Delft University of Technology

Delft Center for Systems and Control

Technical report 25-006

Reinforcement Learning With Model Predictive Control for Highway Ramp Metering*

F. Airaldi, B. De Schutter, and A. Dabiri

To cite this work, please refer to the published version:

F. Airaldi, B. De Schutter, and A. Dabiri, "Reinforcement learning with model predictive control for highway ramp metering," *IEEE Transactions on Intelligent Transportation Systems*, vol. 26, no. 5, pp. 5988–6004, May 2025. doi:10.1109/TITS.2025.3549227

Delft Center for Systems and Control
Delft University of Technology
Mekelweg 2, 2628 CD Delft
The Netherlands
phone: +31-15-278.24.73 (secretary)
URL: https://www.dcsc.tudelft.nl

* This report can also be downloaded via https://dspub.eu/25-006

Reinforcement Learning with Model Predictive Control for Highway Ramp Metering

Filippo Airaldi, Bart De Schutter, *Fellow, IEEE*, 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 ramp metering control that embeds Reinforcement Learning (RL) techniques within the Model Predictive Control (MPC) 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 the constraint on the maximum number of vehicles in queue. An MPC-based RL approach, which leverages the MPC optimal problem as a function approximation for the RL algorithm, is proposed to learn to efficiently control an on-ramp and satisfy its constraints despite uncertainties in the system model and variable demands. Simulations are performed on a benchmark small-scale highway network to compare the proposed methodology against other state-of-the-art control approaches. 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 that is superior to the other controllers.**

***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].

Ramp metering (RM) control has been proven to be an effective tool in regulating the traffic flow entering the network at on-ramps [5]. It 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 historic traffic data [6]. More efficient strategies involve real-time measurements of the traffic conditions, enabling online computation of the timings in a traffic-responsive way. In these cases, the underlying algorithms range from simpler feedback strategies [7], [8], [9] to more complex architectures, such as Model Predictive Control (MPC) [10], [11], [12] and Reinforcement Learning (RL) [13], [14]. One of the primary research challenges in RM today is the design of strategies that can control the entering flow in a way that effectively balances congestion in the freeway and the queue length at the on-ramp, two often conflicting goals [15], [16]. Most strategies deployed on actual ramps are either feedback-based local policies, which are inherently myopic, or model-based policies and, as such, are vulnerable to model inaccuracies and thus require precise identification of traffic parameters, as well as an attentive assessment of their efficacy [17]. Conversely, data-driven approaches, which are one of the main focuses of current research, typically necessitate extensive datasets of real or synthetic traffic data for training.

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 [18]. Its versatility further allows for alternative formulations that can systematically handle disturbances affecting the system, e.g., in a robust [19] or a stochastic [20] manner. 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 settings where constraint satisfaction is critical. 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.

Nonetheless, it is well known that the performance of such predictive controllers is heavily bound to the quality of the prediction model, which calls 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., synthesise 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 advance-

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). The Associate Editor for this paper was XX. (*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).

ments 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 [21]. 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. Early RL focused on tabular approaches that enjoy simplicity, some degree of interpretability, and convergence guarantees, but they are only applicable to small and simple environments [22], and are often limited in handling larger state spaces, exhibit slow convergence, and struggle to generalised. As the field progressed, researchers recognised the potential of function approximators and, in particular, of Neural Networks (NNs) [23], leading to the development of Deep RL (DRL). However, while DRL has demonstrated great efficacy [24], its NN-based policies generally lack interpretability, and the integration of prior knowledge (e.g., from domain experts) is still difficult. Thus, providing formal guarantees on properties relevant to a controller, such as stability and constraint satisfaction, in the context of NN-enhanced control architectures is one of today's challenges in the field. Additionally, existing model-free approaches are usually plagued by sample inefficiency and poor real-world performance, since they are often trained offline in approximate simulators and require large amounts of data for meaningful converge [25].

At the same time, on the other side of the coin, the control community is also dedicating 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 robust control policies with guarantees on constraint satisfaction. Notably, learning-based MPC methodologies [28] have been devised to, e.g., leverage Gaussian Process regression to learn unmodelled nonlinear dynamics in the prediction model via state transition observations [29], or use Bayesian Optimisation to automatically tune and optimise the controller hyperparameters [30]. In the literature, MPC has been successfully combined also with RL, e.g., as a safety filter for the learning-based agents (also called *shielding*) [26], [31], as a model reference and baseline control provider [32], [33], and to solve mixed-logical control problems more efficiently [34].

Recently, [35] 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. Unlike the more traditional pipeline, where a system identification phase (which selects model parameters that match the system's behaviour with observed data by, e.g., minimising the predicted mean-squared-error) is followed by the controller design (possibly, iteratively), this approach directly tunes the parametrisation to maximise performance, irrespective of the controller's predictive accuracy. Such a rationale is often called Identification for Control, and advocates that the best parametrisation for control shall not yield the best predictions, but rather the best performance on the true system [30], [35], [36]. In contrast to NN-based RL, the key advantages of the approach adopted in this paper are multiple:

- The MPC scheme at its core allows to easily integrate prior information on the system in the form of an expert-based, possibly imperfect, prediction model.
- MPC-based agents are in general more amenable to analysis and certification in terms of stability and constraint satisfaction, a benefit supported by the extensive body of literature on MPC [18]. Although challenging and beyond the scope of the proposed approach, it is possible to ensure that theoretical properties, such as stability, recursive feasibility and constraint guarantees, are preserved during the RL learning process [36], [37]. On the other hand, NNs are notoriously black-box, lack interpretability, and often provide no guarantees prior to convergence.
- MPC controllers 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.

Compared to traditional non-learning MPC formulations, the proposed approach enjoys the following benefits:

- The need for extensive open-loop data collection and system identification phase is eliminated, as the MPC parametrisation is automatically adjusted based on closed-loop data to improve performance, rather than prediction accuracy. This bypasses the issue of model inaccuracies and endows the scheme with online adaptation capabilities.
- By appropriately penalising constraint violations, the RL algorithm can guide the learning process to produce an MPC policy that satisfies system constraints.

All combined, these advantages can foster a safer training where the agent may enjoy a more stable and constraint-abiding learning process, opening up in the future the possibility of training directly in the real world [25], and contributing to the suitability of this framework for the RM control problem. In particular, the proposed MPC-RL method automatically learns from data a policy that optimally balances congestion and queue lengths, without requiring further manual tuning. This bypasses the issue of model inaccuracies in traffic predictions, which are often the primary concern in non-myopic control strategies. It also embeds adaptivity in the RM strategy in the face of stochasticities (e.g., demand forecasts or the behaviour of drivers can vary significantly across different cities or countries, meaning that system iden-

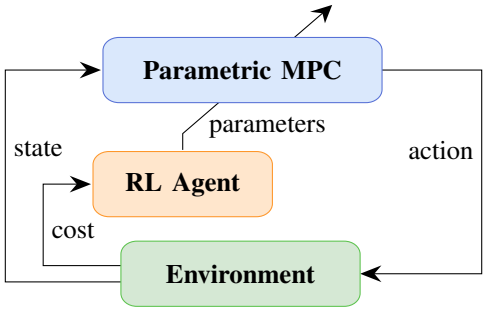

Fig. 1. Diagram of the MPC-based RL architecture

tification performed in one region may not generalise well to another). Besides, we stress again that improving the accuracy of the prediction model may not necessarily result in a better closed-loop performance. Hence, this framework allows for improvements to the control policy without focusing on nor needing an accurate prediction model or identification as a separate step in the design process. Additionally, the approach strengthens constraint satisfaction, which is crucial in real-world applications like RM control to avoid critical congested scenarios.

This paper contributes to the state of the art by proposing a novel approach, which embeds the MPC framework within an RL algorithm, to tackle the highway RM control problem. Specifically:

1) To the best of the authors' knowledge, an MPC-based RL method inspired by [35] is proposed for the first time to solve the RM control problem. The method combines a differentiable parametrised MPC scheme, which acts as policy provider and function approximation, with an online second-order Q-learning algorithm. The Q-learning algorithm adjusts the MPC parametrisation via gradient updates based only on observed closed-loop data, allowing to automatically learn a congestion-free and constraint-abiding control policy. As explained earlier, this approach overcomes the issue of model inaccuracies and uncertainties in the MPC and in the context of RM.
2) The proposed RM strategy is then implemented and tested on a benchmark highway stretch, where it is trained in an online fashion to simultaneously avoid bottlenecks and excessive queue lengths at the on-ramp, incorporating penalties for constraint violations directly into the MPC objective and the RL reward function. Simulation results underscore the effectiveness of the MPC-RL approach. Notably, starting from a poorly tuned controller with an imprecise prediction model, the proposed approach demonstrates quick learning in reducing congestion and satisfying the queue constraint. These results are further validated with a comparison with other state-of-the-art traffic controllers, both model-based and learning-based. Our method empirically outperforms them in performance and sample-efficiency.

The paper is organised as follows. Section II summarises relevant works in MPC, RL, and their combination in RM applications. Background on modelling and controlling traffic networks via MPC is provided in Section III. Section IV 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 V. Finally, Section VI concludes the paper with some final remarks and future directions.

II. Related Work

Several methods have been proposed to address the RM problem [3]. In this section, we report and discuss recent developments in model-based and model-free approaches, as well as the combination of the two, in comparison to our proposed methodology.

A. MPC Approaches for Ramp Metering Control

A well-studied approach to tackle RM is MPC. First formalised in [38], it was deployed for RM [39] and coordinated traffic control [10], [11], [12], and is still subject of research, e.g., [40], [41]. As explained in Section I, its predictive capabilities make MPC highly desirable, but at the same time leave it susceptible to model mismatches and uncertainties. When these disturbances are assumed to be modelled, one can resort to robust or stochastic formulations to mitigate their impact, as proposed in, e.g., [42]; however, such assumptions are in general hard to satisfy in practice, and the additional computational burden can be taxing. For these reasons, MPC techniques for traffic control that can adapt to and circumvent model mismatches and uncertainties are still a vibrant research area.

