta \\ \text{s.t.} \quad & \theta_{\text{lb}} \leq \theta + \Delta\theta \leq \theta_{\text{ub}}, \\ & \Delta\theta_{\text{lb}} \leq \Delta\theta \leq \Delta\theta_{\text{ub}}, \end{aligned} \tag{39}$$

where $\alpha \geq 0$ is the learning rate, p is the gradient, and H is the Hessian. This formulation allows to limit the rate of change of each parameter with $\Delta\theta_{\text{lb}}$ and $\Delta\theta_{\text{ub}}$. The parametrisation is then updated as $\theta \leftarrow \theta + \Delta\theta^\star$.

Lastly, as in [48], exploratory behaviour is ensured during learning by adding to the MPC objective (31a) the perturbation term $q^\top u_0$, with q randomly chosen from, e.g., a normal distribution. When properly calibrated, this perturbation on the first action ensures that enough exploration is added on top of the policy in order to prevent the Q-learning agent from getting stuck in very suboptimal local minima.

IV. NUMERICAL CASE STUDY

To examinate the performance of the proposed method for ramp metering, we implement and simulate the algorithm when applied to a highway network benchmark.

A. Configuration

1) Network Environment: Let us consider a simple highway traffic network as pictured in Fig. 2, which will serve as the RL environment for the task at hand. The network consists of three segments, each 1 km in length and with two lanes. The first segment S_1 is supplied by the uncontrolled mainstream origin O_1, which simulates incoming traffic from the unmodelled upstream region of the network, and is characterized by a capacity of 3500 veh/h. Between segments S_2 and S_3, additional traffic can enter the network via the on-ramp O_2. This origin has capacity $C_2 = 2000$ veh/h, and offers the only control measure in order to avoid congestion in the network. However, we would also like to keep the queue length on this on-ramp to 50 vehicles or fewer. Lastly, traffic exits the network via the only available destination D_1 with congested outflow. As in [11], we use the following parameters to simulate the underlying process dynamics: $T = 10$ s, $\tau = 18$ s, $\eta = 60$ km^2/lane, $\kappa = 40$ veh/km/lane, $\mu = 0.0122$, $\rho_{\text{max}} = 180$ veh/km/lane, $\rho_{\text{crit}} = 33.5$ veh/km/lane, $v_{\text{free}} = 102$ km/h, and $a = 1.867$.

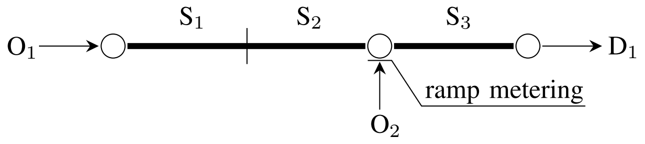

Fig. 2. Structure of the three-segment highway network

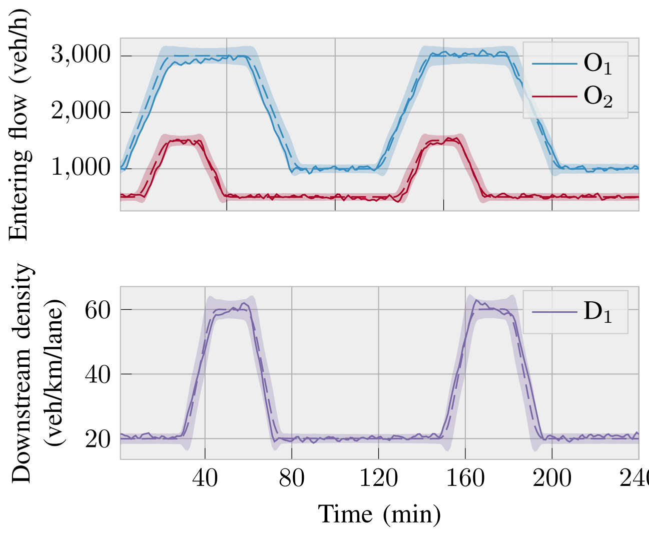

Fig. 3. External inputs affecting the network's dynamics, in the form of the demands at the origins (upper) and the congestion scenario at the destination (lower). The shaded areas represent the 2-standard-deviation ranges from which the random demands are sampled from, whereas the solid lines represent one random sample for each.

As described in Section II-A, the network environment is subject to one external input per origin and destination. Fig. 3 depicts the demands at both origins, starting low and then increasing to near capacity within 20 minutes. With some delay, also a congestion at the destination rapidly appears. These profiles are intended to simulate a peak-hour-like traffic (e.g., morning or evening rush), and are devised in such a way to induce a variable degree of congestion in the network, if the on-ramp is not properly controlled. At the start of each training episode, these profiles are generated randomly, so that a degree of variability is introduced in the task at each episode, with the hope that the agent will be able to generalise its learning to a wider variety of congestion scenarios.

2) Model Predictive Control: The controller (31) is deployed to control the on-ramp flow with the aim of avoiding congestions in the traffic network environment. In line with other similar benchmarks in the literature, e.g., [9], [10], [11], $M = 6$, meaning that, while the process dynamics are stepped every 10 seconds, the controller is run only once per minute. The prediction horizon is set to $N_\text{p} = 24$ (which is in the order of the average travel time through the network), and the control horizon to $N_\text{c} = 3$.

