Delft University of Technology

Delft Center for Systems and Control

Technical report 24-010

Adaptive parameterized model predictive control based on reinforcement learning: A synthesis framework*

D. Sun, A. Jamshidnejad, and B. De Schutter

If you want to cite this report, please use the following reference instead:

D. Sun, A. Jamshidnejad, and B. De Schutter, "Adaptive parameterized model predictive control based on reinforcement learning: A synthesis framework," *Engineering Applications of Artificial Intelligence*, vol. 136-B, p. 109009, Oct. 2024. doi:10.1016/j.engappai.2024.109009

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://pub.bartdeschutter.org/abs/24_010

Adaptive Parameterized Model Predictive Control Based on Reinforcement Learning: A Synthesis Framework*

Dingshan Sun[a,*], Anahita Jamshidnejad[b], Bart De Schutter[a]

[a]*Delft Center for Systems and Control, Delft University of Technology, Mekelweg 2, 2628 CD, Delft, The Netherlands*
[b]*Department of Control and Operations, Delft University of Technology, Kluyverweg 1, 2629 HS, Delft, The Netherlands*

Abstract

Parameterized model predictive control (PMPC) is one of the many approaches that have been developed to alleviate the high computational requirement of model predictive control (MPC), and it has been shown to significantly reduce the computational complexity while providing comparable control performance with conventional MPC. However, PMPC methods still require a sufficiently accurate model to guarantee the control performance. To deal with model mismatches caused by the changing environment and by disturbances, this paper first proposes a novel framework that uses reinforcement learning (RL) to adapt all components of the PMPC scheme in an online way. More specifically, the novel framework integrates various strategies to adjust different components of PMPC (e.g., objective function, state-feedback control function, optimization settings, and system model), which results in a synthesis framework for RL-based adaptive PMPC. We show that existing adaptive (P)MPC approaches can also be embedded in this synthesis framework. The resulting combined RL-PMPC framework provides a solution for an efficient MPC approach that can deal with model mismatches. A case study is performed in which the framework is applied to freeway traffic control. Simulation results show that for the given case study the RL-based adaptive PMPC approach reduces computational complexity by 98% on average compared to conventional MPC while achieving better control performance than the other controllers, in the presence of model mismatches and disturbances.

Keywords: Parameterized control, model predictive control, learning-based control, deep reinforcement learning, freeway traffic management, synthesis control framework.

1. Introduction & Motivation

Model predictive control (MPC) has been studied extensively within the last century, and mature theoretical results have been established for it (Morari and Lee, 1999; Camacho and Alba, 2013). MPC operates in a receding-horizon style, where an optimization problem is solved at every control step to determine a sequence of control inputs based on a prediction of the future states using a prediction model. Only the first element of this control input sequence is implemented in practice, and after shifting the prediction horizon to the next control step, the entire procedure is repeated. In addition, since MPC can explicitly deal with state and input constraints and provide robust control performance (Pannocchia et al., 2011), it has been widely used in engineering practice, such as industrial processes, power systems, robotics, and management of transportation networks (Qin and Badgwell, 2003; Hegyi et al., 2005). However, for a large number of real-life systems where the dynamics are in general nonlinear and nonconvex, the optimization problem of MPC may become too complex that is not feasible for real-time implementations. Besides, a sufficiently accurate model is needed to ensure the control performance, which is not always available in practice.

*This research has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (Grant agreement No. 101018826 - ERC Advanced Grant CLariNet), and also from the China Scholarship Council (CSC Grant No. 201806230254).

*Corresponding-author: Dingshan Sun

Email addresses: d.sun-1@tudelft.nl (Dingshan Sun), a.jamshidnejad@tudelft.nl (Anahita Jamshidnejad), b.deschutter@tudelft.nl (Bart De Schutter)

Therefore, the applications of MPC are often impacted by two main issues: high online computational complexity and model mismatches.

To address the first issue, a large number of studies have investigated computationally efficient MPC approaches, and have achieved satisfying results (see e.g., Alessio and Bemporad (2009); Karamanakos et al. (2015)). One of the successful methods is parameterized MPC (PMPC) (Lofberg, 2003; Goulart et al., 2006), which simplifies the optimization problem of conventional MPC by reducing the number of the optimization variables. More specifically, PMPC introduces a parametric state-feedback function as the control law. This means that the PMPC input function does not vary across the prediction horizon, and only the parameters in the control law need to be optimized per control step. More details of PMPC and its applications will be given in Section 2.1. Nonetheless, although PMPC can reduce the online computational complexity, it still suffers from model mismatches that are caused by the changing environment and by unknown disturbances.

To address this issue, extensive research efforts have been made. Two representative directions are robust MPC and stochastic MPC (Bemporad and Morari, 2007; Mesbah, 2016). Robust MPC solves a robust optimal control problem at each control step within the MPC scheme by considering the worst-case scenarios for the external disturbances, which results in conservatism in the control performance. Stochastic MPC considers the probability distribution of the uncertainties to guarantee the chance-constraint satisfaction on the basis of conventional MPC. However, these approaches design the controllers by assuming certain knowledge of the uncertainties. In addition, these methods require even more computational power than conventional MPC.

Thanks to the advancements in machine learning techniques, learning-based methods are gaining increasing attention and are being widely studied in engineering applications (Singh et al., 2023; Jiang et al., 2022). Accordingly, another representative direction to address the model mismatch issue of MPC is learning-based or data-driven MPC methods (Hewing et al., 2020; Li et al., 2022). Most of these methods focus on the identification of a system model, and are, thus, also known as adaptive MPC. More details about learning-based MPC will be given in Section 2.2. Another research direction that has recently drawn great interest is to combine MPC and reinforcement learning (RL) (Sutton and Barto, 2018). RL is a technique from the field of machine learning that learns how to take action in an uncertain environment so that to maximize an accumulative return. Deep reinforcement learning (DRL) incorporates deep learning techniques within conventional RL schemes, allowing agents to deal with problems with a large state space (Mnih et al., 2013). For simplicity, we use the term RL to refer to all RL-related algorithms, including DRL. RL has recently shown great potential in many fields, including optimal control (Mnih et al., 2015; Arulkumaran et al., 2017). Since RL is able to improve its control policy via interacting with the system and with its environment, it can naturally deal with unknown environments and disturbances. Despite these appealing features, RL still struggles with its own shortcomings, including constraint violations (i.e., safety issues) and low sample efficiency (i.e., a prolonged training process) (Dulac-Arnold et al., 2021). Due to the very complementary characteristics of RL and MPC (Görges, 2017), many studies have developed various methods to exploit the advantages of both techniques. More details about relevant research that combine MPC with RL will also be presented in Section 2.3.

Although there is a large body of research on solutions for each of the two main issues of MPC, i.e., high online computational complexity and model mismatches, very few papers consider addressing these two issues simultaneously. Therefore, this paper contributes to the state-of-the-art by developing a novel integrated RL-based adaptive PMPC synthesis framework (called RL-PMPC) that employs RL to adjust the PMPC scheme, in order to deal with the changing environment and disturbances, and also with the online computational complexity. This also leads to a unified framework for extension of existing adaptive MPC methods, and further improves PMPC in terms of control performance. The resulting RL-PMPC controller can overcome the two main challenges of conventional MPC by exploiting the advantages of RL and PMPC, and thus allows to broaden the application range of MPC-based methods.

The remainder of this paper is organized as follows. Section 2 introduces more details on related work. Section 3 presents the novel RL-PMPC synthesis framework and all the available strategies to adjust PMPC via RL techniques. Section 4 performs a case study to illustrate the performance of the proposed approach by applying it to traffic management for a freeway network. Section 5 concludes the paper and proposes topics for future work.

2. Related work

This section first explains PMPC, and then presents related work on learning-based MPC and adaptive MPC. In addition, recent studies that utilize RL within the MPC framework are presented.

2.1. PMPC

Lofberg (2003) introduced a feedback loop in the control input sequences of a robust MPC problem, by parameterizing the future control inputs in terms of future states and several new parameters. This parameterization process reduces the number of decision variables from the number of control inputs over the prediction horizon to the number of the introduced parameters. Note that conventional MPC can be regarded as a special case of PMPC, with an identity function as the parameterized control law. Goulart et al. (2006) further extended this approach by parameterizing the control input sequence as an affine function of the sequence of past disturbances. In this way, the disturbances can be accounted for by solving a convex optimization problem resulting from the parameterization. All of these studies focused on the robust MPC problem for linear systems. However, most systems in practice are nonlinear.

Zegeye et al. (2012) applied PMPC in freeway traffic management for the first time by parameterizing ramp metering rates and variable speed limits, such that compared to solving the original nonlinear MPC problem the computation time was significantly reduced without much loss of performance. Van Kooten et al. (2017) also employed the idea of parameterization to design a state-based adaptive controller for an urban traffic network. Pippia et al. (2018) applied PMPC to the operation of microgrids. Jeschke and De Schutter (2021) applied PMPC in signal control for urban traffic management, and achieved comparable control performance with substantially decreased computation time, compared to conventional MPC. However, the design of the parametric function that maps the states to control inputs is difficult and often requires expert knowledge. Jeschke et al. (2023) addressed this issue by introducing a grammatical evolution method to generate the parametric function automatically. They applied this method for traffic signal control of an urban network. The results show that the generated parametric state-feedback function even outperformed the handcrafted function.