B. Model-free Approaches for Ramp Metering Control

Perhaps the most established RM approach is ALINEA [7], a local feedback strategy that seeks to maximise freeway flow in a merging area by keeping the bottleneck density near critical value. Many extensions have been proposed to enhance its basic formulation, such as PI-ALINEA, which deals with distant downstream bottlenecks [8], and FF-ALINEA, which anticipates the evolution of a bottleneck density [9]. However, these closed-loop controllers are fixed since they have no means to adapt their gains to changing conditions, and thus suffer from, e.g., uncertainties in traffic parameters and poor tuning. Ways have been devised to automatically account for it, e.g., via Kalman filtering to estimate the traffic parameters [43] or based on historic data [44]. Unfortunately, these methods are still plagued with a fundamentally myopic control strategy, and either assume that parameters evolve according to a known observable model, or that a huge amount of offline recorded data is available and is representative of both current and future traffic scenarios. Moreover, these feedback methods, alongside MPC, often need to be paired with a mechanism for traffic state estimation, enabling them to respond to real-time traffic behaviour [45].

More recently, RL has gained popularity thanks to its ability to automatically learn an optimal policy solely by interacting with the environment, addressing the issue of uncertain and unknown dynamics. Earlier works adopted tabular Q-learning for the RM problem [46], [47], [48]. However, with function approximations becoming more and more predominant,

especially NNs, DRL algorithms have emerged as a relevant alternative to model-based controllers. In [13], different policy gradient DRL algorithms are deployed in freeway control and compared. While results are promising, the issue of state constraints is not addressed, especially on the queue length, which should be contained to avoid spillback [3]. This is a fundamental issue in RL, since these algorithms can often be made aware of the presence of constraints only when violations occur and are appropriately penalised in the reward, whereas MPC can handle constraints explicitly in its formulation. In [14] it is proposed to use a combination of offline historic data and online synthetic data in an iterative way to avoid the RL model from being trapped in an inaccurate training environment. This work underscores the severe impact of high-quality data on the RL learning process, and raises the question of whether research efforts might be more effectively directed towards the development of online model-based learning algorithms that can learn directly from closed-loop interactions with the target environment while addressing constraints more explicitly. DRL has also facilitated the advancement of multi-agent solutions [49], which are especially useful in large-scale highway control problems. However, in addition to the aforementioned issues, multi-agent RL must also address the nonstationarity of its learning targets.

C. Combination of MPC and RL for Ramp Metering

The use of learning-based MPC schemes for traffic management has been investigated in the recent literature. For instance, [50] shows how recurrent NNs can be used to predict traffic behaviour, and [51] proposes to adopt such a network as prediction model in their MPC scheme.

Nonetheless, to the best of the authors' knowledge, the combination of MPC and RL in traffic control has received scarce attention, with [52], [53] the only relevant works. Therein, an MPC-DRL architecture is proposed that hierarchically combines the two controllers: the high-level MPC controller, which runs at a lower frequency, provides initial optimality while explicitly incorporating constraints, whereas the low-level DRL agent operates at a higher frequency and aims to compensate for model mismatches in the MPC. The MPC and RL control actions are then linearly combined together and fed to the freeway environment as a unique control input.

Given that the final control signal is a summation of an optimisation-based one and a learning-based one, it is anticipated that the resulting policy may not always be able to prevent constraint violations, particularly early in the training process. However, a benefit of [53] compared to the proposed methodology is the higher control frequency at which the DRL policy is run, granting quicker control response. This is especially advantageous in applications with fast dynamics, although traffic systems are typically slow, with sampling times up to tens of seconds. Note that the slower control frequency of our method is not inherently limiting as, with the improving hardware capabilities and faster solvers in recent years, increasing this frequency is feasible. The two methodologies also differ in computational complexity. While our approach is more demanding during the online training phase due to the need to solve the MPC problem twice per time step, it becomes more efficient at deployment, as it requires only one MPC calculation per time step [35]. In contrast, [53] runs both the MPC and DRL policies at every step, leading to higher computational costs at inference. Moreover, note that if training is performed offline, e.g., via an off-policy algorithm, then the additional computational burden of our method becomes irrelevant to online performance.

D. Comparison with Proposed Methodology

Compared to feedback strategies such as ALINEA, the MPC-based RL framework [35] we propose to use for RM leverages as its core an MPC controller, which, thanks to its predictive capabilities, can yield non-myopic policies. At the same time, its RL module allows to automatically adjust the MPC controller, and thus, it circumvents some of the challenges that conventional MPC typically struggles to address, i.e., model misspecifications and poor identification and tuning. This framework also boasts a higher degree of interpretability, better handling of constraints and integration of prior knowledge compared to other model-free RL methods. Furthermore, in contrast to [53], only one policy provider (the differentiable MPC controller itself) is present. For this reason, as remarked in Section I, this approach has the potential to result in fewer or no constraint violations at all depending on, e.g., the complexity of the underlying model and objective and the presence or lack of uncertainties [36], [37]. Likewise, at deployment, the proposed method relies only on the learnt MPC problem, whereas the method from [53] must additionally compute the output from the NN actor.

III. Background

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

Notation: The set of indices $\{1, \dots, N\} \subseteq \mathbb{N}$ is denoted as $\mathcal{N}$. Vector quantities are in bold. Inequalities on vectors are applied element-wise. To avoid confusion when other subscripts are present, dynamic quantities (e.g., ρ) 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., $\hat{y}_{i|k}$.

A. METANET Modelling Framework

Based on the traffic phenomena to investigate, different approaches to modelling a highway network exist [54]. 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 RM control. Proposed as a simulation tool in [55], [56], it made one of its first

appearances in combination with MPC in [39], [12], and has since been extended to cover several phenomena and traffic measures [57], [58].

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 the 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 characterised 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 characterised 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:

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

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

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

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

where T is the sampling time, L_j and λ_j are respectively the length of the segment and its number of lanes, (3) describes the traffic flow $q_j(k)$ of segment j at time step k, (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 [57]. Having defined the queue length as in (5), the on-ramp flow can be computed as

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

where C_o is the capacity of the origin, $\rho_{j_0}(k)$ is the density of the segment j_o that is connected to this ramp, and $\rho_{\max}$ is the maximum density segments can withstand (assumed here independent of index j_o, for sake of simplicity). Lastly, $\tilde{r}_o(k) \in [0,1]$ is the metering rate control action, which allows modulating the entering ramp flow to prevent, e.g., congestion and spillback. However, to avoid numerical issues in the MPC gradient-based solvers due to the $\min$ operator in (6), which can cause the gradient to be zero over a vast region of the state-action space, it is more efficient to control the ramp flow r_o directly. This requires imposing the additional terms in (6) as constraints on the control action, as shown in the next subsection.

Finally, the downstream boundary conditions are here considered to be affected by congestion, i.e., the highway segment j_i that ends in destination $i \in \mathcal{D}$ is subject to the (virtual) downstream density modelled in [11] as

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

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 disturbance.

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

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

$$\boldsymbol{u}_k = \big[r_1(k) \quad \dots \quad r_{|\mathcal{O}|}(k)\big]^\top, \quad (9)$$

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

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

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{d}_k), \quad (11)$$

where $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_d} \to \mathbb{R}^{n_x}$, with $n_x = 2|\mathcal{S}| + |\mathcal{O}|$, $n_u = |\mathcal{O}|$, $n_d = |\mathcal{O}| + |\mathcal{D}|$. The next section details how these dynamics can be used in designing a model-based controller.

B. MPC for Ramp Metering Control

In what follows, the classical MPC-based paradigm for the RM control problem is presented and summarised.

MPC [18] 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 RM control problems.

We define the MPC prediction horizon N_p and the control horizon $N_\mathrm{c} \leq N_\mathrm{p}/M$, where M is a natural number to distinguish the process timescale T from the controller timescale $T_\mathrm{c} = T/M$. 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 $i_\mathrm{c}(i) = \min\{\lfloor i/M \rfloor, N_\mathrm{c} - 1\}$.[1] Additionally, we define a constrained RM control scenario by imposing at time step k, for the on-ramps in the network, the constraint

$$w_o(k) \leq w_\mathrm{max}, \quad \forall o \in \mathcal{O}, \tag{12}$$

where w_max represents the maximum queue length the ramps should experience. For the sake of simplicity, it is assumed here that w_max is independent of the index o, and that all queue lengths are subject to this constraint, though it could be readily applied to only a subset of them. As mentioned in Section II, this is usually done to prevent spillback, and similarly constrained scenarios can be found in, e.g., [12]. This constraint is not to be considered hard, but we favour control strategies that are able to satisfy it, if not always, at least for most time steps.

Altogether, given the current state x_k of the network, the MPC controller is given by

$$\min_{\hat{\boldsymbol{X}}_k, \hat{\boldsymbol{U}}_k, \boldsymbol{\Sigma}} \quad \sum_{i=0}^{N_\mathrm{p}} L_\mathrm{T}(\hat{\boldsymbol{x}}_{i|k}) + \xi \sum_{i=0}^{N_\mathrm{c}-1} L_\mathrm{V}(\hat{\boldsymbol{u}}_{i|k}) + \sum_{i=0}^{N_\mathrm{p}} \boldsymbol{\zeta}_i^\top \boldsymbol{\sigma}_i \tag{13a}$$

$$\text{s.t.} \quad \hat{\boldsymbol{x}}_{0|k} = \boldsymbol{x}_k, \tag{13b}$$

$$\hat{\boldsymbol{x}}_{i+1|k} = f(\hat{\boldsymbol{x}}_{i|k}, \hat{\boldsymbol{u}}_{i_\mathrm{c}(i)|k}, \boldsymbol{d}_k), \; i = 0, \ldots, N_\mathrm{p} - 1, \tag{13c}$$

$$h(\hat{\boldsymbol{x}}_{i|k}, \hat{\boldsymbol{u}}_{i_\mathrm{c}(i)|k}) \leq 0, \qquad i = 0, \ldots, N_\mathrm{p}, \tag{13d}$$

$$g(\hat{\boldsymbol{x}}_{i|k}, \boldsymbol{\sigma}_i) \leq 0, \qquad i = 0, \ldots, N_\mathrm{p}, \tag{13e}$$

$$\boldsymbol{\sigma}_i \geq 0, \qquad i = 0, \ldots, N_\mathrm{p}, \tag{13f}$$

where the optimisation variables are the actions and states along the MPC control and prediction horizons, and the slack variables (whose purpose is explained later in the section), respectively in order

$$\hat{\boldsymbol{X}}_k = \begin{bmatrix} \hat{\boldsymbol{x}}_{0|k}^\top & \ldots & \hat{\boldsymbol{x}}_{N_\mathrm{p}|k}^\top \end{bmatrix}^\top \in \mathbb{R}^{n_x(N_\mathrm{p}+1)}, \tag{14}$$

$$\hat{\boldsymbol{U}}_k = \begin{bmatrix} \hat{\boldsymbol{u}}_{0|k}^\top & \ldots & \hat{\boldsymbol{u}}_{N_\mathrm{c}-1|k}^\top \end{bmatrix}^\top \in \mathbb{R}^{n_u N_\mathrm{c}}, \tag{15}$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\sigma}_0^\top & \ldots & \boldsymbol{\sigma}_{N_\mathrm{p}}^\top \end{bmatrix}^\top \in \mathbb{R}^{|\mathcal{O}|(N_\mathrm{p}+1)}, \tag{16}$$

where $\hat{\boldsymbol{X}}_k$ and $\hat{\boldsymbol{U}}_k$ are defined in analogy to (8) and (9) respectively, and $\boldsymbol{\sigma}_i$ collects the slack variables at the predicted step i corresponding to all on-ramps, i.e., $\boldsymbol{\sigma}_i = \begin{bmatrix} \sigma_i^1 & \ldots & \sigma_i^{|\mathcal{O}|} \end{bmatrix}^\top$. As aforementioned, once this optimisation problem is solved, we apply the optimal control action $\hat{\boldsymbol{u}}_{0|k}^\star$ from time step k to $k + M - 1$, as per the receding horizon approach.