To showcase the ability of RL of automatically improving the performance of the MPC controller, the parametrisation θ of the controller is poorly initialised on purpose. In particular, in order to replicate the effects of a poor system identification phase, the parameters of the prediction model are initialised with a 30% error with respect to their true values, i.e., $\tilde{\rho}_\text{crit} = 0.7\rho_\text{crit}$, $\tilde{a} = 1.3a$ (both learnable), and $1.3v_\text{free}$ (fixed). Furthermore, the remaining parametrisation of the objective function of the MPC optimisation problem is left untuned, imitating a suboptimal, low-expertise design process of the controller. Table II reports the initial values of each learnable

TABLE II
INITIAL VALUES, BOUNDS AND DIMENSIONS OF THE PARAMETRISATION θ OF (31)

Symbol	Initial value	Bounds	Dimension
$\tilde{\rho}_\text{crit}$	23.45	$[10, 162]$	veh/km/lane
$\tilde{a}$	2.4271	$[1.1, 3]$	-
θ_T	1	$[10^{-3}, \infty)$	veh^{-1}h^{-1}
θ_V	160000	$[10^{-3}, \infty)$	veh^2/h^2
θ_C	5	$[10^{-3}, \infty)$	veh^{-1}
$\theta_\lambda^{\{\rho,v,w\}}$	1	$(-\infty, \infty)$	-
$\theta_{\{\ell,\Gamma\}}^{\{\rho,v,w\}}$	1	$[10^{-6}, \infty)$	-

parameter. The other non-learnable parameters in (31) are set to the same values as in the real process. The queue threshold on O_2 is set to $w_\text{max} = 50$ (i.e., $\tilde{o} = 2$). Lastly, the setpoints are set to $\rho_\text{sp} = 0.7\rho_\text{crit}$, $v_\text{sp} = 1.3v_\text{free}$, and the normalization coefficients to $\rho_\text{max} = 180$ veh/km/lane, $w_\text{max} = 50$ veh, and $v_\text{max} = 1.3v_\text{free}$.

3) Reinforcement Learning: The agent is trained in an episodic fashion, with each episode lasting 4 hours, i.e., each episode features two peaks in demands and congestion in a row (for reference, see Fig. 3). At the end of each episode, the demands are generated anew randomly, and the state of the network is reset to steady-state in order to avoid unreasonable accumulation of vehicles in queues from past episodes. With the same frequency, the parametrisation θ is updated, and a new episode is started, till a termination condition of the learning process is met, typically in the form of a maximum number of iterations. Here, we train our agent for 80 episodes.

The stage cost function $L(x_k, u_k)$, with which the traffic network feeds a cost signal back to the agent according to (29), has coefficients $c_\text{T} = 5$, $c_\text{V} = 1600$, and $c_\text{C} = 5$.

Based on results obtained from initial simulations, the algorithm's hyperparameters are set as follows. The discount factor is set to $\gamma = 0.98$. The learning rate is initially set to $\alpha = 0.925$, and decayed by a multiplicative factor of 0.925 at the end of each update. The maximum update change of each parameter is set to 30% of its current value, i.e., $\Delta\theta = 0.3\theta$. As aforementioned, a replay buffer is used to store past observations in memory and re-use them. In particular, the buffer size is set up to store transitions from the 10 past episodes and, before performing an update, a batched sample half its size is drawn from this buffer. In turn, half this batch is dedicated to containing information from the latest two and half episodes, whereas the remaining half is sampled uniformly at random from even older transitions. All together, these measures contribute to stabilising the learning process.

Lastly, exploration is induced as described in Section III-C by sampling the cost perturbation from a normal distribution in an ε-greedy fashion, i.e., $q \sim \mathcal{N}(0, \sigma_q)$ with probability ε; otherwise, $q = 0$. The exploration strength is $\sigma_q = 0.025$, and the exploration probability $\varepsilon = 0.5$. Both are decayed by half at the end of each episode.

Table III summarises all the hyper- and non-learnable parameters of the traffic environment, the MPC controller, and the RL algorithm.

TABLE III
HYPER- AND OTHER NON-LEARNABLE PARAMETERS

METANET	Symbol	T	τ	η	κ	μ	$\rho_{\max}$	ρ_{crit}	v_{free}	a	$C_{\{1,2\}}$
	Value	10	18	60	40	0.0122	180	33.5	102	1.867	{3500, 2000}
	Dimension	s	s	km^2/lane	veh/km/lane	-	veh/km/lane	veh/km/lane	km/h	-	veh/h
MPC	Symbol	M	N_{p}	N_{c}	$w_{\max}$	$v_{\max}$	ρ_{sp}	v_{sp}			
	Value	6	24	3	50	132.6	23.45	132.6			
	Dimension	-	-	-	veh	km/h	veh/km/lane	km/h			
RL	Symbol	c_{T}	c_{V}	c_{C}	batch size	γ	α	α decay	$\Delta\theta$	σ_q	ε
	Value	5	1600	5	5 episodes	0.98	0.925	0.925	0.3	0.025	0.5

4) Setup: The simulations were run on a Linux Ubuntu 20.04.6 server equipped with 16 AMD EPYC 7252 (3.1 GHz) processors and 252GB of RAM, and implemented in Python 3.11.4. The nonlinear optimisation problems were formulated and solved with the symbolic framework CasADi [49] and its interface to the IPOPT solver [50]. The source code and simulation results are open and available in the following repository: https://github.com/FilippoAiraldi/mpcrl-for-ramp-metering.

B. Results

We evaluate the performance of the MPC-based RL agent deployed on the traffic network environment, and average the results over 15 simulations with different seeds to iron out the randomness due to, e.g., exploration. Figures that are shown and discussed next report average results as well as 95% confidence intervals across the 15 runs.