2.2. Learning-based adaptive MPC

This section presents a brief overview of the literature on learning-based MPC, whose major purpose is to handle model uncertainties during the implementation of MPC. Hewing et al. (2020) gave a comprehensive review of learning-based MPC approaches, in which the reader can find more details. Conventional adaptive MPC refers to the studies that focus on system identification to compensate for model uncertainties. In this paper, we broaden the scope of adaptive MPC to any MPC approach that can adapt to model uncertainties and disturbances.

2.2.1. Adaptive MPC by system identification

Lorenzen et al. (2017) considered a constrained linear system with unknown but constant system parameters. A set-membership system identification method is used to estimate the set that contains the real parameter values, which results in a robust MPC problem. Then tube MPC techniques (Raković et al., 2012) are used to solve the problem, and to construct the terminal constraint and terminal set to guarantee stability and recursive feasibility. Heirung et al. (2017) proposed an adaptive dual MPC approach for a single-input single-output linear time-invariant system for which the dynamic matrices are known and determined by orthogonal basis functions. A recursive least squares method is used to estimate the unknown parameters using observed data. The resulting optimization problem is subject to probabilistic output constraints, and is then reformulated as a quadratic programming problem that can be solved efficiently. Tanaskovic et al. (2019) also employed a two-stage method for adaptive MPC of a linear time-varying multiple-input multiple-output system subject to model uncertainties and measurement noise. First, a set-membership algorithm is used to estimate the parameter matrix. Then the obtained set is exploited in the MPC optimization problem to enforce constraints, which results in a robust finite-horizon optimal control problem. A similar method was used by Zhang and Shi (2020) for adaptive linear MPC, where the main contribution consists in adding extra variables to the optimization problem in order to adjust the shape and size of the cross section of the tube. The obtained results are less conservative w.r.t. conventional adaptive MPC, while guaranteeing closed-loop stability and recursive feasibility.

The above studies are all about linear systems. There are also a few studies that focus on nonlinear adaptive MPC. Adetola et al. (2009) proposed adaptive MPC for constrained nonlinear systems, where the model uncertainties are assumed to be static and can be expressed by unknown constant parameters. An uncertainty set is updated recursively to estimate the bounds of these parameters, which results in a robust MPC problem that is solved via both a Min-Max approach and a Lipschitz-based approach. Köhler et al. (2021) presented a tube-based robust adaptive MPC approach for an uncertain nonlinear system subject to unknown constant model parameters and additive disturbances.

Compared to ordinary robust MPC problems, the work improves the computational efficiency by modifying the tube formulations, while providing robust recursive feasibility and robust constraint satisfaction. Akpan and Hassapis (2011) proposed to utilize neural networks to approximate the system model for MPC, since neural networks can approximate any nonlinear function with an arbitrary high accuracy (Kumpati et al., 1990). The neural network is trained online based on a recursive least squares algorithm, and the resulting optimization problem with a neural network-based model is solved using gradient-based methods.

All the studies presented so far enforce assumptions on the structure of the system dynamics and model uncertainties. The uncertainty parameters (whether constant or varying) are assumed to be parametric (i.e., the system dynamics is linear in the parameters), which limits the application of adaptive MPC. In addition, robust MPC techniques used to solve the optimization problem will introduce conservatism in control performance.

2.2.2. Adaptive MPC by adjusting the controller

Another direction of learning-based adaptive MPC is to adjust the controller design, such as learning the cost function, the constraint set, or the terminal components. Marco et al. (2016) considered the design of a linear quadratic regulator (LQR) for a linearized model. Instead of identifying the model, they directly tuned the introduced parametric cost function by iteratively evaluating the controller on the real system. This approach can also be integrated into an MPC scheme as in Bansal et al. (2017). Piga et al. (2019) adjusted the model parameters oriented towards the overall performance of the MPC controller, instead of minimizing the error between the model and the real system. In addition, the size of the prediction horizon is also added to the parameters to be adapted. Brunner et al. (2015) worked on enlarging the terminal set of an MPC controller for linear systems by using the collected historical data. Rosolia and Borrelli (2017) focused on iterative tasks with nonlinear MPC, in which the terminal cost and the terminal set are adjusted at every iteration in order to guarantee constraint satisfaction and system stability. Experiences from previous iterations are employed, and it is ensured that the cost does not increase from iteration to iteration. They further extended the result to a robust control context in Rosolia et al. (2017).

2.3. RL-based adaptive MPC

RL-based adaptive MPC is another direction that utilizes RL techniques to obtain adaptive MPC. Due to the complementary features of MPC and RL, as mentioned in the previous section, combining MPC and RL is a promising research direction that has been drawing more and more attention recently. The very early study that employed RL in an MPC scheme is Negenborn et al. (2005), in which the value function of RL was used to represent the infinite-horizon objective function value of MPC for Markov decision processes. The value function can be learned on-line during the implementation, and the MPC prediction horizon reduced to one step due to the approximation of the objective function. This idea opened up a research direction to combine MPC and RL, and has been adopted in many studies, such as Zhong et al. (2013); Zhang et al. (2020); Arroyo et al. (2022). Although this approach can alleviate the computational issue of conventional MPC by reducing the prediction horizon, it still suffers from model mismatches since the control inputs are optimized based on the prediction model. In addition, solving the optimization problem with a value function in the objective function is even more difficult due to the introduced extra nonlinearity. By connecting the objective function of MPC and the value function of RL, Gros and Zanon (2019) parameterized the objective function of MPC and adapted it using RL. It is shown that the optimal policy can be obtained even based on an inaccurate model by modifying the objective function. However, it is not explained how to parameterize the objective function in a structured way, which is the core procedure for implementing this method.

The other main direction to combine MPC and RL is to merge their control inputs directly. Zhang et al. (2021) integrated an RL agent in a model-reference scheme together with a conventional nonlinear controller. The RL agent is trained by performing repetitive tasks to compensate for the mismatches between the nominal model and the real system, and to eliminate the errors between the real states and the desired states. This idea is adopted by Remmerswaal et al. (2022), which combines RL and MPC in a model-reference framework and applies the resulting MPC-RL framework to traffic signal control for urban networks. Sun et al. (2024) further extended the work by constructing a hierarchical framework in which the MPC and RL controllers work with different control frequencies and their control inputs are summed. The resulting framework is applied to traffic management of freeway networks, and the results show that the combined MPC-RL controller can excellently deal with model mismatches. Another related study is by Hosseini et al. (2023). They proposed a hierarchical structure for power distribution system restoration, in which the

Table 1: Definitions of the mathematical notations

Notation	Definition
$F(\cdot)$	Prediction model for the controlled system
k_s	Simulation step counter of the prediction model
k_c	Control step counter of the controlled system
k_p	Operation step counter of the PMPC scheme
k_rl	Operation step counter of the RL agent
T_s	Simulation sampling time of the prediction model
T_c	Control sampling time of the controlled system
T_p	Operation sampling time of the PMPC scheme
T_rl	Operation sampling time of the RL agent
$\boldsymbol{x}(k_\mathrm{s})$	Measured state at time step k_s
$\hat{\boldsymbol{x}}(k_\mathrm{s})$	Predicted state at time step k_s
$\tilde{\boldsymbol{x}}(k_\mathrm{p})$	Sequence of the predicted states over the prediction horizon at the PMPC operation step k_p
$\bar{\boldsymbol{x}}(k_\mathrm{rl})$	Measured system states at the RL operation step k_rl
$\boldsymbol{u}_\theta(k_\mathrm{p})$	Optimization variables of PMPC at the PMPC operation step k_p
$\boldsymbol{u}_\mathrm{c}(k_\mathrm{c})$	Control input generated by the parameterized control law at control step k_c
$\tilde{\boldsymbol{u}}(k_\mathrm{p})$	Sequence of control inputs over the prediction horizon at the PMPC operation step k_p
$\bar{\boldsymbol{u}}(k_\mathrm{rl})$	Implemented control inputs at the RL operation step k_rl
$N_\mathrm{p,o}$	Prediction horizon size counted in terms of the PMPC operation steps
$N_\mathrm{p,c}$	Prediction horizon size counted in terms of the control time steps
$N_\mathrm{p,s}$	Prediction horizon size counted in terms of the simulation time steps
N_b	Number of PMPC operation steps where $\boldsymbol{u}_\theta(k_\mathrm{p})$ remains constant within the prediction window

Note: Without loss of generality, $F(\cdot)$ is the discretized model of the controlled system with a sampling time T_s. For simplicity, the measurement sampling time is taken to be equal to the simulation sampling time. Each time step k_s corresponds to the time interval $[k_\mathrm{s}T_\mathrm{s}, k_\mathrm{s}T_\mathrm{s} + T_\mathrm{s})$ for the real system. Similar statements hold for step $k_\mathrm{c}, k_\mathrm{p}$, and k_rl.

RL agents work at the lower level to make fast decisions on the active power dispatches, and a quadratic programming agent operates at the higher level using the local RL decisions to check the major grid constraints and to ensure system resilience. Based on the commands from the high-level controller, the RL agents revised their actions accordingly.