The objective (13a) comprises three terms. The first is the total time spent (TTS), a metric frequently used to quantify efficiency of traffic control, defined as $L_\mathrm{T} : \mathbb{R}^{n_x} \to \mathbb{R}$

$$L_\mathrm{T}(\boldsymbol{x}_k) \coloneqq T \left(\sum_{j \in \mathcal{S}} L_j \lambda_j \rho_j(k) + \sum_{o \in \mathcal{O}} w_o(k) \right). \tag{17}$$

The second term of (13a) is a cost proportional to the variability of the inputs along the control horizon, where $L_\mathrm{V} : \mathbb{R}^{n_u} \to \mathbb{R}$ is

$$L_\mathrm{V}(\boldsymbol{u}_k) \coloneqq \sum_{o \in \mathcal{O}} \left(\frac{r_o(k) - r_o(k-1)}{C_o} \right)^2. \tag{18}$$

Lastly, the third term in (13a) acts as a penalty for the slack variables. The nonnegative weights $\xi \in \mathbb{R}$ and $\boldsymbol{\zeta}_i \in \mathbb{R}^{|\mathcal{O}|}$, $i = 0, \ldots, N_\mathrm{p}$, govern the trade-off between these three terms and are selected by the designer so as to encode the desired behaviour of the controller [10]. Section IV will show how these parameters (under the new symbols θ_V and $\boldsymbol{\Theta}_\mathrm{C}$), together with others, can be adjusted automatically via RL instead of having to be specified manually.

In (13d) and (13e), $h : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{4|\mathcal{O}|}$ and $g : \mathbb{R}^{n_x} \times \mathbb{R}^{|\mathcal{O}|} \to \mathbb{R}^{|\mathcal{O}|}$ correspond to all the inequality constraints to be imposed in the MPC problem. In particular, they are defined as

$$h(\boldsymbol{x}_i, \boldsymbol{u}_{i_\mathrm{c}}) \coloneqq \begin{bmatrix} h_1(\boldsymbol{x}_i, \boldsymbol{u}_{i_\mathrm{c}})^\top & \ldots & h_{|\mathcal{O}|}(\boldsymbol{x}_i, \boldsymbol{u}_{i_\mathrm{c}})^\top \end{bmatrix}^\top, \tag{19}$$

$$g(\boldsymbol{x}_i, \boldsymbol{\sigma}_i) \coloneqq \begin{bmatrix} g_1(\boldsymbol{x}_i, \boldsymbol{\sigma}_i) & \ldots & g_{|\mathcal{O}|}(\boldsymbol{x}_i, \boldsymbol{\sigma}_i) \end{bmatrix}^\top, \tag{20}$$

where

$$h_o(\boldsymbol{x}_i, \boldsymbol{u}_{i_\mathrm{c}}) \coloneqq \begin{bmatrix} -r_o(i_\mathrm{c}) \\ r_o(i_\mathrm{c}) - C_o \\ r_o(i_\mathrm{c}) - d_o^w(i) - \frac{w_o(i)}{T} \\ r_o(i_\mathrm{c}) - C_o \left(\frac{\rho_\mathrm{max} - \rho_{j_o}(i)}{\rho_\mathrm{max} - \rho_\mathrm{crit}} \right) \end{bmatrix}, \quad \forall o \in \mathcal{O}, \tag{21}$$

$$g_o(\boldsymbol{x}_i, \boldsymbol{\sigma}_i) \coloneqq w_o(i) - w_\mathrm{max} - \sigma_i^o, \qquad \forall o \in \mathcal{O}, \tag{22}$$

and we have dropped the dependency on i in i_c for readability. The constraints $h_o(\boldsymbol{x}_i, \boldsymbol{u}_{i_\mathrm{c}}) \leq 0$ emulate the behaviour of (6) and prevent the control action from assuming invalid values. The constraints $g_o(\boldsymbol{x}_i, \boldsymbol{\sigma}_i) \leq 0$ instead incorporate the queue

[1] This definition of $i_\mathrm{c}(i)$ 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.

requirements (12) into the MPC controller as soft constraints, where $\boldsymbol{\sigma}_k \geq 0$ is a slack variable whose purpose is to relax the constraint in order to avoid infeasibility issues, and so it must be appropriately penalised in the objective (13a). We opt for a soft constraint because 1) it is assumed that small violations to this constraint, although undesired, are tolerable, and 2) we need a mechanism to both relax constraints during the learning process and penalise violations in order to inform the agent about policies that violate constraints and ones that do not.

Note that in (13b) we assume that the current process state $\boldsymbol{x}_k$ is fully measurable. If this is not the case, an observer is required to provide an estimate of it. Likewise, in (13c) we assume the external inputs $\boldsymbol{d}_k$ to be fully known. If this is not the case, a forecast of the evolution of these quantities along the MPC horizon is required.

C. 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 [21]. Given the continuous state $\boldsymbol{s}$ and action $\boldsymbol{a}$, a discrete-time Markov Decision Process with state transitions $\boldsymbol{s} \xrightarrow{\boldsymbol{a}} \boldsymbol{s}_+$ and underlying conditional probability density

$$\mathbb{P}[\boldsymbol{s}_+ \mid \boldsymbol{s}, \boldsymbol{a}] : \mathbb{R}^{n_s} \times \mathbb{R}^{n_s} \times \mathbb{R}^{n_a} \to [0, 1] \tag{23}$$

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

$$J(\pi_{\boldsymbol{\theta}}) \coloneqq \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k L\big(\boldsymbol{s}_k, \pi_{\boldsymbol{\theta}}(\boldsymbol{s}_k)\big)\right], \tag{24}$$

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 $\pi^\star_{\boldsymbol{\theta}}$ as

$$\pi^\star_{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \quad J(\pi_{\boldsymbol{\theta}}). \tag{25}$$

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 NN and, as in this paper, MPC) have been employed as a powerful alternative for tackling problem (25) [23]. Depending on the algorithm, these allow formulating the approximations $V_{\boldsymbol{\theta}}$, $Q_{\boldsymbol{\theta}}$, and $\pi_{\boldsymbol{\theta}}$, which are then used to solve (25) either directly or indirectly via iterative gradient updates of the parametrisation

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \sum_{i=1}^{m} \psi\left(\boldsymbol{s}_i, \boldsymbol{a}_i, \boldsymbol{s}_{i+1}, \boldsymbol{\theta}\right), \tag{26}$$

where $\alpha \in \mathbb{R}_+$ is the learning rate, m is 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 $\pi_{\boldsymbol{\theta}}$ of the optimal policy itself. For instance, policy gradient methods attempt to solve (25) directly by moving along the gradient of the policy, i.e.,

$$\mathbb{E}[\psi\left(\boldsymbol{s}_i, \boldsymbol{a}_i, \boldsymbol{s}_{i+1}, \boldsymbol{\theta}\right)] = J(\pi_{\boldsymbol{\theta}}). \tag{27}$$

On the other hand, indirect (or value-based) methods opt to indirectly derive the optimal policy by estimating and learning the value functions $V_{\boldsymbol{\theta}}, Q_{\boldsymbol{\theta}}$ 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

$$\min_{\boldsymbol{\theta}} \quad \mathbb{E}\Big[\|Q_\star(s, a) - Q_{\boldsymbol{\theta}}(s, a)\|^2\Big], \tag{28}$$

with the hope of indirectly recovering the optimal policy from it. In its first-order recursive (i.e., with $m = 1$) formulation, $\boldsymbol{\theta}$ is updated via (26) with $\psi(\boldsymbol{s}_i, \boldsymbol{a}_i, \boldsymbol{s}_{i+1}, \boldsymbol{\theta}) = \delta_i^2$, where δ_i is the well-known temporal difference (TD) error:

$$\delta_i = L(\boldsymbol{s}_i, \boldsymbol{a}_i) + \gamma V_{\boldsymbol{\theta}}(\boldsymbol{s}_{i+1}) - Q_{\boldsymbol{\theta}}(\boldsymbol{s}_i, \boldsymbol{a}_i). \tag{29}$$

IV. MPC-based RL for Ramp Metering

As discussed in Section I, the closed-loop performance of the MPC controller (13) 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, thus impacting the controller performance and its ability to avoid constraint violations. 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 [59]. 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 RM task, we first need to define a proper stage cost $L(x_k, u_k)$ to quantify the performance of a policy in controlling the metering, as per (24). Also referred to as reward in literature (i.e., the negative of the cost), it defines the overall return that the RL algorithm must minimise and it encodes the objective that the designer wants to see minimised by the MPC control policy. Consider

$$L(\boldsymbol{x}_k, \boldsymbol{u}_k) \coloneqq c_{\mathrm{T}} L_{\mathrm{T}}(\boldsymbol{x}_k) + c_{\mathrm{V}} L_{\mathrm{V}}(\boldsymbol{u}_k) + c_{\mathrm{C}} L_{\mathrm{C}}(\boldsymbol{x}_k), \tag{30}$$

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

$$L_{\mathrm{C}}(\boldsymbol{x}_k) \coloneqq \sum_{o \in \mathcal{O}} \max\left\{0, w_o(k) - w_{\max}\right\}. \tag{31}$$

The first term in (30) 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 RM 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 acts as the RL counterpart to the MPC soft constraint (22) and penalisation of the corresponding slack variables in the MPC objective(13a): it penalises violations of the queue constraint so that the agent is encouraged to come up with policies that are able to satisfy this constraint. Scalars c_{T}, c_{V}, and c_{C} are weights that balance

each term differently, and it is the designer's task to choose these in such a way to properly embed the learning objective in the RL stage cost.[2] 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

While the RL stage cost (30) defines and guides the learning process, a function approximation is needed to estimate the underlying optimal value and policy functions. A recurrent candidate for this rule in the literature is NNs. However, in Section I it was discussed how [35] 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 (13). 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 (24).