Fig. 4 shows how the RL stage cost evolves during learning in its three contributions. Due to the untuned objective weights and the 30% mismatches in the predictive model's parameters, the initial parametrisation leads to a poorly performing and unsafe controller, i.e., it cannot reliably avoid congestion and constraint violations, respectively, as it can be seen from the costs in the first episodes. However, despite these initial shortcomings, it can be noticed that in the next 20 episodes, by exclusively leveraging the observed transitions via Q-learning, the controller is able to achieve a substantial improvement of the Total-Time-Spent cost, i.e., the time spent by vehicles in the network on average, from around 1100 to 700 veh $\cdot$ h (or a 35% reduction). At the same time, as constraint violations of $w_{\max}$ in O_2 induce the most severe cost realization by several orders of magnitude, one can see that the agent is even quicker in learning to avoid such an unsafe behaviour within the first 5 episodes, despite the fact it has no knowledge of the exact traffic dynamics parameters.

This phenomenon can be further appreciated in Fig. 5, which shows the progression of the average queue in the on-ramp O_2 with respect to its constraint, as the parametrised MPC scheme gets better and better at yielding a safe policy, i.e., with less or no constraint violation. It must be noted here that, despite the non-stationary nature of the traffic environment, the agent is able to learn a policy that is robust to the variability of the randomly generated demand and congestion scenarios, and therefore is capable of avoiding all constraint violations.

Further analysis of the results indicates that, while learning to satisfy the constraint on the on-ramp O_2 is directly ben-

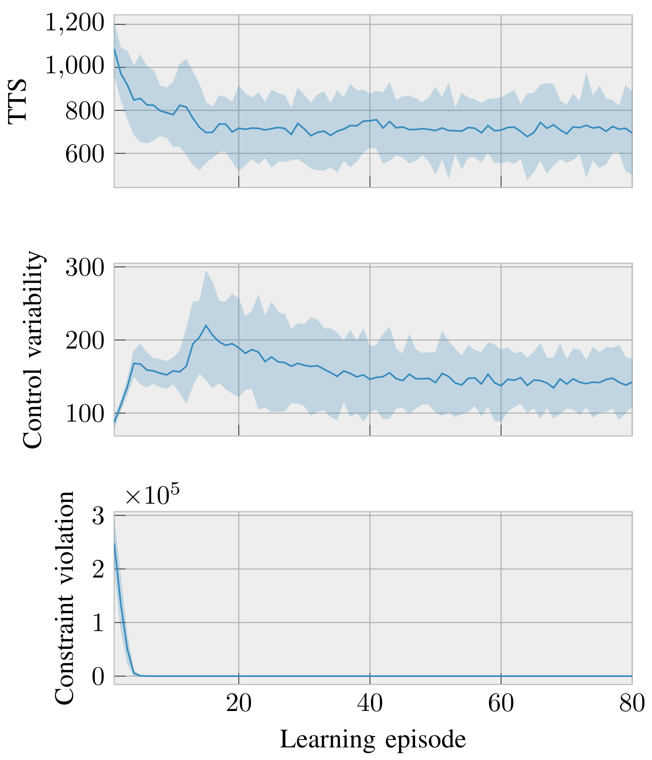

Fig. 4. Evolution of the three contributions to the RL cost (29) during the learning process, namely, from top to bottom, the Total-Time-Spent (TTS), variability of the control action, and violation of the maximum queue constraint on O_2

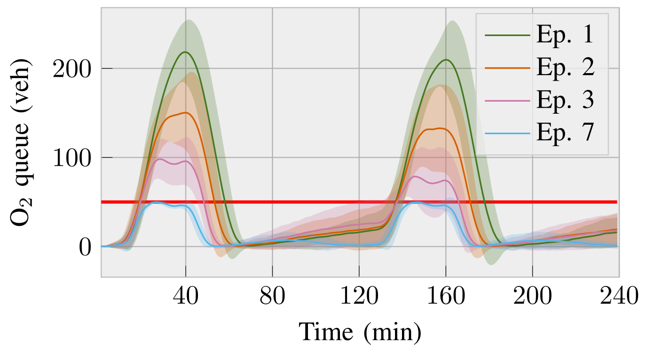

Fig. 5. On average, the policy π_θ gets better at avoiding constraint violations as it learns (the red line represents the threshold $w_{\max}$ imposed on the queue on the on-ramp O_2)

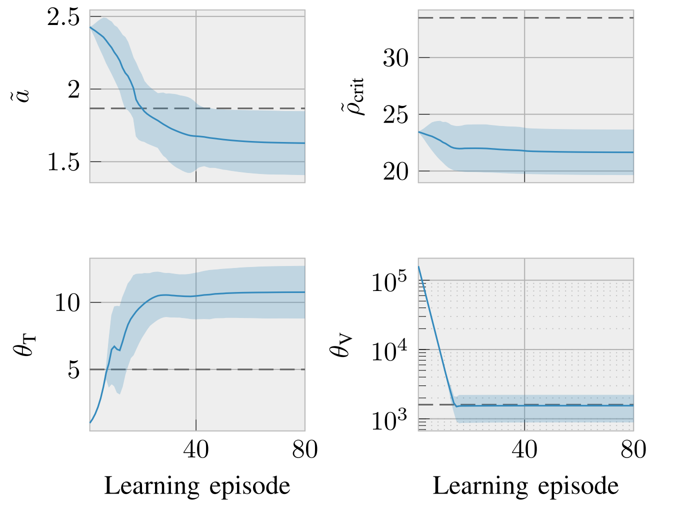

Fig. 6. Evolution of a subset of θ during the learning process (the dashed grey lines represent, in the top plots, the true values of the parameters a and ρ_{crit}, and, in the bottom ones, the constants c_{T} and c_{V})