Although many studies have explored the combination of MPC and RL in various fields, so far there is not a comprehensive survey on this topic. The authors believe that the potentials of combining MPC and RL have not yet been fully developed. In addition, despite the fruitful results of adaptive MPC, relevant research on PMPC is quite limited. Therefore, in this paper, a synthesis framework that utilizes RL to adjust PMPC is presented. It will be shown that not only the RL-MPC methods in Section 2.3, but also the learning-based adaptive MPC techniques introduced in Section 2.2 can be extended to PMPC and embedded in this framework.

3. The synthesis framework of RL-based adaptive PMPC

In this section we first extend the conventional definition of PMPC such that all the components of PMPC can be modified. Based on this definition, we present a novel synthesis framework for RL-PMPC, and further consider five cases, each corresponding to a specification of the novel framework by parameterizing a different component of PMPC. The frequently used mathematical notations are defined in Table 1.

3.1. Extended PMPC scheme

In a general PMPC scheme, there are three time scales: the simulation sampling time T_s of the prediction model, the control sampling time T_c, and the PMPC operation sampling time T_p, and the corresponding counting steps are k_s, k_c, and k_p. The relationships between them are:

$$T_\mathrm{p} = m_2 \cdot T_\mathrm{c} = m_2 m_1 \cdot T_\mathrm{s}, \quad m_1, m_2 \in \mathbb{N}^+, \tag{1}$$

with m_1 and m_2 positive integers, $\mathbb{N}^+$ the set of positive integer values. The output parameters generated by PMPC at operation step k_p are assumed to remain constant during time interval $[k_\mathrm{p}T_\mathrm{p}, (k_\mathrm{p} + 1)T_\mathrm{p})$, while the control inputs given to the system are updated every T_c time units based on the parameterized control laws, and the states which are

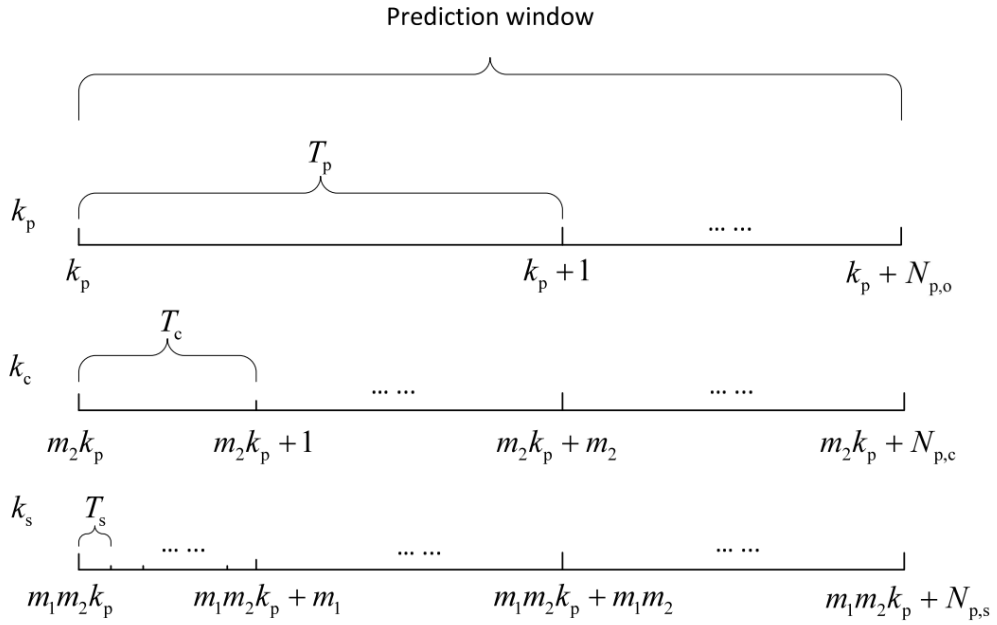

Figure 1: Illustration of different time scales of the PMPC scheme within one prediction window, in which k_p is the PMPC operation step, k_c is the control step, and k_s is the simulation step of the prediction model.

measured every T_s time units[1]. Within the prediction window, each PMPC operation step k_p corresponds to the control steps $\{m_2k_\mathrm{p}, \ldots, m_2k_\mathrm{p}+m_2-1\}$, and each control step k_c corresponds to the simulation steps $\{m_1k_\mathrm{c}, \ldots, m_1k_\mathrm{c}+m_1-1\}$ of the prediction model. The relationships among different time scales over a prediction window are illustrated in Figure 1, in which

$$N_{\mathrm{p,s}} = m_1 \cdot N_{\mathrm{p,c}} = m_1 m_2 \cdot N_{\mathrm{p,o}}, \quad m_1, m_2 \in \mathbb{N}^+. \tag{2}$$

Considering a PMPC problem for a general nonlinear system with state and input constraints, the following optimization problem needs to be solved at every PMPC operation step k_p:

$$\min_{\boldsymbol{u}_\theta(k_\mathrm{p})} \quad J(\tilde{\boldsymbol{x}}(k_\mathrm{p}), \tilde{\boldsymbol{u}}(k_\mathrm{p}), \boldsymbol{\theta}_J) \tag{3}$$

$$\begin{aligned}\text{s.t. } \hat{\boldsymbol{x}}(m_1(m_2k_\mathrm{p}+k)+\ell+1) = \\ F(\hat{\boldsymbol{x}}(m_1(m_2k_\mathrm{p}+k)+\ell), \boldsymbol{u}_\mathrm{c}(m_2k_\mathrm{p}+k), \boldsymbol{\theta}_F), \\ \text{for } \ell = 0, \ldots, m_1-1,\ k = 0, \ldots, N_{\mathrm{p,c}}-1,\end{aligned} \tag{4}$$

$$\mathcal{G}(\tilde{\boldsymbol{x}}(k_\mathrm{p}), \tilde{\boldsymbol{u}}(k_\mathrm{p}), \boldsymbol{\theta}_\mathcal{G}) \leq 0, \tag{5}$$

$$\tilde{\boldsymbol{x}}(k_\mathrm{p}) = [\hat{\boldsymbol{x}}^\top(m_1m_2k_\mathrm{p}+1), \ldots, \hat{\boldsymbol{x}}^\top(m_1m_2k_\mathrm{p}+m_1N_{\mathrm{p,c}})]^\top, \tag{6}$$

$$\tilde{\boldsymbol{u}}(k_\mathrm{p}) = [\boldsymbol{u}_\mathrm{c}^\top(m_2k_\mathrm{p}), \ldots, \boldsymbol{u}_\mathrm{c}^\top(m_2k_\mathrm{p}+N_{\mathrm{p,c}}-1)]^\top, \tag{7}$$

$$\begin{aligned}\boldsymbol{u}_\mathrm{c}(m_2k_\mathrm{p}+k) = f(\hat{\boldsymbol{x}}(m_1(m_2k_\mathrm{p}+k)), \boldsymbol{u}_\theta(k_\mathrm{p}), \boldsymbol{\theta}_f), \\ \text{for } k = 0, \ldots, N_{\mathrm{p,c}}-1,\end{aligned} \tag{8}$$

$$\hat{\boldsymbol{x}}(m_1m_2k_\mathrm{p}) = \boldsymbol{x}(m_1m_2k_\mathrm{p}), \tag{9}$$

in which $\boldsymbol{u}_\theta(k_\mathrm{p})$ denotes the parameter variables to be optimized every operation step, $F(\cdot)$ is the prediction model that is parameterized by $\boldsymbol{\theta}_F$, and $\boldsymbol{x}(m_2m_1k_\mathrm{p})$ is the measured state vector at the time instant $m_1m_2k_\mathrm{p}$; $J(\cdot)$ is the objective function parameterized by $\boldsymbol{\theta}_J$, and $\mathcal{G}$ represents the constraint for the control inputs and states parameterized by $\boldsymbol{\theta}_\mathcal{G}$; f is the state-feedback function, which maps $\boldsymbol{u}_\theta$ to $\boldsymbol{u}_\mathrm{c}$ and which is parameterized by $\boldsymbol{\theta}_f$. Typically, only the first element of the optimized parameter vector, i.e., $\boldsymbol{u}_\theta(k_\mathrm{p})$, is implemented, and the optimization problem is solved again at the next operation step $k_\mathrm{p}+1$.

[1] In general, the measurement sampling time can be different to allow for measurement of states, constraints, and performance criteria.