Consider the MPC controller with parametrisation $\boldsymbol{\theta}$

$$\begin{aligned}
\min_{\hat{\boldsymbol{X}}_k, \hat{\boldsymbol{U}}_k, \boldsymbol{\Sigma}} \quad & \sum_{i=0}^{N_\mathrm{p}} \gamma^i \Big(\theta_\mathrm{T} L_\mathrm{T}(\hat{\boldsymbol{x}}_{i|k}) + \boldsymbol{\theta}_\mathrm{C}^{i\,\top} \boldsymbol{\sigma}_i \Big) \\
& + \theta_\mathrm{V} \sum_{i=0}^{N_\mathrm{c}-1} \gamma^{iM} L_\mathrm{V}(\hat{\boldsymbol{u}}_{i|k}) + \lambda_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{0|k}) \\
& + \sum_{i=1}^{N_\mathrm{p}-1} \gamma^i \ell_{\boldsymbol{\theta}}(\boldsymbol{x}_k) + \gamma^{N_\mathrm{p}} \Gamma_{\boldsymbol{\theta}}(\boldsymbol{x}_{N_\mathrm{p}}) && \text{(32a)} \\
\text{s.t.} \quad & \hat{\boldsymbol{x}}_{0|k} = \boldsymbol{x}_k && \text{(32b)} \\
& \hat{\boldsymbol{x}}_{i+1|k} = f_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{i|k}, \hat{\boldsymbol{u}}_{i_\mathrm{c}(i)|k}, \boldsymbol{d}_k), i = 0, \dots, N_\mathrm{p} - 1, && \text{(32c)} \\
& h_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}_{i|k}, \hat{\boldsymbol{u}}_{i_\mathrm{c}(i)|k}) \leq 0, \qquad i = 0, \dots, N_\mathrm{p}, && \text{(32d)} \\
& g(\hat{\boldsymbol{x}}_{i|k}, \boldsymbol{\sigma}_i) \leq 0, \qquad i = 0, \dots, N_\mathrm{p}, && \text{(32e)} \\
& \boldsymbol{\sigma}_i \geq 0, \qquad i = 0, \dots, N_\mathrm{p}. && \text{(32f)}
\end{aligned}$$

with $\boldsymbol{\Theta}_\mathrm{C} = \begin{bmatrix} \boldsymbol{\theta}_\mathrm{C}^{0\,\top} & \dots & \boldsymbol{\theta}_\mathrm{C}^{N_\mathrm{p}\,\top} \end{bmatrix}^\top \in \mathbb{R}^{|\mathcal{O}|(N_\mathrm{p}+1)}$. Because this controller acts as function approximation and must be able to accurately predict the realisations of the RL stage cost along the prediction horizon, the objective (32a) is devised so as to mimic (30). Furthermore, note the additional parameterised terms $\lambda_{\boldsymbol{\theta}}, \ell_{\boldsymbol{\theta}}, \Gamma_{\boldsymbol{\theta}} : \mathbb{R}^{n_x} \to \mathbb{R}$. These are the learnable initial, stage, and terminal costs respectively, and serve the purpose of enriching the parametrisation and facilitating generalisation across the state-action space. The initial cost is a linear combination of the state

$$\lambda_{\boldsymbol{\theta}}(\boldsymbol{x}_k) \coloneqq \sum_{j \in \mathcal{S}} \left(\theta^\rho_{\lambda,j} \frac{\rho_j(k)}{\rho_\mathrm{max}} + \theta^v_{\lambda,j} \frac{v_j(k)}{v_\mathrm{max}} \right) + \sum_{o \in \mathcal{O}} \theta^w_{\lambda,o} \frac{w_o(k)}{w_\mathrm{max}}, \tag{33}$$

[2]In our work, the values for c_T, c_V, and c_C were manually adjusted to achieve a reasonable learning process.

where ρ_max, v_max, w_max are here used as fixed, non-learnable normalisation constants in order to improve the stability of the learning process. As explained in [35], 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 chosen to be qua quadratic

$$\begin{aligned}
\ell_{\boldsymbol{\theta}}(\boldsymbol{x}_k) \coloneqq & \sum_{j \in \mathcal{S}} \theta^\rho_{\ell,j} \left(\frac{\rho_j(k) - \rho_\mathrm{sp}}{\rho_\mathrm{max}} \right)^2 \\
& + \sum_{j \in \mathcal{S}} \theta^v_{\ell,j} \left(\frac{v_j(k) - v_\mathrm{sp}}{v_\mathrm{max}} \right)^2 \\
& + \sum_{o \in \mathcal{O}} \theta^w_{\ell,o} \left(\frac{w_o(k)}{w_\mathrm{max}} \right)^2, && \text{(34)} \\
\Gamma_{\boldsymbol{\theta}}(\boldsymbol{x}_k) \coloneqq & \sum_{j \in \mathcal{S}} \theta^\rho_{\Gamma,j} \left(\frac{\rho_j(k) - \rho_\mathrm{sp}}{\rho_\mathrm{max}} \right)^2 \\
& + \sum_{j \in \mathcal{S}} \theta^v_{\Gamma,j} \left(\frac{v_j(k) - v_\mathrm{sp}}{v_\mathrm{max}} \right)^2 \\
& + \sum_{o \in \mathcal{O}} \theta^w_{\Gamma,o} \left(\frac{w_o(k)}{w_\mathrm{max}} \right)^2, && \text{(35)}
\end{aligned}$$

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 $f_{\boldsymbol{\theta}}$ in (32c), we allow the RL algorithm to adjust two fundamental parameters in the dynamics, i.e., $\tilde{\rho}_\mathrm{crit}$ 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 (24) 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 $h_{\boldsymbol{\theta}}$ (32d) get parameterised, since $\tilde{\rho}_\mathrm{crit}$ appears in it; see (13d) and (21).

Remark 1: In general, a parametrisation that makes use of quadratic terms with fixed setpoints, as in (34) and (35), can lead to suboptimal 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 [35]. For this reason, in this work, we opt for the aforementioned adjustable quadratic cost terms. That being 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.

Remark 2: The MPC parameters $\theta_\mathrm{T}, \theta_\mathrm{V}, \boldsymbol{\Theta}_\mathrm{C}$ are disjoint from the RL stage cost weights $c_\mathrm{T}, c_\mathrm{V}, c_\mathrm{C}$ of (30). While they encode similar notions, there is in general no reason to expect the values of the former to converge to the values of the latter during learning. Rather, they will converge to those values that maximise the controller's closed-loop performance.

TABLE I
PARAMETRISATION $\boldsymbol{\theta}$ OF THE MPC FUNCTION APPROXIMATION (32)

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	$\mathbb{R}$
$\boldsymbol{\Theta}_{\text{C}}$	cost - slack weights	$\mathbb{R}^{\lvert\mathcal{O}\rvert(N_{\text{p}}+1)}$
$\boldsymbol{\theta}^{\{\rho,v\}}_{\{\lambda,\ell,\Gamma\}}$	{init., stage, term.} cost - $\{\rho,\, v\}$ weights	$\mathbb{R}^{\lvert\mathcal{S}\rvert}$
$\boldsymbol{\theta}^{w}_{\{\lambda,\ell,\Gamma\}}$	{init., stage, term.} cost - w weights	$\mathbb{R}^{\lvert\mathcal{O}\rvert}$

The whole parametrisation is

$$\boldsymbol{\theta} = \begin{bmatrix} \tilde{\rho}_{\text{crit}} & \tilde{a} & \theta_{\{\text{T,V}\}} & \boldsymbol{\Theta}_{\text{C}}^{\top} & \boldsymbol{\theta}^{\{\rho,v,w\}\,\top}_{\{\lambda,\ell,\Gamma\}} \end{bmatrix}^{\top}. \tag{36}$$

Table I shows a summary of the 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.

Scheme (32) yields the approximation $V_{\boldsymbol{\theta}} : \mathbb{R}^{n_x} \to \mathbb{R}$ of the value function as

$$\begin{aligned} V_{\boldsymbol{\theta}}(\boldsymbol{x}_k) = \min_{\hat{\boldsymbol{X}}_k, \hat{\boldsymbol{U}}_k, \boldsymbol{\Sigma}} \quad & \text{(32a)} \\ \text{s.t.} \quad & \text{(32b)} - \text{(32f)}. \end{aligned} \tag{37}$$

The value function (37) satisfies the fundamental equalities of the Bellman equations [35], so that

$$\begin{aligned} Q_{\boldsymbol{\theta}}(\boldsymbol{x}_k, \boldsymbol{u}_k) = \min_{\hat{\boldsymbol{X}}_k, \hat{\boldsymbol{U}}_k, \boldsymbol{\Sigma}} \quad & \text{(32a)} \\ \text{s.t.} \quad & \text{(32b)} - \text{(32f)}, \\ & \hat{\boldsymbol{u}}_{0|k} = \boldsymbol{u}_k. \end{aligned} \tag{38}$$

$$\pi_{\boldsymbol{\theta}}(\boldsymbol{x}_k) = \arg\min_{\boldsymbol{u}} \quad Q_{\boldsymbol{\theta}}(\boldsymbol{x}_k, \boldsymbol{u}). \tag{39}$$

In practice, policy (39) is found as the first optimal action $\hat{\boldsymbol{u}}^{\star}_{0|k}$ from the $\arg\min$ of (37). The goal of this parametrisation is to give the MPC scheme (32) enough degrees of freedom to sufficiently adapt to the RL task at hand. The cardinal point is that $\boldsymbol{\theta}$ must be rich enough to allow the scheme to capture the optimal policy $\pi^{\star}$ [35]. Of course, the choice of parametrisation is not unique, and other solutions may be viable: similarly to NNs, where the topology of the network and the number of neurons and activation functions to used in each layer are arbitrary, here also different choices are possible. In the context of RM, the inclusion of some parameters in $\boldsymbol{\theta}$ comes from the knowledge of the system and its control, e.g., the system's high sensitivity to the dynamics parameters a, ρ_{crit} and the cost weights θ_{T}, θ_{V}, $\boldsymbol{\theta}_{\text{C}}$. Other elements of the parametrisation, such as the additional quadratic terms, are instead dictated by the framework [35] to ensure a rich approximation scheme. However, unlike NNs, the benefit of this model-based approach is that knowledge about the system can be incorporated in the parametrisation rather trivially. Further discussion on different parametrisations and their impact on the learning outcome can be found in the appendix.