eficial to the reduction of cost related to violations, it also indirectly fosters better TTS, since longer queues in O_2 proportionally relate to a higher TTS. Nonetheless, improvements to the violation and TTS contributions seem to occur at the expense of the third contribution, i.e., variability of the control action, which clearly is on the rise during learning. However, in the overall context of learning, one must notice that at convergence the sacrifice in control variability (of around 60 points) allows the agent to gain a larger improvement in TTS and to achieve zero constraint violations (which are indeed the largest contributors to the cost of the RL agent). Furthermore, as the learning proceeds past the 16th episode and the other two costs have settled, the agent is also able to achieve further reduction of the cost associated to the control action variability.

From the point of view of the parametrisation θ, the learning process can be in part explained by its evolution, shown in Fig. 6. In particular, one can see that

- the evolution of both θ_{T} and θ_{V} agrees with the observations made in Fig. 4, that is, the agent learns to favour the TTS cost over the control action variability, and thus increases the weight of the former at the expense of the latter
- the parameter $\tilde{a}$, which the METANET model is known to be very sensitive to, develops at first towards its true value a, but then convergences to a slightly lower value, resulting in a less pronounced, less aggressive equilibrium speed V_{e} (4), i.e., favouring predictions of lower speeds at lower densities, and higher speeds at higher densities
- $\tilde{\rho}_{\text{crit}}$ grows in the opposite direction of its true value ρ_{crit}, thus settling for a prediction model that is pessimistic towards congestion, i.e., it tends to overestimate congestion scenarios, as well as resulting in a tighter constraint h_2 (19), which in turn restricts the control action further.

Fig. 7 depicts the TD error during the learning process, whose moving average converges to smaller and smaller values as the episodes increase. This is a good indication that the MPC scheme (31), with its parametrisation θ, is able to provide a reliable approximation Q_θ of the true unknown action-value function (and, indirectly, of the optimal policy). Yet, it is important to note that the convergence of the TD error, while consistently diminishing, does not reach an absolute zero. This observation can be attributed to factors such as imperfect parametrisation and the inherent stochasticity present in the environment, which fluctuates in its demands, and thus renders the prediction of the value functions more challenging.

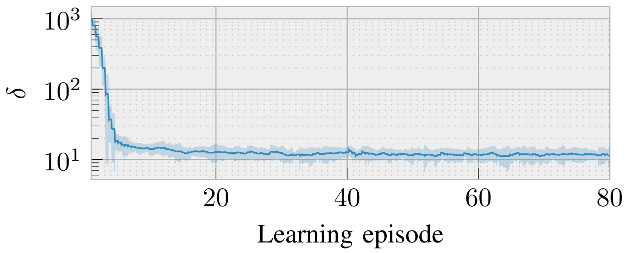

Fig. 7. Moving average of the TD error during the learning process (window length = 239, i.e., number of transitions per episode)

Finally, Fig. 8 reports the evolution of the traffic quantities of the three segments (i.e., density ρ, speed v, and flow q) that make up the network. Interestingly, it can be noticed how, compared to the initial episode, the final controller is able to better prevent the congestion in the last segment from propagating backwards, through the network, and from causing speed drops in the first segment.

V. Conclusions

In this paper, we have proposed a novel learning-based and model-based approach to the ramp metering problem that combines MPC and RL. The two frameworks are integrated in such a way to promote the strengths of each while countering the disadvantages by exploiting the parametrised MPC scheme as a function approximation of the action-value function and leveraging RL to adjust the parametrisation based on observed data to improve closed-loop performance. Even with wrong model parameters and a poorly tuned initial controller, the proposed methodology shows a remarkable ability in learning to improve the traffic control performance and satisfy constraints in an automatic, data-driven fashion.

Future works will focus on 1) more complex parametrisations of the MPC scheme, e.g., with neural networks, in order to better address the nonlinear nature of the METANET framework and to better capture the shape of the true action-value function, as well as 2) a similar approach for the coordinated control of ramps and variable speed limits (VSLs), which has been shown to achieve better results in traffic management than ramp metering alone.

References

[1] O. Johansson, D. Pearce, and D. Maddison, *Blueprint 5: True costs of Road Transport*. Routledge, 2014.
[2] *Global status report on road safety 2018*. Geneva: World Health Organization, 2018.
[3] S. Siri, C. Pasquale, S. Sacone, and A. Ferrara, "Freeway traffic control: A survey," *Automatica*, vol. 130, p. 109655, 2021.