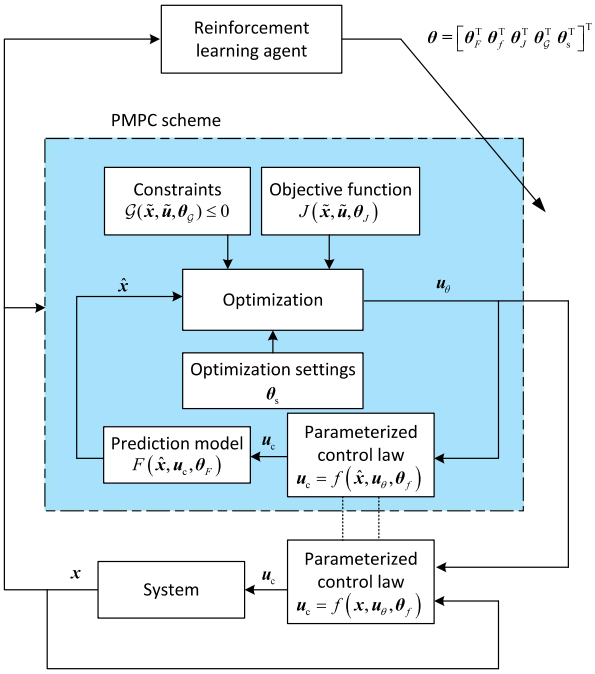

Figure 2: The synthesis framework of RL-PMPC

Remark 1. *For the sake of simplicity, it is assumed in the PMPC formulation* (3)-(9) *that* $\boldsymbol{u}_\theta(k_\mathrm{p})$ *remains constant during the entire PMPC prediction window. However, it is straightforward to allow* $\boldsymbol{u}_\theta(k_\mathrm{p})$ *to vary with the PMPC operation step within the PMPC prediction window. One choice is to keep the parameters constant for an interval (e.g., several time steps) and then change for the next interval over the prediction horizon, which is a hybrid option called move blocking (Cagienard et al., 2007).*

In conventional PMPC (Lofberg, 2003; Goulart et al., 2006), only the control inputs are parameterized and the resulting parameters $\boldsymbol{u}_\theta$ are optimized. If we simplify the state-feedback function (8) to an identity function such that $\boldsymbol{u}_\mathrm{c} = \boldsymbol{u}_\theta$, then the PMPC problem reduces to a conventional MPC problem. In the PMPC problem (3)-(9), in addition to the state-feedback function, the other components of the PMPC scheme are all parameterized, including the constraint sets, the objective function, and the system model. This yields an extended PMPC scheme that can cover the existing MPC methods introduced in Section 2. However, if we take all the parameters as decision variables in (3)-(9), the resulting optimization problem will be difficult to solve, due to the large number of optimization variables and the nonlinearity and nonconvexity introduced by the parameters. This issue can be addressed by the proposed RL-based adaptive PMPC synthesis framework.

3.2. The synthesis framework

Instead of designing a specific scheme and tailor a solution for each possible parameterization case separately, we propose to integrate all the possible solutions in an RL-PMPC synthesis framework. As shown in Figure 2, all parameterization cases are embedded in this framework, and a high-level RL agent is employed to adapt the parameterized components, such that the complex optimization problem with multiple parameters is avoided. Note that the RL agent in general works with a lower frequency than PMPC to adjust the parameters $\boldsymbol{\theta} = [\boldsymbol{\theta}_F^\top, \boldsymbol{\theta}_f^\top, \boldsymbol{\theta}_J^\top, \boldsymbol{\theta}_{\mathcal{G}}^\top, \boldsymbol{\theta}_\mathrm{s}^\top]^\top$ of all the parametric components.

The high-level RL agent directly adjusts the parameters such that they can be regarded as constants during the PMPC computation procedure. This simplifies the optimization problem of PMPC and makes the framework computationally efficient. In addition, by parameterizing the control inputs via $\boldsymbol{\theta}_f$, the number of the optimization variables of the proposed framework can be further reduced. One additional advantage of the proposed framework is that each parameterization case can be implemented either alone or jointly with other parameterization cases. This significantly

improves the ability and flexibility of the framework to deal with varying or unknown environments and disturbances. Next, we first define the RL agent and then illustrate the proposed framework by detailing each case separately.

3.2.1. Definition of the RL agent

The definition of the RL agent within the framework is related to the learning goal and the learning process, whereas the extended PMPC framework together with the controlled system can be regarded as the environment of the RL agent. The high-level RL agent introduces an extra time scale, i.e., RL adapts the PMPC controller with an operation sampling time T_{rl}:

$$T_{\mathrm{rl}} = m_3 \cdot T_{\mathrm{p}}, \quad m_3 \in \mathbb{N}^+, \tag{10}$$

and with the corresponding adaption step k_{rl}. The reinforcement learning process is modeled as a discrete-time stochastic control process (i.e., a Markov decision process (MDP)), which can be represented by a five-tuple $\langle S, A, P, \mathcal{R}, \gamma \rangle$. According to the RL agent within the framework (see Figure 2), these elements are defined as follows:

- S: State space, which is the set of all possible states $s_{k_{\mathrm{rl}}}$ of the environment per step k_{rl}. In this framework, the state space may include the measured system states $\boldsymbol{x}(k_{\mathrm{s}})$ where $k_{\mathrm{s}} = m_3 m_2 m_1 k_{\mathrm{rl}}$, the measurable external disturbances, and the control inputs generated by the low-level PMPC controller. To facilitate the learning process, the state values are normalized to the same scale.

- A: Action space, which is the set of all possible actions $a_{k_{\mathrm{rl}}}$ that can be taken by the DRL agent based on state $s_{k_{\mathrm{rl}}}$ at operation step k_{rl}. In this framework, the action can include values of the parameters, which is defined in the most general case by:

  $$a_{k_{\mathrm{rl}}} = \boldsymbol{\theta} = [\boldsymbol{\theta}_F^\top, \boldsymbol{\theta}_f^\top, \boldsymbol{\theta}_J^\top, \boldsymbol{\theta}_{\mathcal{G}}^\top, \boldsymbol{\theta}_{\mathrm{s}}^\top]^\top. \tag{11}$$

  The action can contain the parameters of a single or multiple components of the PMPC scheme during implementations. In order to avoid safety issues or significant performance fluctuations during the learning process of the framework, the action space (i.e., the range of the parameters) of the RL agent can be restricted to a relatively safe set based on previous experiences or expert knowledge. When the modified values $\boldsymbol{\theta}$ of the parameters violate their given upper and lower bounds (i.e., $\overline{\boldsymbol{\theta}}$ and $\underline{\boldsymbol{\theta}}$), the values will be saturated within the bounds.

- P: A function of the state and the action that determines the transition probability among the states when taking the corresponding action. In this framework, this function is implicitly defined jointly by the PMPC scheme and the system.

- $\mathcal{R}$: Reward function, which generates the immediate reward $r_{k_{\mathrm{rl}}}(s_{k_{\mathrm{rl}}}, a_{k_{\mathrm{rl}}})$ when taking action $a_{k_{\mathrm{rl}}}$ at state $s_{k_{\mathrm{rl}}}$. The reward function is the core component of an RL agent, as it determines the learning goal. Since the proposed framework is performance-driven, the reward function should contain the performance criteria of the system, which can include the objective function $J(\cdot)$ used in the PMPC scheme and other extra performance indices (e.g., computation time or penalty on constraint violations).

- $\gamma \in [0, 1)$: A user-defined discount factor on future rewards.

The goal of learning is to find a policy $\pi : A \times S \to [0, 1]$, $\pi(a, s) = \Pr(a_{k_{\mathrm{rl}}} = a | s_{k_{\mathrm{rl}}} = s)$, that maximizes the accumulative long-term reward, which is the expected return defined by:

$$\begin{aligned} Q^\pi\left(s_{k_{\mathrm{rl}}}, a^\pi_{k_{\mathrm{rl}}}\right) =& \mathbb{E}_{r,s\sim E}\left[\sum_{k_{\mathrm{rl}}=0}^{\infty} \gamma^{k_{\mathrm{rl}}} r_{k_{\mathrm{rl}}}(s_{k_{\mathrm{rl}}}, a^\pi_{k_{\mathrm{rl}}})\right] \\ =& \mathbb{E}_{r,s\sim E}\left[r_{k_{\mathrm{rl}}}(s_{k_{\mathrm{rl}}}, a^\pi_{k_{\mathrm{rl}}}) + \gamma Q^\pi\left(s_{k_{\mathrm{rl}}+1}, a^\pi_{k_{\mathrm{rl}}+1}\right)\right], \end{aligned} \tag{12}$$

where the subscript $r, \boldsymbol{s} \sim E$ denotes the stochastic transitions among the states in the environment, and $a^\pi_{k_{\mathrm{rl}}}$ is the action taken at step k_{rl} based on the policy $\pi(\cdot)$. Depending on the specific problem, various RL algorithms can be chosen. In particular, when dealing with large-scale problems with multiple parameters, deep RL algorithms that can address continuous state space are preferred, such as Deep Q-Network (DQN) or actor-critic algorithms (Mnih et al., 2013, 2016; Lillicrap et al., 2015; Haarnoja et al., 2018).