Beyond a proper selection of $\boldsymbol{\theta}$, other factors influence the suboptimality of the MPC-based RL solution. During training, the RL algorithm can only process a finite amount of data, whereas theoretical optimality is achieved only as this amount approaches infinity [21]. The MPC approximation scheme is also inherently suboptimal due to its nonlinearity, converging to global optimality only in the asymptotic limit, though the issue can be alleviated via, e.g., multi-start [18].

C. Second-Order LSTD Q-learning

In order to adjust the parametrisation $\boldsymbol{\theta}$ of (32), a second-order least-squares temporal difference (LSTD) Q-learning algorithm [60] is employed to find the policy $\pi_{\boldsymbol{\theta}}$ 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 $Q_{\boldsymbol{\theta}}$ to the observed data, with the aim of recovering via this approximation the unknown optimal $Q_{\star}$ and, indirectly from that, the optimal policy $\pi_{\star}$. It has also shown very promising results in the context of MPC [61]. The second-order Newton's method, coupled with an experience replay buffer of the past observed transitions [62] and a reformulation of the Q-fitting problem as a least-squares, ensures faster convergence and higher sample efficiency compared to traditional first-order methods. For more details, see [61].

More concretely, given the approximation $Q_{\boldsymbol{\theta}}$ (irrespective of its nature, e.g., MPC, NN, etc.), Q-learning solves the least-squares Bellman residual problem (28) via the second-order gradient update

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \boldsymbol{H}^{-1} \boldsymbol{p}, \tag{40}$$

where $\alpha \geq 0$ is the learning rate, and the gradient $\boldsymbol{p}$ and Hessian $\boldsymbol{H}$ are found as

$$\boldsymbol{p} = -\sum_{i=1}^{m} \delta_i \nabla_{\boldsymbol{\theta}} Q_{\boldsymbol{\theta}}(\boldsymbol{s}_i, \boldsymbol{a}_i), \tag{41}$$

$$\boldsymbol{H} = \sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} Q_{\boldsymbol{\theta}}(\boldsymbol{s}_i, \boldsymbol{a}_i) \nabla_{\boldsymbol{\theta}} Q_{\boldsymbol{\theta}}^{\top}(\boldsymbol{s}_i, \boldsymbol{a}_i) - \delta_i \nabla^2_{\boldsymbol{\theta}} Q_{\boldsymbol{\theta}}(\boldsymbol{s}_i, \boldsymbol{a}_i), \tag{42}$$

with m denoting the experience sample batch size and δ_i the TD error (29). The update rule (40) requires differentiation of the MPC scheme (38) with respect to $\boldsymbol{\theta}$ to find $\nabla_{\boldsymbol{\theta}} Q_{\boldsymbol{\theta}}$ and $\nabla^2_{\boldsymbol{\theta}} Q_{\boldsymbol{\theta}}$. A nonlinear programming sensitivity analysis demonstrates how these quantities can be computed through differentiation of the Lagrangian of (38) [63]. Prior to any update, the computed Hessian matrix is modified to be positive-definite by the addition of a multiple of the identity matrix [64]. Furthermore, to impose lower and upper bounds on the parameters (as reported in Table II), (40) is cast as the following optimisation problem

$$\begin{aligned} \Delta\boldsymbol{\theta}^{\star} = \arg\min_{\Delta\boldsymbol{\theta}} \quad & \frac{1}{2} \Delta\boldsymbol{\theta}^{\top} \boldsymbol{H} \Delta\boldsymbol{\theta} + \alpha \boldsymbol{p}^{\top} \Delta\boldsymbol{\theta} \\ \text{s.t.} \quad & \boldsymbol{\theta}_{\text{lb}} \leq \boldsymbol{\theta} + \Delta\boldsymbol{\theta} \leq \boldsymbol{\theta}_{\text{ub}}, \\ & \Delta\boldsymbol{\theta}_{\text{lb}} \leq \Delta\boldsymbol{\theta} \leq \Delta\boldsymbol{\theta}_{\text{ub}}. \end{aligned} \tag{43}$$

This formulation allows to limit the rate of change of each parameter with $\Delta\boldsymbol{\theta}_{\text{lb}}$ and $\Delta\boldsymbol{\theta}_{\text{ub}}$. The parametrisation is then

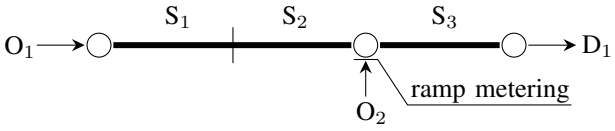

Fig. 2. Structure of the three-segment highway network

updated as $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}^\star$. Note that, compared to (32), (43) adds negligible computational complexity as it is a convex quadratic problem and is usually solved at a lower, batch-update frequency (as explained in Section V-A3).

Lastly, as in [36], exploratory behaviour is ensured during learning by adding to the MPC objective (32a) the perturbation term $\boldsymbol{q}^\top \boldsymbol{u}_0$, with $\boldsymbol{q}$ 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.

V. Numerical Case Study

To examine the performance of the proposed method for RM control, 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 taken from [11] and 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 characterised by its capacity C_1. 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 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 described in Section III-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 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 two-peak profiles are generated randomly to introduce a degree of variability in the task, aimed at enhancing the agent's generalisation to a wider variety of congestion scenarios. Nevertheless, generalisation in RL is challenging [65], and a performance drop is anticipated were the controller to face a new unforeseen scenario, e.g., with a different number of peaks. At the same time, our approach exhibits an inherent level of robustness: it incorporates an online learning process that can seamlessly integrate novel training data on the fly,

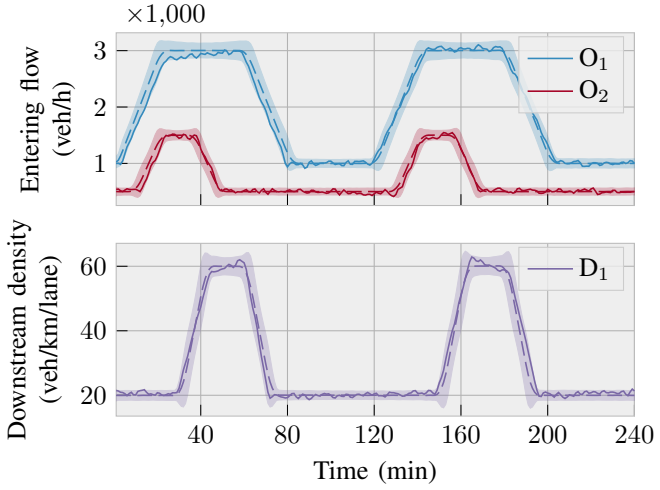

Fig. 3. External inputs affecting the network's dynamics, in the form of the demands at the origins (upper) and the congestion scenario at the destination (lower). The shaded areas represent the 2-standard deviation ranges from which the random demands are sampled, whereas the solid lines represent one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

and at its core is a model-based controller, which generally extrapolates better to new situations in comparison to its model-free counterparts.

2) Model Predictive Control: The controller (32) is deployed to control the on-ramp flow with the aim of avoiding congestion in the traffic network environment. In line with other similar benchmarks in the literature, e.g., [10], [11], [12], $M = 6$, 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 $N_\mathrm{p} = 24$ (which is in the order of the average travel time through the network), and the control horizon to $N_\mathrm{c} = 3$.

To showcase the ability of RL to automatically improve the performance of the MPC controller, the parametrisation $\boldsymbol{\theta}$ 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., $\tilde{\rho}_\mathrm{crit} = 0.7\rho_\mathrm{crit}$, $\tilde{a} = 1.3a$ (both learnable), and $1.3v_\mathrm{free}$ (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 parameter. The other non-learnable parameters in (32) are set to the same values as in the real process. The queue threshold on O_2 is set to $w_\mathrm{max} = 50$ veh. Lastly, the setpoints are set to $\rho_\mathrm{sp} = 0.7\rho_\mathrm{crit}$, $v_\mathrm{sp} = 1.3v_\mathrm{free}$, and the normalization coefficients to $\rho_\mathrm{max} = 180$ veh/km/lane, $w_\mathrm{max} = 50$ veh, and $v_\mathrm{max} = 1.3v_\mathrm{free}$.

3) Reinforcement Learning: The agent is trained in an episodic fashion, with each episode lasting $T_\mathrm{fin} = 4$ h, 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

TABLE II
INITIAL VALUES, BOUNDS AND DIMENSIONS OF THE PARAMETRISATION $\boldsymbol{\theta}$ OF (32)

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	$[10^{-3}, \infty)$	1/veh/h
θ_{V}	160000	$[10^{-3}, \infty)$	veh^2/h^2
$\boldsymbol{\Theta}_{\text{C}}$	5	$[10^{-3}, \infty)$	1/veh
$\boldsymbol{\theta}_{\lambda}^{\{\rho,v,w\}}$	1	$(-\infty, \infty)$	-
$\boldsymbol{\theta}_{\{\ell,\Gamma\}}^{\{\rho,v,w\}}$	1	$[10^{-6}, \infty)$	-

unreasonable accumulation of vehicles in queues from past episodes. With the same frequency, the parametrisation $\boldsymbol{\theta}$ 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. With the aforementioned update frequency, the proposed learning setup is off-policy, since θ remains fixed throughout each episode, ensuring stable data collection.