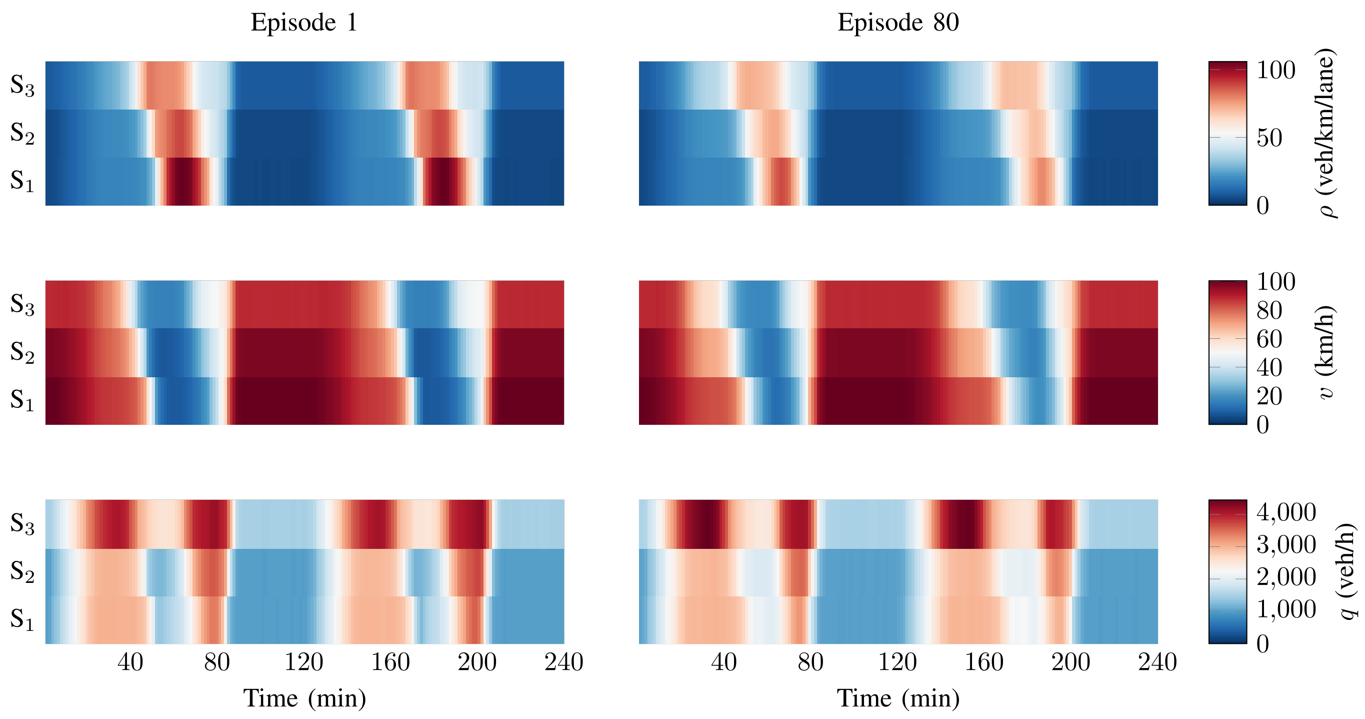

Fig. 8. Differences in traffic quantities of the network's three segments between the first episode and the last episode of the learning process, averaged across the 15 simulation runs

[4] K. Castañeda, O. Sánchez, R. F. Herrera, and G. Mejía, "Highway planning trends: A bibliometric analysis," *Sustainability*, vol. 14, no. 9, p. 5544, 2022.

[5] M. Papageorgiou and A. Kotsialos, "Freeway ramp metering: an overview," *IEEE Transactions on Intelligent Transportation Systems*, vol. 3, no. 4, pp. 271–281, 2002.

[6] J. A. Wattleworth, "Peak-period analysis and control of a freeway system," *Highway Research Record*, vol. 157, pp. 1–21, 1965.

[7] M. Papageorgiou, H. Hadj-Salem, and F. Middelham, "ALINEA local ramp metering: Summary of field results," *Transportation Research Record*, vol. 1603, no. 1, pp. 90–98, 1997.

[8] J. R. D. Frejo and B. De Schutter, "Feed-forward ALINEA: A ramp metering control algorithm for nearby and distant bottlenecks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 7, pp. 2448–2458, 2019.

[9] A. Hegyi, B. De Schutter, H. Hellendoorn, and T. van den Boom, "Optimal coordination of ramp metering and variable speed control-an mpc approach," in *2002 American Control Conference (ACC)*, vol. 5, 2002, pp. 3600–3605.

[10] A. Hegyi, "Model predictive control for integrating traffic control measures," Ph.D. dissertation, Delft University of Technology, 2004.

[11] A. Hegyi, B. De Schutter, and H. Hellendoorn, "Model predictive control for optimal coordination of ramp metering and variable speed limits," *Transportation Research Part C: Emerging Technologies*, vol. 13, no. 3, pp. 185–209, 2005.

[12] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation, and Design*, 2nd ed. Nob Hill Publishing, 2018.

[13] A. Bemporad and M. Morari, "Robust model predictive control: A survey," in *Robustness in Identification and Control*, A. Garulli and A. Tesi, Eds. London: Springer, 1999, pp. 207–226.

[14] A. Mesbah, "Stochastic model predictive control: An overview and perspectives for future research," *IEEE Control Systems Magazine*, vol. 36, no. 6, pp. 30–44, 2016.

[15] T. Bellemans, B. De Schutter, and B. De Moor, "Model predictive control for ramp metering of motorway traffic: A case study," *Control Engineering Practice*, vol. 14, no. 7, pp. 757–767, 2006.

[16] Y. Han, M. Ramezani, A. Hegyi, Y. Yuan, and S. Hoogendoorn, "Hierarchical ramp metering in freeways: An aggregated modeling and control approach," *Transportation Research Part C: Emerging Technologies*, vol. 110, pp. 1–19, 2020.

[17] U. Todorović, J. R. D. Frejo, and B. De Schutter, "Distributed MPC for large freeway networks using alternating optimization," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 3, pp. 1875–1884, 2022.