Algorithm 1 Offline learning process of the RL-PMPC synthesis framework

1: Initialize the DDPG agent: the critic and actor networks, the corresponding target networks and the experience replay buffer
2: Initialize the PMPC scheme by determining the parameterization, and define the state space, action space, and reward function of the DDPG agent
3: **for** episode from 1 to M **do**
4: Initialize the system states
5: **for** every RL adaption step k_{rl} **do**
6: Observe state $s_{k_{\mathrm{rl}}}$
7: Take action $a_{k_{\mathrm{rl}}}$ according to state $s_{k_{\mathrm{rl}}}$ and policy $\pi(\cdot)$, and update parameters $\boldsymbol{\theta}$ of the PMPC scheme
8: **for** every PMPC operation step k_{p} **do**
9: Measure state $\boldsymbol{x}(m_2 m_1 k_{\mathrm{p}})$
10: Solve the PMPC problem (3), and get the optimized parameter $\boldsymbol{u}_\theta(k_{\mathrm{p}})$
11: **for** every control step k_{c} **do**
12: Measure state $\boldsymbol{x}(m_1 k_{\mathrm{c}})$ and calculate the control inputs $\boldsymbol{u}_{\mathrm{c}}(k_{\mathrm{c}})$ according to (8)
13: **for** every step k_{s} **do**
14: Implement control input $\boldsymbol{u}_{\mathrm{c}}(k_{\mathrm{c}})$ on the simulation model; measure and record states $\boldsymbol{x}(k_{\mathrm{s}}+1)$
15: **end for**
16: **end for**
17: **end for**
18: Observe the reward $r_{k_{\mathrm{rl}}}$ and next state $s_{k_{\mathrm{rl}}+1}$
19: Store transition $(s_{k_{\mathrm{rl}}}, a_{k_{\mathrm{rl}}}, s_{k_{\mathrm{rl}}+1}, r_{k_{\mathrm{rl}}})$ in the replay buffer
20: Sample a mini-batch of N data points from replay buffer randomly
21: Update the critic and actor (target) networks based on the sampled data according to Lillicrap et al. (2015)
22: **end for**
23: **end for**

Remark 2. *The learning process can be conducted offline (i.e., using a detailed simulation model to generate data), which is known as pre-training, or online (i.e., interacting with the real system), or via a combination of online and offline processes. Both variants have a similar training process (see Algorithm 1).*

Algorithm 1 summarizes the overall learning process of the proposed framework, and a deep RL algorithm, i.e., deep deterministic policy gradient (DDPG) (Lillicrap et al., 2015), is used for example[2]. The RL agent is trained for M episodes, and each episode starts from the initial state and ends in the terminal state or at the terminal time step. Note that Algorithm 1 can be easily extended to other RL algorithms. In addition, the online variant of Algorithm 1 can be obtained by changing line 14, in which the simulation model is replaced by the real system. Next, the RL-based modification of each component of the PMPC scheme is discussed separately in detail.

3.2.2. Case A: RL modifying the system model

Adjusting the model parameters can reduce the mismatch between the prediction model and the real system, thus resulting in more accurate predictions and better control performance. Therefore, the states $s_{k_{\mathrm{rl}}}$ of the RL agent in the adaptation step k_{rl} can include the measured states of the real system at the corresponding time step, i.e., $\boldsymbol{x}(k_{\mathrm{s}})$ with $k_{\mathrm{s}} = m_3 m_2 m_1 k_{\mathrm{rl}}$, and other necessary information about the environment, such as disturbances and PMPC inputs. In this section, the objective of RL (i.e., the reward function) can be either minimizing the modeling errors, as in Wang et al. (2020) and Hu et al. (2022), or optimizing the control performance directly. For the former case, the reward function can be defined to minimize the error between the predicted states and the measured states. For the latter case, the reward function can be defined to minimize the objective function in PMPC:

$$r_{k_{\mathrm{rl}}}(s_{k_{\mathrm{rl}}}, a_{k_{\mathrm{rl}}}) = -R(\bar{\boldsymbol{x}}(k_{\mathrm{rl}}), \bar{\boldsymbol{u}}(k_{\mathrm{rl}})), \tag{13}$$

[2]The explanation of the DDPG techniques, i.e., critic and actor structure, experience replay, and target networks, is omitted here for compactness. For more details, the reader can refer to Mnih et al. (2013); Lillicrap et al. (2015).

where $R(\cdot)$ can be similar to the PMPC objective function $J(\cdot)$, and $\bar{\boldsymbol{x}}$ and $\bar{\boldsymbol{u}}$ include the measured states and implemented PMPC inputs during time interval $[k_{\mathrm{rl}}T_{\mathrm{rl}}, (k_{\mathrm{rl}}+1)T_{\mathrm{rl}})$. For linear systems with parametric uncertainties (e.g., see Zhang and Shi (2020); Lorenzen et al. (2017)), one way is to parameterize the system as:

$$F(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\theta}_F) = A(\boldsymbol{\theta}_F)\boldsymbol{x} + B(\boldsymbol{\theta}_F)\boldsymbol{u}. \tag{14}$$

The parameters $\boldsymbol{\theta}_F$ can be adjusted by the RL agent directly at every adaptation step k_{rl}, and the action can be the corresponding parameter values. In this way, RL can adjust the varying parameters caused by the changing environment, via interacting with the environment or a simulation model. This strategy can also be extended to more general linear systems with other parametric models than (14).

For nonlinear systems, consider a widely-used parametric nonlinear model (Adetola et al., 2009):

$$F(\hat{\boldsymbol{x}}, \boldsymbol{u}, \boldsymbol{\theta}_F) = F_f(\hat{\boldsymbol{x}}, \boldsymbol{u}) + F_g(\hat{\boldsymbol{x}}, \boldsymbol{u})\boldsymbol{\theta}_F, \tag{15}$$

where $\boldsymbol{\theta}_F$ can be the adjusted by the RL agent. Moreover, according to Tóth (2010), basis functions can be used to construct linear parameter-varying models. Then the RL agent can be used to tune the weights of the chosen basis functions. This can also be applied to the cases where artificial neural networks (ANNs) are used to approximate the nonlinear system. For example, the RL agent can be used to tune the weights of the neurons of ANNs (Akpan and Hassapis, 2011), or to compensate for the modeling errors between the real system and the ANNs (Perrusquía and Yu, 2021). Furthermore, some systems have different modes or dynamics under different working conditions, which can be described by switching among several system models. Then the RL agent can be used to select the suitable system model to adapt to varying conditions.

3.2.3. Case B: RL modifying the parameterized control laws

Parameterizing the control laws with state-feedback functions (i.e., control law $f(\cdot)$) is the main way to reduce computation time in this framework. Several studies (see, e.g., Zegeye et al. (2012); Van Kooten et al. (2017); Pippia et al. (2018)) consider a fixed control law that is pre-designed based on the experience or expert knowledge. This handcrafted design may work well for some scenarios, but the control performance may deteriorate when the system conditions change over time. Therefore, in this section, the RL agent within this framework allows to tune the control law $f(\cdot)$, which is parameterized by $\boldsymbol{\theta}_f$. Consider the following example, where the control law $f(\cdot)$ is a combination of several basis functions:

$$f(\hat{\boldsymbol{x}}, \boldsymbol{u}_\theta, \boldsymbol{\theta}_f) = \sum_{i=1}^{n_f} \theta_{f,i}\boldsymbol{\phi}_{f,i}(\hat{\boldsymbol{x}}, \boldsymbol{u}_\theta), \tag{16}$$

in which $\boldsymbol{\theta}_f = [\theta_{f,1}, \ldots, \theta_{f,n_f}]^\top$ with n_f the number of the basis functions. Accordingly, the action of the RL agent can be defined by (11). The basis functions $\boldsymbol{\phi}_{f,i}(\hat{\boldsymbol{x}}, \boldsymbol{u}_\theta), i = 1, \ldots, n_f$ should be designed a priori to handle various system conditions. They can be constructed empirically or by resorting to a learning-based method. For example, the grammatical evolution algorithm can be used to generate the control laws automatically in an offline style (Jeschke et al., 2023). Furthermore, the state space and the reward function for this case can be defined in the same way as in Case A.

3.2.4. Case C: RL modifying the objective function and constraint sets

It has been shown that the objective function and constraints within the PMPC scheme can be adjusted to further improve the control performance (Gros and Zanon, 2019), while it has not been illustrated how to systematically parameterize the objective function. Arroyo et al. (2022) also approximated the infinite cost of the objective function by using a stage cost and a value function. However, this leads to increased complexity in solving the optimization problem, since the value function introduces extra nonlinearity and nonconvexity to the optimization problem. Arroyo et al. (2022) used an exhaustive search method to find an optimal action per time step. In this section, we propose to rewrite the objective function (3) as in Gros and Zanon (2019), which is given by:

$$\begin{aligned} &J(\tilde{\boldsymbol{x}}(k_{\mathrm{p}}), \tilde{\boldsymbol{u}}(k_{\mathrm{p}}), \boldsymbol{\theta}_J) \\ &= \sum_{k=0}^{N_{\mathrm{p,c}}-1} \sum_{\ell=0}^{m_1-1} L(\hat{\boldsymbol{x}}(m_1(m_2k_{\mathrm{p}}+k)+\ell), \boldsymbol{u}_{\mathrm{c}}(m_2k_{\mathrm{p}}+k), \boldsymbol{\theta}_J) \\ &\quad + T(\hat{\boldsymbol{x}}(m_1m_2k_{\mathrm{p}} + m_1N_{\mathrm{p,c}}), \boldsymbol{\theta}_J), \end{aligned} \tag{17}$$

where $L(\cdot)$ and $T(\cdot)$ are the stage cost function and terminal cost function parameterized by $\boldsymbol{\theta}_J$. One possible realization of the parameterized stage cost $L(\cdot)$ and terminal cost $T(\cdot)$ can be similar to (16), i.e., a combination of basis functions weighted by the parameters. In addition to the methods presented in Section 3.2.3 for constructing the basis functions, radial basis functions (RBFs) can also be considered since they have the universal approximation property (Micchelli, 1984; Park and Sandberg, 1991). The selection of the centers and weights of the RBFs can be done by the RL agent within this framework, where these parameters can be integrated into $\boldsymbol{\theta}_J$, and the action of the RL agent can be the same as (11).