The stage cost function $L(\boldsymbol{x}_k, \boldsymbol{u}_k)$, with which the traffic network feeds a cost signal back to the agent according to (30), 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., $\Delta\boldsymbol{\theta}_{\text{ub}} = -\Delta\boldsymbol{\theta}_{\text{lb}} = 0.3\boldsymbol{\theta}$. 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 of this batch is dedicated to containing information from the latest two and a 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 IV-C by sampling the cost perturbation from a normal distribution in an ε-greedy fashion, i.e., $\boldsymbol{q} \sim \mathcal{N}(0, \sigma_q)$ with probability ε; otherwise, $\boldsymbol{q} = 0$. 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-parameters and non-learnable parameters of the traffic environment and the MPC controller, as found in [11], as well as of the RL algorithm.

4) Setup: The simulations were run on a 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 [66] and its interface to the IPOPT solver [67]. The source code and simulation results are open and available in the following repository: https://github.com/FilippoAiraldi/mpcrl-for-ramp-metering.

B. Comparison

Alongside the proposed MPC-based RL controller, we implement and simulate on the same setup three other policies. These include another learning-based approach based on DDPG, a DRL algorithm that has been proven effective in freeway control [13], [53], and two other non-learning solutions: the classical non-learning MPC formulation (13) [11], as well as the well-known local ramp metering PI-ALINEA [7], [8]. To promote a fair comparison, all of the tested controllers suffer from the same inexact knowledge of the traffic parameters.

The relevant hyper-parameters for DDPG are taken from [53]. The PI-ALINEA controller is equipped with a queue management strategy to take into account the queue length constraint [68], and its gains are fine-tuned to this specific task via Bayesian Optimisation.

C. Results

We evaluate the performance of all the aforementioned methods deployed on the traffic network environment and average the results over 15 simulations with different seeds to iron out randomness due to, e.g., exploration. Figures that are shown and discussed next report the average results; however, 95% confidence intervals are shown only for our proposed method to reduce visual clutter.

Fig. 4 shows, for each method, the total RL cost (30) incurred in each episode, i.e., $\sum_{k=1}^{T_{\text{in}}/T} L(\boldsymbol{x}_k, \boldsymbol{u}_k)$, subdivided in its three contributions, and how these evolve during learning. Due to the untuned objective weights and the 30% mismatches in the predictive model's parameters, the initial MPC-based RL controller is poorly performing and cannot reliably avoid congestion and constraint violations, as 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 h (or a 35% reduction). At the same time, as constraint violations of w_{max} in O_2 induce the most severe cost realisation by several orders of magnitude, one can see that the agent is even quicker in learning to avoid such a constraint-violating 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 policy 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 beneficial to the reduction of cost related to violations, it

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	$\Delta\boldsymbol{\theta}_{\{\text{lb,ub}\}}$	$\sigma_{\boldsymbol{q}}$	ε
	Value	5	1600	5	5 episodes	0.98	0.925	0.925	{-0.3θ,0.3θ}	0.025	0.5

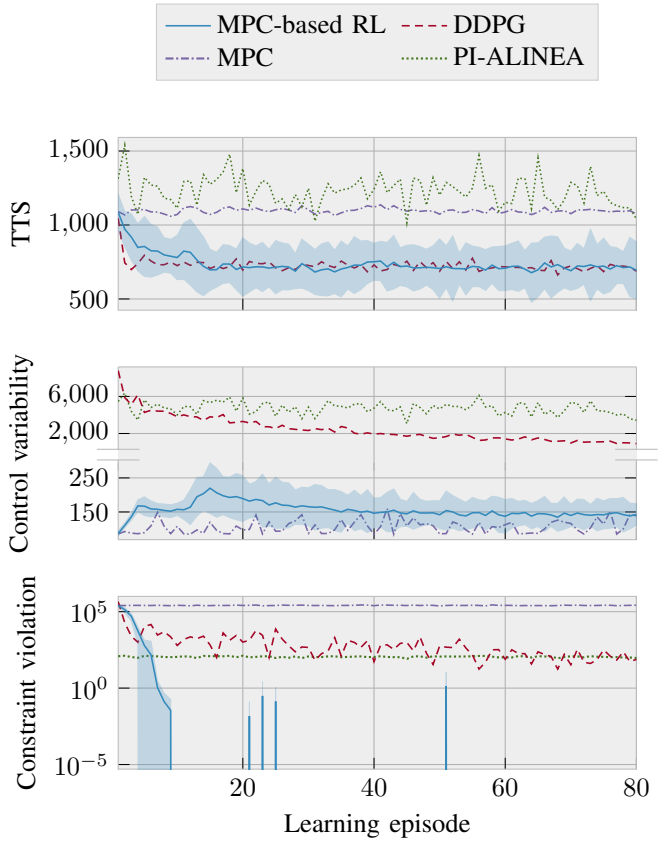

Fig. 4. Evolution of the three contributions to the RL cost (30) during the learning process, namely, from top to bottom, the Total-Time-Spent (TTS), variability of the control action, and violation of the maximum queue constraint on O_2

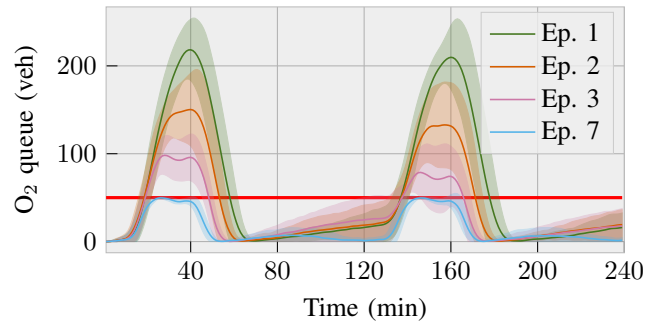

Fig. 5. On average, the policy $\pi_{\boldsymbol{\theta}}$ gets better at avoiding constraint violations as it learns (the red line represents the threshold $w_{\max}$ imposed on the queue on the on-ramp O_2)

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 for the MPC-based RL agent. 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 16$^{\text{th}}$ 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.

The results from the other methods appear to corroborate the usefulness of data-driven adaptability in such a context. The non-learning MPC formulation is stuck at the initial poor performance of its RL-enhanced counterpart, with high TTS and constraint violations. On the contrary, thanks to its queue management strategy, PI-ALINEA is able to curb violations, but at the expense of a much higher control variability (since the strategy frequently requires to switch control action abruptly) and higher TTS. The DDPG agent follows a similar trend to our method in quickly learning to reduce TTS. However, variability and violations are decreasing at a substantially lower rate. This is not unexpected, as the NN approximation of this agent has significantly more learnable parameters (namely, 293 380 vs. 53) and is thus less data-efficient and requires more episodes to establish convergence.

As discussed in Section I, an advantage of the proposed MPC-RL approach over DRL is that its learning process can be more readily inspected and explained via the evolution of its parametrisation $\boldsymbol{\theta}$, reported in Fig. 6. In particular

- the evolution of 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_o^1

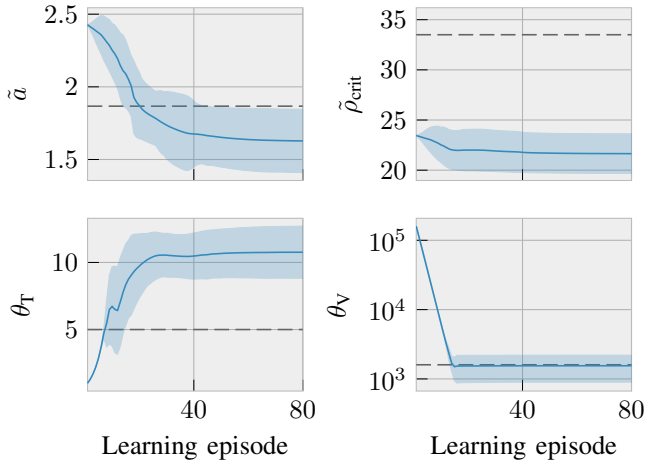

Fig. 6. Evolution of a subset of $\boldsymbol{\theta}$ during the learning process (the dashed grey lines represent, in the top plots, the true values of the parameters a and ρ_{crit}, and, in the bottom ones, the constants c_{T} and c_{V})

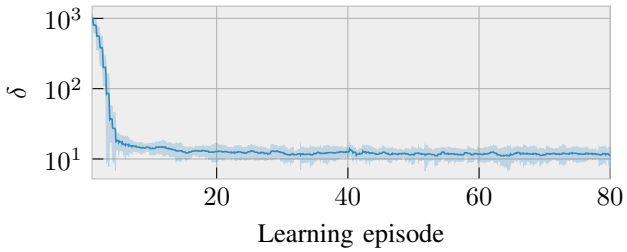

Fig. 7. Moving average of the TD error during the learning process (window length = 239, i.e., number of transitions per episode)

(21), 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 (32), with its parametrisation $\boldsymbol{\theta}$, is able to provide a reliable approximation $Q_{\boldsymbol{\theta}}$ 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. Besides these issues, another challenge related to the TD error is its sparsity. Specifically, the instantaneous TD signal remains low and largely uninformative for most of the episode, only to spike significantly during congestion events. This sparsity can lead to inefficient updates if not properly addressed. Therefore, we advocate that the use of experience replay and batch updates is essential to smooth the learning process and mitigate the risk of destabilisation due to abrupt parameter adjustments.

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.

VI. 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 as 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 traffic control performance and satisfy constraints in an automatic, data-driven fashion.

Future work will focus on 1) the validation of the proposed methodology with microscopic traffic model simulators, 2) the study of formal guarantees on stability and recursive feasibility for the proposed approach during learning and at convergence, 3) 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 4) extending the current approach by integrating variable speed limits (VSLs) with ramp metering, a coordinated control strategy that has been shown to achieve better results than ramp metering alone, especially in high demand/density regimes.

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.

[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 Trans. Intell. Transp. Syst.*, vol. 3, no. 4, pp. 271–281, 2002.

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

[7] M. Papageorgiou, H. Hadj-Salem, J.-M. Blosseville *et al.*, "ALINEA: A local feedback control law for on-ramp metering," *Transp. Res. Rec.*, vol. 1320, no. 1, pp. 58–64, 1991.

[8] Y. Wang, E. B. Kosmatopoulos, M. Papageorgiou, and I. Papamichail, "Local ramp metering in the presence of a distant downstream bottleneck: Theoretical analysis and simulation study," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 5, pp. 2024–2039, 2014.