[18] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[19] M. Davarynejad, A. Hegyi, J. Vrancken, and J. van den Berg, "Motorway ramp-metering control with queuing consideration using Q-learning," in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2011, pp. 1652–1658.

[20] A. Fares and W. Gomaa, "Freeway ramp-metering control based on reinforcement learning," in *11th IEEE International Conference on Control and Automation (ICCA)*. IEEE, 2014, pp. 1226–1231.

[21] E. Ivanjko, D. Koltovska Nečoska, M. Gregurić, M. Vujić, G. Jurković, and S. Mandžuka, "Ramp metering control based on the Q-learning algorithm," *Cybernetics and Information Technologies*, vol. 15, no. 5, p. 88–97, 2015.

[22] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, 1992.

[23] L. Busoniu, R. Babuska, B. De Schutter, and D. Ernst, *Reinforcement Learning and Dynamic Programming using Function Approximators*. CRC Press, 2017.

[24] N. Parvez Farazi, B. Zou, T. Ahamed, and L. Barua, "Deep reinforcement learning in transportation research: A review," *Transportation Research Interdisciplinary Perspectives*, vol. 11, p. 100425, 2021.

[25] F. Belletti, D. Haziza, G. Gomes, and A. M. Bayen, "Expert level control of ramp metering based on multi-task deep reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 4, pp. 1198–1207, 2018.

[26] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based model predictive control: toward safe learning in control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 269–296, 2020.

[27] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe learning in robotics: from learning-based control to safe reinforcement learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, no. 1, pp. 411–444, 2022.

[28] A. Mesbah, K. P. Wabersich, A. P. Schoellig, M. N. Zeilinger, S. Lucia, T. A. Badgwell, and J. A. Paulson, "Fusion of machine learning and

mpc under uncertainty: What advances are on the horizon?" in *2022 American Control Conference (ACC)*. IEEE, 2022, pp. 342–357.

[29] L. Hewing, J. Kabzan, and M. N. Zeilinger, "Cautious model predictive control using Gaussian process regression," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2736–2743, 2020.

[30] D. Piga, M. Forgione, S. Formentin, and A. Bemporad, "Performance-oriented model learning for data-driven MPC design," *IEEE Control Systems Letters*, vol. 3, no. 3, pp. 577–582, 2019.

[31] S. Gros and M. Zanon, "Data-driven economic NMPC using reinforcement learning," *IEEE Transactions on Automatic Control*, vol. 65, no. 2, pp. 636–648, 2020.

[32] ——, "Learning for MPC with stability & safety guarantees," *Automatica*, vol. 146, p. 110598, 2022.

[33] J. Kabzan, L. Hewing, A. Liniger, and M. N. Zeilinger, "Learning-based model predictive control for autonomous racing," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3363–3370, 2019.

[34] B. Chen, Z. Cai, and M. Bergés, "Gnu-RL: A practical and scalable reinforcement learning solution for building HVAC control using a differentiable MPC policy," *Frontiers in Built Environment*, vol. 6, 2020.

[35] W. Cai, A. B. Kordabad, and S. Gros, "Energy management in residential microgrid using model predictive control-based reinforcement learning and shapley value," *Engineering Applications of Artificial Intelligence*, vol. 119, p. 105793, 2023.

[36] W. Remmerswaal, D. Sun, A. Jamshidnejad, and B. De Schutter, "Combined mpc and reinforcement learning for traffic signal control in urban traffic networks," in *2022 26th International Conference on System Theory, Control and Computing (ICSTCC)*. IEEE, 2022, pp. 432–439.

[37] M. Treiber and A. Kesting, *Traffic Flow Dynamics: Data, Models and Simulation*. Berlin, Heidelberg: Springer, 2013.

[38] M. Papageorgiou, J.-M. Blosseville, and H. Hadj-Salem, "Macroscopic modelling of traffic flow on the Boulevard Périphérique in Paris," *Transportation Research Part B: Methodological*, vol. 23, no. 1, pp. 29–47, 1989.

[39] A. Messner and M. Papageorgiou, "METANET: a macroscopic simulation program for motorway networks," *Traffic Engineering & Control*, vol. 31, no. 8-9, pp. 466–470, 1990.

[40] M. Papageorgiou, I. Papamichail, A. Messmer, and Y. Wang, *Traffic Simulation with METANET*. New York: Springer, 2010, pp. 399–430.

[41] A. Dabiri and B. Kulcsár, "Distributed ramp metering—a constrained discharge flow maximization approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 9, pp. 2525–2538, 2017.

[42] D. Ngoduy and S. Hoogendoorn, "An automated calibration procedure for macroscopic traffic flow models," *IFAC Proceedings Volumes*, vol. 36, no. 14, pp. 263–268, 2003.

[43] M. G. Lagoudakis, R. Parr, and M. L. Littman, "Least-squares methods in reinforcement learning for control," in *Methods and Applications of Artificial Intelligence*. Berlin, Heidelberg: Springer, 2002, pp. 249–260.

[44] H. N. Esfahani, A. B. Kordabad, and S. Gros, "Approximate robust NMPC using reinforcement learning," in *2021 European Control Conference (ECC)*. IEEE, 2021, pp. 132–137.

[45] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Machine Learning*, vol. 8, no. 3, pp. 293–321, 1992.