The terminal constraint set is important in MPC theory to guarantee the recursive feasibility and it can be adapted online (Rosolia and Borrelli, 2017). However, Rosolia and Borrelli (2017) did not consider the changing environment or disturbances. In this case, all the constraint sets are integrated in the function $\mathcal{G}(\hat{\boldsymbol{x}}, \boldsymbol{u}, \boldsymbol{\theta}_{\mathcal{G}})$, which is parameterized by $\boldsymbol{\theta}_{\mathcal{G}}$. Therefore, $\boldsymbol{\theta}_{\mathcal{G}}$ can be tuned by the RL agent to respond to the varying environmental conditions. Furthermore, the state space and reward function for this case can be defined as in Case A.

3.2.5. Case D: RL modifying the optimization settings

In this section, we propose to use the RL agent to modify the optimization settings, such as the length of the prediction horizon as in Piga et al. (2019) and the optimization options of a solver. Proper tuning of the prediction horizon can result in a balance between computational complexity and performance, so as the solver options. Different optimization algorithms may lead to various results, and for each algorithm, the optimization settings such as the constraint tolerance, step size tolerance, function value tolerance, and other parameters, which significantly influence the optimization speed and accuracy, should be pre-selected. Within our proposed framework, these parameter values are allowed to be adapted according to the varying environment and control objective. The reward function of the RL agent for this case should be revised, for instance to be defined as a combination of the control performance and the computational efficiency, which is given by:

$$r_{k_{\mathrm{rl}}}(s_{k_{\mathrm{rl}}}, a_{k_{\mathrm{rl}}}) = -R(\bar{\boldsymbol{x}}(k_{\mathrm{rl}}), \bar{\boldsymbol{u}}(k_{\mathrm{rl}})) - J_C(a_{k_{\mathrm{rl}}}), \tag{18}$$

where $J_C(\cdot)$ is an index that denotes the computational efficiency of the solver (e.g., the computation time for solving the optimization problem). Accordingly, the state space of the RL agent can be defined as in Case A, and the action can be given by (11).

As mentioned in Remark 1, the optimization variables $\boldsymbol{u}_\theta$ can be changed in a move blocking way. Let N_{b} denote the number of the PMPC operation steps where the parameters remain constant within the prediction window. If $N_{\mathrm{b}} = N_{\mathrm{p,o}}$, then $\boldsymbol{u}_\theta$ is constant over the prediction window. In this case, the computation time is reduced, but will in general result in less optimal performance. Therefore, N_{b} can also be a parameter that is tuned by the RL agent to reach a trade-off between the performance and the computation time.

3.2.6. Case E: RL modifying parameterized control inputs

This is a degenerate case with the PMPC module, in which the RL agent is used to tune $\boldsymbol{u}_\theta$ generated by the optimization process of the PMPC module. This is different from conventional studies (Zhang et al., 2021; Remmerswaal et al., 2022; Sun et al., 2024), in which the RL agent directly adjusts the control inputs that are fed into the system. Considering a simple case for (8), where the parameterized control law is a linear function:

$$\boldsymbol{u}_{\mathrm{c}}(k_{\mathrm{c}}) = \boldsymbol{u}_\theta \hat{\boldsymbol{x}}(m_1 k_{\mathrm{c}}), \tag{19}$$

the corresponding RL action is

$$a_{k_{\mathrm{rl}}} = \Delta \boldsymbol{u}_\theta, \tag{20}$$

where $\Delta \boldsymbol{u}_\theta$ is the adjustment value to $\boldsymbol{u}_\theta$. Compared to adjusting $\boldsymbol{u}_{\mathrm{c}}$ or determining $\boldsymbol{u}_\theta$ directly, tuning the parameters $\Delta \boldsymbol{u}_\theta$ is expected to be more robust in terms of the control performance during the learning process. In particular, without the MPC scheme (i.e., the receding horizon optimization process), the framework will be reduced to an RL-based adaptive state feedback controller, as in Sun et al. (2023). The state space and reward function can be defined as in Case A.

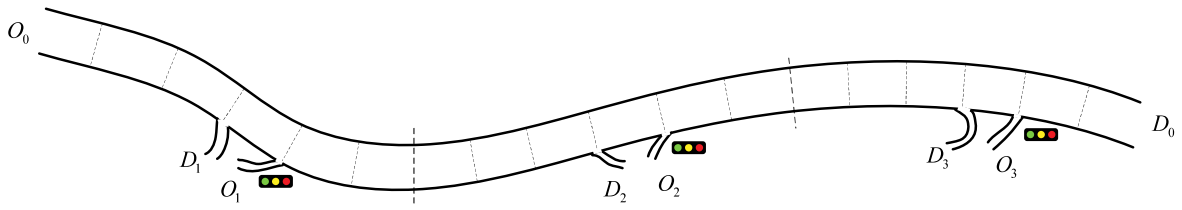

Figure 3: The layout of the freeway network in this case study

4. Case study

In this section, Case B from Section 3.2.3 is implemented on a freeway network, in order to illustrate the proposed framework. First, the freeway network is presented, followed by the traffic demand profiles, external disturbances, and weather conditions that introduce parameters uncertainties. Then the parameterized control law based on ramp metering (RM) is introduced. A DQN agent is utilized in the proposed framework to tune the parameters of the parameterized RM control law. The performance of the proposed framework is compared with conventional MPC, PMPC, and a standalone RL controller[3].

4.1. Freeway network

The benchmark freeway network from Liu et al. (2022) is used in this case study, which is presented in Figure 3. This freeway network is divided into 18 segments of 1000 m long. There are 1 mainstream origin (O_0), 3 on-ramps (O_1, O_2, O_3), 1 unrestricted destination (D_0), and 3 unrestricted off-ramps (D_1, D_2, D_3). All three on-ramps are regulated by a traffic light, which can control the ramp metering rate (i.e., ramp metering (RM) control). Therefore, there are 3 control signals in total for this network. In this case study, METANET is used to represent the freeway network, and the perturbed version of the same model is used as the prediction model for PMPC. Therefore, both the controlled system and the prediction model have the same simulation sampling time. METANET is a second-order macroscopic traffic flow model that has been widely used in freeway traffic control (Hegyi et al., 2005; Liu et al., 2022) thanks to its ability to reproduce freeway traffic phenomena with relatively less computational complexity. More details about METANET can be found in (Messner and Papageorgiou, 1990; Kotsialos et al., 2002).

In this case study, a scenario of recurrent traffic demand for 2 hours during the rush hour is considered. Traffic demands from all origins (O_0, O_1, O_2, O_3) are presented in Figure 4. In addition, a shock wave from the downstream boundary is generated to produce extra traffic jams. Such a shock wave is an abrupt increase in traffic density that will propagate from downstream to upstream. The downstream density profile is presented in Figure 5. The parameters of the freeway model are taken from Liu et al. (2022). In this case study, environment changes are considered that can influence the parameters of the traffic network. More specifically, the weather condition is taken into consideration, which is classified into three levels: good weather, bad weather, and extreme weather. The real parameters of the freeway model and the estimated parameters of the prediction model are presented in Table 2, for different weather conditions. The mathematical notations of the parameters are the same as in (Liu et al., 2022). For definitions of the parameters, the reader can refer to Hegyi et al. (2005); Liu et al. (2022).

Six weather scenarios are considered, each of which corresponds to a 2-hour simulation interval where the weather condition remains unchanged for the first hour and switches to another condition for the next hour. The weather scenarios are defined as:

- Scenario 1: from good weather to bad weather;
- Scenario 2: from good weather to extreme weather;
- Scenario 3: from bad weather to extreme weather;
- Scenario 4: from bad weather to good weather;
- Scenario 5: from extreme weather to good weather.
- Scenario 6: from extreme weather to bad weather.

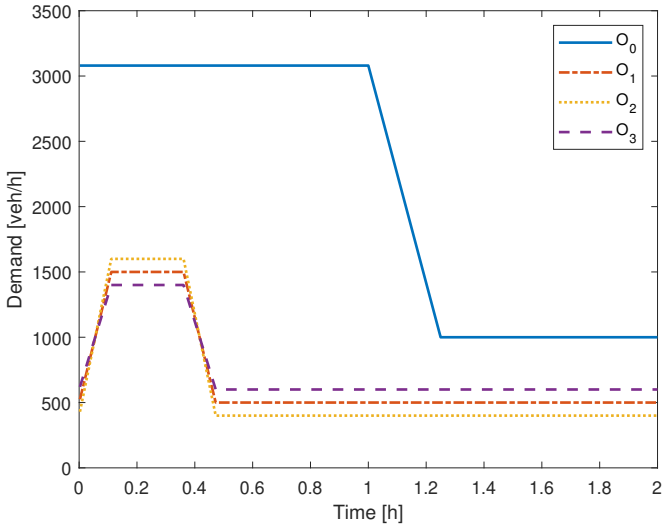

Figure 4: Traffic demand profiles for the origins of the freeway network

Table 2: Parameters of the freeway network under different weather conditions

	Weather condition	T[s]	τ[s]	κ[veh/km/lane]	η[km^2/h]	a_m	σ	v_{free}[km/h]	ρ_{crit}[veh/km/lane]	α	$\rho_{\max}$[veh/km/lane]	L_m[m]
Real	Good	10	18.0	40	65.0	1.867	0.10	102	33.5	0.10	180	1000
	Bad	10	27.0	60	55.0	1.667	0.30	92	26.5	0.00	150	900
	Extreme	10	36.0	80	45.0	1.467	0.50	82	19.5	-0.10	120	800
Estimated	Good	10	18.9	42	68.3	1.960	0.11	107	36.9	0.11	189	1050
	Bad	10	28.4	63	57.8	1.750	0.32	97	29.5	0.00	158	945
	Extreme	10	37.8	84	47.3	1.540	0.53	86	22.5	-0.11	126	840

Figure 6 gives an illustrative description of the scenarios. In this case study, all the controllers are implemented on these six scenarios and the performances of the resulting controlled systems are compared per scenario. Note that each simulation starts with a fixed initial state, which is obtained by starting with an empty freeway network and considering a constant demand of 3000 veh/h from the mainstream origin and 500 veh/h from the on-ramps for a period of 15 min; the state of the freeway network at the end of this period is used as the initial state for each of the simulations.