[9] J. R. D. Frejo and B. De Schutter, "Feed-forward ALINEA: A ramp metering control algorithm for nearby and distant bottlenecks," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 7, pp. 2448–2458, 2019.

[10] 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 Am. Control Conf.*, vol. 5, 2002, pp. 3600–3605.

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

[12] A. Hegyi, B. De Schutter, and H. Hellendoorn, "Model predictive control for optimal coordination of ramp metering and variable speed limits," *Transp. Res. Part C: Emerg. Technol.*, vol. 13, no. 3, pp. 185–209, 2005.

[13] C. Wang, Y. Xu, J. Zhang, and B. Ran, "Integrated traffic control for freeway recurrent bottleneck based on deep reinforcement learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 15 522–15 535, 2022.

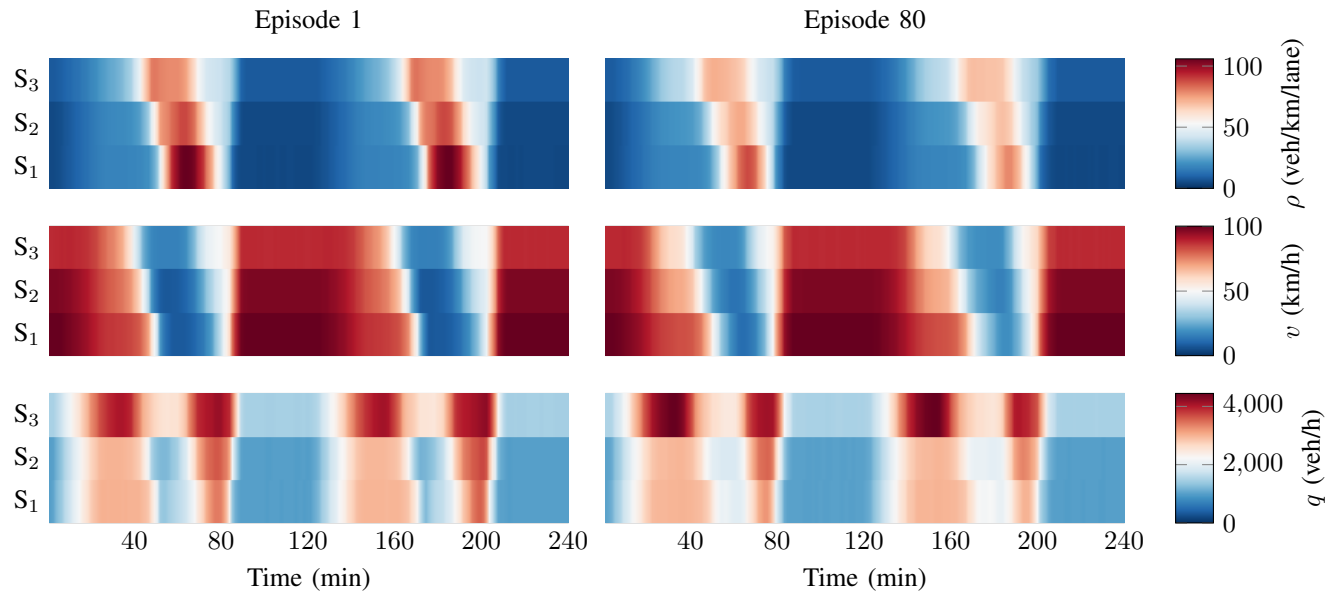

Fig. 8. Differences in traffic quantities of the network's three segments between the first episode and the last episode of the learning process, averaged across the 15 simulation runs

[14] Y. Han, M. Wang, L. Li, C. Roncoli, J. Gao, and P. Liu, "A physics-informed reinforcement learning-based strategy for local and coordinated ramp metering," *Transp. Res. Part C: Emerg. Technol.*, vol. 137, p. 103584, 2022.

[15] K. Shaaban, M. A. Khan, and R. Hamila, "Literature review of advancements in adaptive ramp metering," *Procedia Comput. Sci.*, vol. 83, pp. 203–211, 2016.

[16] F. Vrbanić, E. Ivanjko, K. Kušić, and D. Čakija, "Variable speed limit and ramp metering for mixed traffic flows: A review and open questions," *Appl. Sci.*, vol. 11, no. 6, 2021.

[17] X. Ma, A. Karimpour, and Y.-J. Wu, "Statistical evaluation of data requirement for ramp metering performance assessment," *Transp. Res. Part A: Policy Pract.*, vol. 141, pp. 248–261, 2020.

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

[19] 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.

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

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

[22] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, 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] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[25] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal, and T. Hester, "Challenges of real-world reinforcement learning: definitions, benchmarks and analysis," *Mach. Learn.*, vol. 110, no. 9, pp. 2419–2468, Sep 2021.

[26] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based model predictive control: toward safe learning in control," *Annu. Rev. Control, Robot., and Auton. Syst.*, 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," *Annu. Rev. Control, Robot., and Auton. Syst.*, 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 Am. Control Conf.* IEEE, 2022, pp. 342–357.

[29] L. Hewing, J. Kabzan, and M. N. Zeilinger, "Cautious model predictive control using Gaussian process regression," *IEEE Trans. Control Syst. Technol.*, 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 Syst. Lett.*, vol. 3, no. 3, pp. 577–582, 2019.

[31] K. P. Wabersich, L. Hewing, A. Carron, and M. N. Zeilinger, "Probabilistic model predictive safety certification for learning-based control," *IEEE Trans. Autom. Control*, vol. 67, no. 1, pp. 176–188, 2022.

[32] A. Romero, Y. Song, and D. Scaramuzza, "Actor-critic model predictive control," in *2024 IEEE Int. Conf. Robot. Autom.*, 2024, pp. 14 777–14 784.

[33] Q. Zhang, W. Pan, and V. Reppa, "Model-reference reinforcement learning for collision-free tracking control of autonomous surface vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 8770–8781, 2022.

[34] C. F. O. da Silva, A. Dabiri, and B. De Schutter, "Integrating reinforcement learning and model predictive control with applications to microgrids," *arXiv preprint arXiv:2409.11267*, 2024.

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

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

[37] S. Gros and M. Zanon, "Learning for MPC with stability & safety guarantees," *Automatica*, vol. 146, p. 110598, 2022.

[38] B. De Schutter and B. De Moor, "Optimal traffic light control for a single intersection," *Eur. J. Control*, vol. 4, no. 3, pp. 260–276, 1998.

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

[40] Y. Han, M. Ramezani, A. Hegyi, Y. Yuan, and S. Hoogendoorn, "Hierarchical ramp metering in freeways: An aggregated modeling and control approach," *Transp. Res. Part C: Emerg. Technol.*, vol. 110, pp. 1–19, 2020.

[41] U. Todorović, J. R. D. Frejo, and B. De Schutter, "Distributed MPC for large freeway networks using alternating optimization," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 3, pp. 1875–1884, 2022.

[42] S. Liu, A. Sadowska, and B. De Schutter, "A scenario-based distributed model predictive control approach for freeway networks," *Transp. Res. Part C: Emerg. Technol.*, vol. 136, p. 103261, 2022.

[43] E. Smaragdis, M. Papageorgiou, and E. Kosmatopoulos, "A flow-maximizing adaptive local ramp metering strategy," *Transp. Res. Part B: Methodol.*, vol. 38, no. 3, pp. 251–270, 2004.

[44] J. Chen, W. Lin, Z. Yang, J. Li, and P. Cheng, "Adaptive ramp metering control for urban freeway using large-scale data," *IEEE Trans. Veh. Technol.*, vol. 68, no. 10, pp. 9507–9518, 2019.

[45] Y. Wang and M. Papageorgiou, "Real-time freeway traffic state estimation based on extended kalman filter: a general approach," *Transp. Res. Part B: Methodol.*, vol. 39, no. 2, pp. 141–167, 2005.
[46] M. Davarynejad, A. Hegyi, J. Vrancken, and J. van den Berg, "Motorway ramp-metering control with queuing consideration using Q-learning," in *2011 14th Int. IEEE Conf. Intell. Transp. Syst.* IEEE, 2011, pp. 1652–1658.
[47] A. Fares and W. Gomaa, "Freeway ramp-metering control based on reinforcement learning," in *11th IEEE Int. Conf. Control Autom.* IEEE, 2014, pp. 1226–1231.
[48] E. Ivanjko, D. Koltovska Nečoska, M. Gregurić, M. Vujić, G. Jurković, and S. Mandžuka, "Ramp metering control based on the Q-learning algorithm," *Cybern. and Inf. Technol.*, vol. 15, no. 5, p. 88–97, 2015.
[49] S. El-Tantawy, B. Abdulhai, and H. Abdelgawad, "Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc): Methodology and large-scale application on downtown toronto," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 3, pp. 1140–1150, 2013.
[50] R. L. Abduljabbar, H. Dia, and P.-W. Tsai, "Development and evaluation of bidirectional LSTM freeway traffic forecasting models using simulation data," *Scientific Reports*, vol. 11, no. 1, p. 23899, Dec 2021.
[51] C. Gu, T. Zhou, and C. Wu, "Deep Koopman traffic modeling for freeway ramp metering," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 6, pp. 6001–6013, 2023.
[52] 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 Int. Conf. Syst. Theory, Control Comput.* IEEE, 2022, pp. 432–439.
[53] D. Sun, A. Jamshidnejad, and B. D. Schutter, "A novel framework combining MPC and deep reinforcement learning with application to freeway traffic control," *IEEE Trans. Intell. Transp. Syst.*, pp. 1–14, 2024.
[54] M. Treiber and A. Kesting, *Traffic Flow Dynamics: Data, Models and Simulation*. Berlin, Heidelberg: Springer, 2013.
[55] M. Papageorgiou, J.-M. Blosseville, and H. Hadj-Salem, "Macroscopic modelling of traffic flow on the Boulevard Périphérique in Paris," *Transp. Res. Part B: Methodol.*, vol. 23, no. 1, pp. 29–47, 1989.
[56] A. Messner and M. Papageorgiou, "METANET: a macroscopic simulation program for motorway networks," *Traffic Eng. & Control*, vol. 31, no. 8-9, pp. 466–470, 1990.
[57] M. Papageorgiou, I. Papamichail, A. Messmer, and Y. Wang, *Traffic Simulation with METANET*. New York: Springer, 2010, pp. 399–430.
[58] A. Dabiri and B. Kulcsár, "Distributed ramp metering—a constrained discharge flow maximization approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 9, pp. 2525–2538, 2017.
[59] D. Ngoduy and S. Hoogendoorn, "An automated calibration procedure for macroscopic traffic flow models," *IFAC Proc. Vol.*, vol. 36, no. 14, pp. 263–268, 2003.
[60] M. G. Lagoudakis, R. Parr, and M. L. Littman, "Least-squares methods in reinforcement learning for control," in *Method. Appl. of Artif. Intell.* Berlin, Heidelberg: Springer, 2002, pp. 249–260.
[61] H. N. Esfahani, A. B. Kordabad, and S. Gros, "Approximate robust NMPC using reinforcement learning," in *2021 Eur. Control Conf.* IEEE, 2021, pp. 132–137.
[62] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Mach. Learn.*, vol. 8, no. 3, pp. 293–321, 1992.
[63] 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.
[64] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. Springer, 2006.
[65] D. Ghosh, J. Rahme, A. Kumar, A. Zhang, R. P. Adams, and S. Levine, "Why generalization in RL is difficult: Epistemic POMDPs and implicit partial observability," in *Adv. Neural Inf. Process. Syst.*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 25 502–25 515.
[66] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: a software framework for nonlinear optimization and optimal control," *Math. Program. Computation*, vol. 11, no. 1, pp. 1–36, 2019.
[67] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, Mar 2006.
[68] A. D. Spiliopoulou, D. Manolis, I. Papamichail, M. Papageorgiou, J. Wu, and X. Jin, "Queue management techniques for metered freeway on-ramps," *Transp. Res. Rec.*, vol. 2178, no. 1, pp. 40–48, 2010.
[69] J. Wu, X.-Y. Chen, H. Zhang, L.-D. Xiong, H. Lei, and S.-H. Deng, "Hyperparameter optimization for machine learning models based on Bayesian Optimization," *J. Electron. Sci. Technol.*, vol. 17, no. 1, pp. 26–40, 2019.
[70] L. Franceschi, P. Frasconi, S. Salzo, R. Grazzi, and M. Pontil, "Bilevel programming for hyperparameter optimization and meta-learning," in *Proc. 35th Int. Conf. Mach. Learn.*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 1568–1577.