[46] C. Büskens and H. Maurer, "Sensitivity analysis and real-time optimization of parametric nonlinear programming problems," in *Online Optimization of Large Scale Systems*, M. Grötschel, S. O. Krumke, and J. Rambau, Eds. Berlin, Heidelberg: Springer, 2001, pp. 3–16.

[47] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. Springer, 2006.

[48] M. Zanon and S. Gros, "Safe reinforcement learning using robust MPC," *IEEE Transactions on Automatic Control*, vol. 66, no. 8, pp. 3638–3652, 2021.

[49] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[50] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, Mar 2006.

Filippo Airaldi received the B.Sc. and M.Sc. degrees from the Polytechnic University of Turin, Italy, in 2017 and 2019, respectively. He is currently a Ph.D. candidate at the Delft Center for Systems and Control, Delft University of Technology, The Netherlands.

His research interests include model predictive control, reinforcement learning, and other machine learning techniques, and in particular in their combination in learning-based control.

Bart De Schutter (Fellow, IEEE) received the Ph.D. degree (summa cum laude) in applied sciences from KU Leuven, Belgium, in 1996.

He is currently a Full Professor with the Delft Center for Systems and Control, Delft University of Technology, The Netherlands. His research interests include learning and optimisation-based control of large-scale and hybrid systems, multilevel and distributed control, with applications in intelligent transportation and smart energy systems.

Prof. De Schutter is a Senior Editor of the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS and an Associate Editor of the IEEE TRANSACTIONS ON AUTOMATIC CONTROL.

Azita Dabiri received the Ph.D. degree from the Automatic Control Group, Chalmers University of Technology, in 2016. She was a Post-Doctoral Researcher with the Department of Transport and Planning, Delft University of Technology, from 2017 to 2019. In 2019, she received an ERCIM Fellowship and also a Marie Curie Individual Fellowship, which allowed her to perform research at the Norwegian University of Technology (NTNU), as a Post-Doctoral Researcher, from 2019 to 2020, before joining the Delft Center for Systems and Control, TU Delft, as an Assistant Professor. Her research interests are in the areas of integration of model-based and learning-based control and its applications in transportation networks.

Appendix

We provide here additional insights on our proposed methodology and the numerical results. First, we provide a short investigation on the effects of different parametrisations of the MPC scheme (31). Then, we report the evolution of all the parameters θ during the learning process implemented in Section IV.

Different MPC parametrisations

During simulations, different parametrisations θ of the MPC scheme (31) have been tested. In particular, we have considered the following variants:

- learning the dynamics parameters $\tilde{a}$, $\tilde{v}_{\text{free}}$, and/or $\tilde{\rho}_{\text{crit}}$
- employing the learnable dynamics parameters $\tilde{v}_{\text{free}}$, $\tilde{\rho}_{\text{crit}}$ as the tracking setpoints of the stage and terminal cost, i.e., $v_{\text{sp}} = \tilde{v}_{\text{free}}$, $\rho_{\text{sp}} = \tilde{\rho}_{\text{crit}}$.

The rationale behind trying out these various combinations is that, while having a richer and richer parametrisation can potentially help in fitting more and more complex RL functions, a drawback is that the learning task becomes much more complex and more likely to converge to a very suboptimal local minimum. Moreover, the sensitivity of the RL solution to the parametrisation can vary significantly from parameter to parameter, so that learning some of them may end up in a less stable process than others. Unfortunately, finding the right parametrisation largely remains a trail-and-error procedure (both for MPC and other function approximators), whose difficulty is attributable to the RL algorithm itself rather than the proposed control scheme.

Fig. 9 shows the influence on the performance of some of the combinations we tested. These tests were carried out with a lower variability of the randomly generated profiles in an effort to filter out the impact of randomness on the performance. One can notice that, when neither the dynamics parameters nor the tracking setpoints are learnt, the resulting controller converges to a solid performance. Adding $\tilde{a}$ to the set of learnable parameters seems to boost the performance even further, which can be attributed to the fact that, among the various parameters, $\tilde{a}$ is the one the METANET model is most sensitive to. Further testing indicates that learning also $\tilde{\rho}_{\text{crit}}$ achieves an additional performance improvement, while learning $\tilde{v}_{\text{free}}$ does not help. The last empirical finding is that making the tracking setpoints learnable, despite increasing the number of degrees of freedom of the parametrisation, does not bring any improvement.

Evolution of parametrisation

Interested readers can find in Fig. 10 the evolution, during the whole learning process, of the whole parametrisation θ of the MPC scheme (31), as detailed in Section III-B and simulated in Section IV.