4.2. Parameterized freeway traffic control laws

Ramp metering (RM) rates have been widely used in freeway traffic management (Hegyi et al., 2005). This control measure was further parameterized by Zegeye et al. (2012) in an MPC framework. The parameterized control law of RM used in this case study is based on Zegeye et al. (2012), and is given by:

$$u_{\text{rm},i}\left(k_{\text{c}}+1\right) = u_{\text{rm},i}\left(k_{\text{c}}\right) + u_{\theta}(k_{\text{p}})\left(\theta_f(k_{\text{rl}}) - \rho_i(k_{\text{c}})\right), \tag{21}$$

where k_{c} is the control step, k_{p} is the PMPC operation time step, k_{rl} is the RL operation time step, $u_{\text{rm},i}\left(k_{\text{c}}\right)$ is the RM control input for the on-ramp that is linked to segment i, $\rho_i(k_{\text{c}})$ is the measured traffic density of the downstream segment i of the on-ramp, and $u_{\theta}(k_{\text{p}})$ is the parameter optimized by PMPC at operation step k_{p}. Note that the control law (21) is derived from ALINEA (Papageorgiou et al., 1991), in which the original parameter θ_f is the setpoint density obtained and pre-defined through experiments and historical data. In this case study, the same parameterized control law (21) is applied to all the three on-ramps, in which $\theta_f(k_{\text{rl}})$ is the parameter that is tuned by the RL agent at operation step k_{rl}.

[3]The source codes of the case study are available via the repository: `https://github.com/dingshansun/RL-based-PMPC`

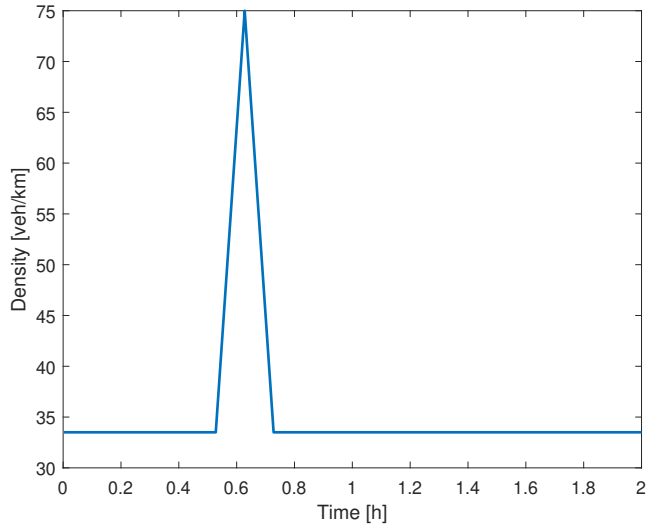

Figure 5: The downstream density used to generate a shock wave for the freeway network

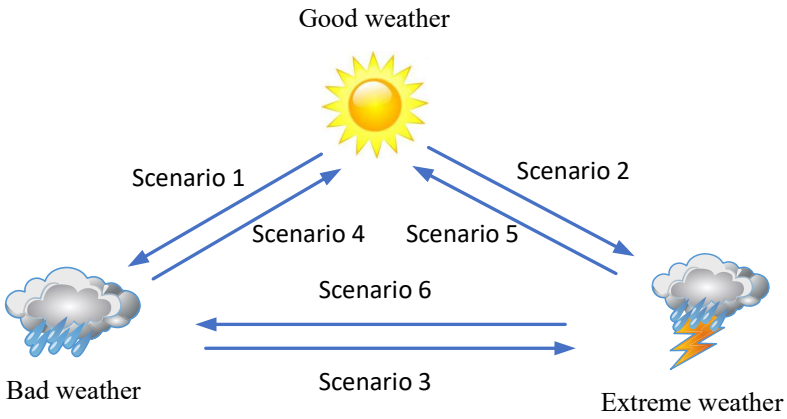

Figure 6: Illustrative depiction of the weather scenarios 1-6 considered in the case study

4.3. Controllers

All the MPC-based controllers in this case study use the prediction model with estimated parameters given in Table 2. In addition, the learning-based controllers (i.e., standalone RL controller and RL-PMPC controller) are trained off-line with the prediction model, and are validated via the system with real parameters. The simulations are performed on a PC with an Intel Xeon Quad-Core E5-1620 V3 CPU with a clock speed of 3.5 GHz.

In this case study, the total time spent (TTS) by all the vehicles in the entire freeway network for all the 6 weather scenarios is taken as the performance criterion for the controllers. For the METANET model, the simulation sampling time is $T_s = 10\,\text{s}$. The control sampling time for the parameterized control laws is $T_c = 60\,\text{s}$; the PMPC scheme operation sampling time is $T_p = 300\,\text{s}$. The control input constraints are given by:

$$0 \leq u_{\text{rm}} \leq 1.$$

Therefore, the RM control inputs generated by (21) should be saturated within the bounds. The projection method in Jeschke et al. (2023) is utilized to enforce the constraints on the control inputs, and it has been illustrated that the projection-based PMPC can achieve better or equal performance than the conventional PMPC with constraints.

Table 3: Training parameters of the DQN agent

Parameter	Value
Maximal episodes M	2000
Mini-batch size N	512
Experience replay buffer size	$1 \cdot 10^5$
Discounter factor γ	0.99
Learning rate	0.001
Target network update rate β	0.01
Initial ϵ value	1.0
ϵ decay rate	0.005
Minimum ϵ value	0.01

During the entire 2-hour simulation, it is assumed that the prediction model used by the (P)MPC controllers is fixed with the estimated parameters corresponding to the initial weather condition. Meanwhile, the parameter θ_f in the parameterized control law remains constant with the value of critical density ρ_{crit} of the initial weather condition for the standalone PMPC controller.

4.3.1. ALINEA controller

ALINEA (Papageorgiou et al., 1991) is a well-known state-feedback controller for freeway ramp metering control that aims at regulating the downstream density at the specified critical density ρ_{crit} to maximize the throughput. In this case study, it is used as a baseline controller. The ALINEA controller has the same expression as (21), but with fixed parameters:

$$u_{\text{rm},i}\left(k_{\text{c}}+1\right) = u_{\text{rm},i}\left(k_{\text{c}}\right) + c_{\text{rm}}\left(\rho_{\text{crit}} - \rho_i(k_{\text{c}})\right),$$

where c_{rm} is the fixed gain coefficient, which is taken as 0.1 in this case study by trial and error, and ρ_{crit} is the estimated critical density of the initial weather condition corresponding to the specified weather scenarios. The values of ρ_{crit} under the different weather conditions are given in Table 2.

4.3.2. Standalone PMPC controller

The objective function of the PMPC scheme only contains the TTS within the prediction window. The PMPC operation sampling time is 300 s, i.e., the parameterized optimization problem is solved every 300 s, while the parameterized control law (21) works on the basis of 60 s (i.e., provides control inputs according to the states every 60 s). The length of both the prediction horizon and the control horizon is 900 s. Thus, $N_{\text{p,s}} = 90$, $N_{\text{p,c}} = 15$, $N_{\text{p,o}} = 3$. Furthermore, $N_{\text{b}} = 1$, which means that the optimized parameters are allowed to change per PMPC operation step (i.e., every 300 s) within the prediction window. Therefore, the number of the optimization variables is $1 \times N_{\text{p,o}}/N_{\text{b}} = 3$.

The sequential quadratic programming (SQP) algorithm (Boggs and Tolle, 1995) is implemented via the `fmincon` function from Matlab to solve the nonlinear constrained optimization problem. To avoid getting stuck in local optima, multiple starting points are selected randomly to solve the optimization problem, and the best solution is taken as the final result. In this case study, the number of initial points is selected to be 40, which is determined according to the experiments, as in this way a balance is achieved between optimality and computational efficiency. In addition, the stopping criteria of the SQP algorithm are also tuned, in which the cost function tolerance, step tolerance, and constraint tolerance are all selected to be 10^{-2}.