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 (32). Then, we report the evolution of all the parameters $\boldsymbol{\theta}$ during the learning process implemented in Section V.

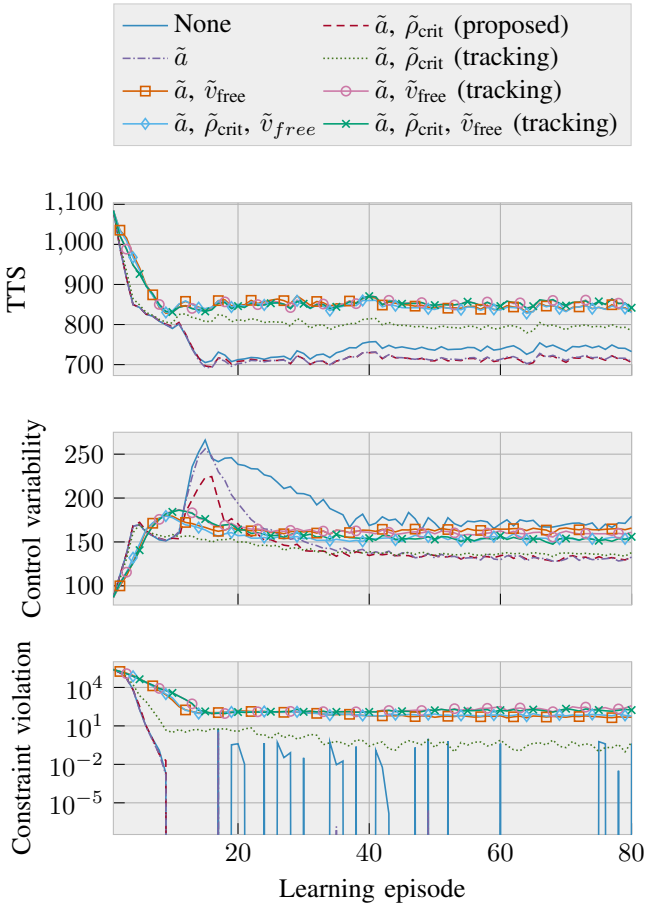

Fig. 9. Evolution of the RL cost contributions for various different parametrisations $\boldsymbol{\theta}$. The legend lists for each entry the dynamics parameters that were allowed to be learnt in the corresponding simulation, including also the parametrisation that we proposed in this work, i.e., *(proposed)*. If either $\tilde{\rho}_{\text{crit}}$, $\tilde{v}_{\text{free}}$ or both were also employed as tracking setpoints, the entry is marked accordingly, i.e., *(tracking)*. To avoid visual clutter, only the average over the 15 simulation runs is shown for each parametrisation.

DIFFERENT MPC PARAMETRISATIONS

Different parametrisations $\boldsymbol{\theta}$ of the MPC scheme (32) have been tested in simulations. In particular, we have considered the following variants:

- learning a combination of the dynamics parameters $\tilde{a}$, $\tilde{v}_{\text{free}}$, and $\tilde{\rho}_{\text{crit}}$ (including the empty set, i.e., learning none of these parameters)
- employing the learnable dynamics parameters $\tilde{v}_{\text{free}}$ and $\tilde{\rho}_{\text{crit}}$ also as tracking setpoints of the stage cost (34) and terminal cost (35), 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 parametrisation can potentially help in fitting more complex value 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 learning some of them may end up in a less stable process than others. As is the case with most function approximators, the choice of parametrisation relies mostly on prior and/or expert knowledge (such as, in our case, the sensitivity analysis of the METANET model w.r.t. its parameters) and is mostly an iterative procedure (both for MPC and other function approximators), whose difficulty is attributable to the RL algorithm itself rather than to the proposed control scheme. However, meta-learning methods can be found in the literature to optimise over the selection of this and other hyper-parameters via, e.g., Bayesian Optimisation [69] and bilevel optimisation [70].

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 $\boldsymbol{\theta}$ of the MPC scheme (32), as detailed in Section IV-B and simulated in Section V.

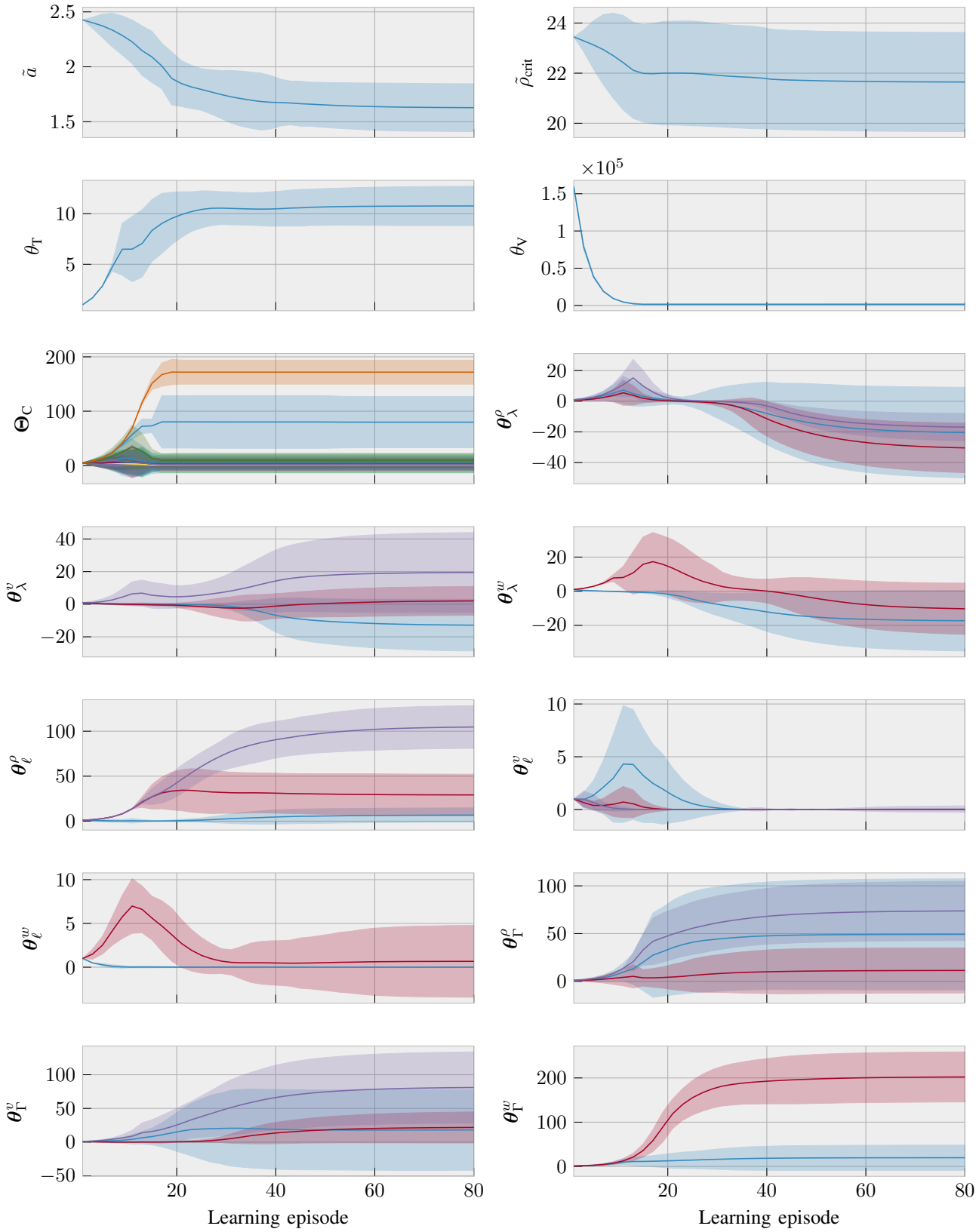

Fig. 10. Average evolution of the parametrisation $\boldsymbol{\theta}$ during the learning process across the 15 simulation runs. For those parameters that are vector quantities, one colour is associated with each entry.