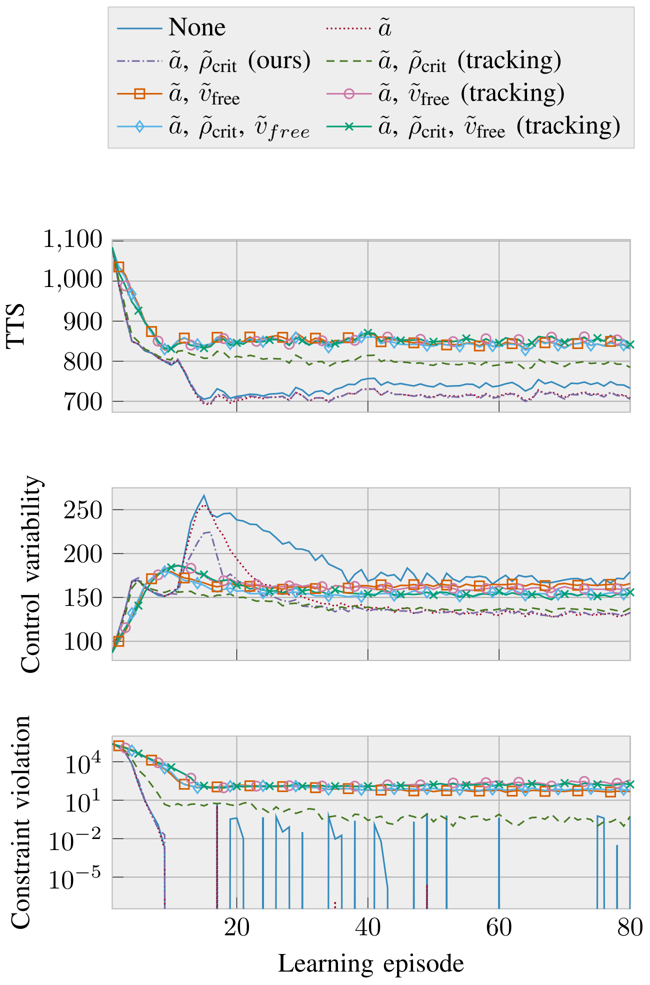

Fig. 9. Evolution of the RL cost contributions for various different parametrisations θ. To avoid visual clutter, only the average of the 15 simulation runs is shown for each parametrisation. Moreover, constraint violations are reported in a logarithmic scale to better highlight discrepancies among the different parametrisations.

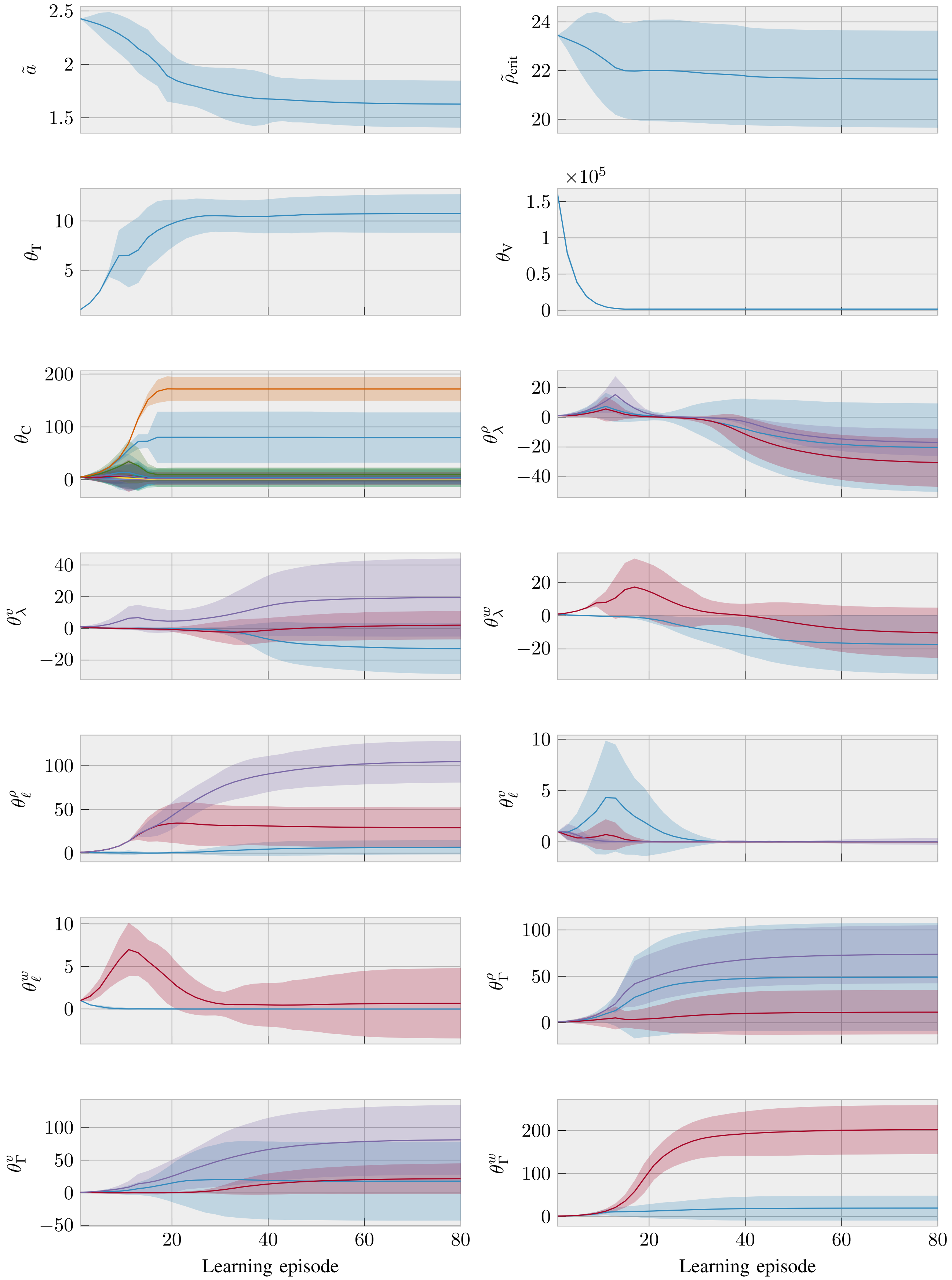

Fig. 10. Average evolution of the parametrisation θ during the learning process across the 15 simulation runs