4.3.3. Standalone MPC controller

A conventional standalone MPC controller is also implemented on the freeway network. It has the same settings as the PMPC controller, and has an operation sampling time of 60 s. Since the conventional MPC controller directly optimizes the ramp metering rates of the entire freeway network for every control step within the prediction window, it has a larger number of optimization variables than the PMPC controller, which is $3 \times N_{\text{p,c}} = 45$.

4.3.4. Standalone RL controller

The definition of the RL agent is similar to what has been explained in Section 3.2.1. More specifically, the state space of the RL agent consists of: the measured traffic state $\boldsymbol{x}$ at RL operation step k_{rl}, the traffic demands, downstream

boundary density, previously implemented ramp metering rates, and the real-time weather condition. The action of the RL agent consists of the ramp metering rates, which are given to the freeway network directly and have the same bound constraints as the PMPC controller. For simplicity, the three on-ramps share the same ramp metering rates, which is discretized into 11 values distributed equidistantly between 0 and 1. Therefore, the dimension of the state space is 46 and the dimension of the action space is 1. The operation sampling time for the standalone RL controller is the same as the network control sampling time, that is, $T_{\mathrm{rl}} = 60\,\mathrm{s}$. Thus the reward is defined as the negative value of the TTS during the simulation interval $[k_{\mathrm{rl}}T_{\mathrm{rl}}, (k_{\mathrm{rl}} + 1)T_{\mathrm{rl}})$ between two RL operation steps.

Accordingly, a deep Q-Network (DQN) (Mnih et al., 2013) agent is used, which can address the continuous state space and the discrete action space. The neural network consists of one input layer, one output layer, and three hidden layers. The size of the input and output layers correspond to the dimensions of state and action spaces, respectively. The three hidden inner layers have 64, 256, and 64 neurons[4], and each of them uses a ReLU activation function. The training parameters of the DQN agent are given in Table 3, in which ϵ is the exploration parameter decaying from the initial value to the minimum value with the decay rate. A higher value ϵ encourages more exploration. Thus, the agent has a high probability to choose actions randomly in the early learning stage, and the probability decreases gradually as the training procedure evolves. Similarly to Sun et al. (2024), $n-$step TD (temporal difference) is also used in this DQN agent to improve the learning and control performance, with $n = 15$.

4.3.5. RL-PMPC control framework

The RL-PMPC control framework consists of a PMPC controller and an RL agent. The PMPC controller is the same as the standalone PMPC controller defined in Section 4.3.2. The DQN agent defined in Section 4.3.4 is also used in this control framework. In addition to the state space of the standalone RL controller, the agent in this framework has extra state variables, i.e., the PMPC input $u_\theta(k_{\mathrm{p}})$. Furthermore, the action space is also different. The RL agent within the framework tunes the parameters $\theta_f(k_{\mathrm{rl}})$ of the control law (16) at every RL operation step k_{rl}. Based on numerical tuning experiments, the range of the parameter θ_f is set from 15 veh/km/lane to 40 veh/km/lane, and the action space is discretized into 11 actions distributed equidistantly within this range. The parameter selected by the RL agent is applied to all the on-ramps.

The time complexity of the framework depends on the number of the optimization variables of the PMPC module. Since PMPC is an optimization-based controller, in the general case, which includes nonlinear nonconvex optimization, the computation time in practice would grow exponentially with the increasing number of optimization variables. Nevertheless, the computation time of the framework has been significantly reduced compared to conventional MPC methods, because of the fact that the number of optimization variables of PMPC is greatly reduced compared to conventional MPC method (e.g., from 45 to 3 in this case study).

Since the aim of the RL agent within this framework is to deal with the changing environment (i.e., the changing weather conditions), this RL agent has an operation sampling time that is in line with the weather-changing frequency. In this case study, we have $T_{\mathrm{rl}} = 1800\,\mathrm{s}$. Consequently, the reward at each RL step k_{rl} is the negative value of the TTS during the simulation interval $[k_{\mathrm{rl}}T_{\mathrm{rl}}, (k_{\mathrm{rl}} + 1)T_{\mathrm{rl}})$.

4.4. Results and discussions

The TTS and CPU time results for each controller per scenario are collected, and the results are presented in Table 4, in which the mean CPU time is the average time required for the optimization process per control step, and the max CPU time corresponds to the maximum over all the control steps of the computation time per step.

Table 4 shows that all the controllers improve the performance in terms of TTS with regard to the no-control case. More specifically, the standalone MPC controller provides the best TTS performance among all the controllers for most scenarios (1-4). This is because the standalone MPC controller optimizes the control inputs directly for each on-ramp and because it has a higher operation frequency. Therefore, MPC can adapt to the changing environment and disturbances implicitly by measuring the real-time traffic states, and can thus provide the optimal control inputs. In contrast, ALINEA also regulates each ramp separately, but performs worse than the other controllers. This is because ALINEA is vulnerable to the estimated network parameter uncertainties, and the local ALINEA controllers cannot be coordinated to optimize the global performance. However, the standalone MPC controller results in the

[4]These numbers have been selected by manual tuning.

Table 4: Comparison of the control performance for different controllers for different weather scenarios, in terms of TTS, mean computation time, and max computation time, in which '-' means that the corresponding item is not applicable to the controller.

Weather scenario	Performance	No control	ALINEA	Standalone MPC	Standalone PMPC	Standalone RL	RL-PMPC
Scenario 1	TTS [veh · h]	4819.15	4040.13	3911.36	4062.56	4003.01	3985.90
	Mean CPU time [s]	-	-	73.46	1.08	-	0.94
	Max CPU time [s]	-	-	104.07	2.03	-	2.24
Scenario 2	TTS [veh · h]	5426.99	4783.06	4645.06	4732.41	4992.53	4697.27
	Mean CPU time [s]	-	-	69.42	1.05	-	1.05
	Max CPU time [s]	-	-	119.64	2.10	-	2.17
Scenario 3	TTS [veh · h]	7932.46	6982.50	6804.84	7073.85	6938.64	6877.34
	Mean CPU time [s]	-	-	74.43	1.29	-	0.83
	Max CPU time [s]	-	-	104.60	2.93	-	3.09
Scenario 4	TTS [veh · h]	6652.77	5446.39	5353.94	5525.41	6084.34	5413.88
	Mean CPU time [s]	-	-	76.69	1.49	-	1.18
	Max CPU time [s]	-	-	116.08	2.88	-	2.62
Scenario 5	TTS [veh · h]	8362.71	7356.78	7305.03	7559.68	7548.57	7293.49
	Mean CPU time [s]	-	-	50.50	1.61	-	1.47
	Max CPU time [s]	-	-	89.82	4.04	-	3.06
Scenario 6	TTS [veh · h]	9266.04	7917.86	8357.42	8474.34	8755.09	7961.16
	Mean CPU time [s]	-	-	52.87	1.87	-	2.02
	Max CPU time [s]	-	-	84.48	4.10	-	4.15

largest computation time for all the scenarios, in terms of both mean and maximum computation time. In comparison, the standalone PMPC controller significantly reduces the computation time by 98% on average with regard to the standalone MPC controller, and it provides comparable control performance to the standalone MPC controller in most scenarios (1-4) and even achieves better performance in scenario 5 and 6.[5] The computation time of the standalone RL controller is negligible since only an online neural network evaluation is required to obtain the control inputs. Nevertheless, the standalone RL controller is sensitive to the model mismatches between the prediction model (i.e., the training model) and the real system (i.e., the validation model). Therefore, even when the RL agent is trained with a sufficient number of data samples, the validation performance still cannot be guaranteed (see Scenarios 2, 4 and 6 in Table 4).

In contrast, the RL-PMPC controller achieves a better performance than both the standalone PMPC and the standalone RL controllers. Through this ablation study, the effectiveness of the proposed framework is validated. With the PMPC module, the RL-PMPC framework can guarantee a basic performance. With the RL module, the RL-PMPC framework can further improve the control performance of the PMPC module by online tuning of the parameters of the parameterized control laws. The RL-PMPC framework can also adapt better to the model mismatches and the changing environment (i.e., changing weather conditions). Furthermore, the RL-PMPC controller inherits the computational efficiency advantage of PMPC and RL. Therefore, the RL-PMPC controller has a significantly reduced online computation time compared to the standalone MPC controller, and meanwhile provides a TTS performance that is comparable to the standalone MPC controller for all the considered scenarios. One additional advantage of RL-PMPC is that the action space of the RL module is reduced with regard to the standalone RL agent, since only the parameters of the parameterized control laws are tuned. The number of the parameters in the parameterized control laws is usually smaller than the number of the control inputs, which therefore makes it easier for the RL agent to explore the environment and learn the optimal policy. Note that in this case study, the number of the parameters in the parameterized control law is 1 and the number of the control inputs is 3.

Remark 3. *In this case study, we only consider one freeway control measure (i.e., ramp metering) to illustrate the concept of the RL-PMPC framework. If we introduce extra freeway control measures (e.g., variable speed limits),*

[5]The performance of standalone MPC deteriorates for scenarios 5 and 6, and this is because standalone MPC cannot handle the large model mismatches under the extreme weather condition without directly adapting the model parameters.

the RL-PMPC controller has more freedom to tune the corresponding parameters, which is hypothesized to further improve the control performance with regard to the standalone PMPC controller.

5. Conclusions and topics for future research

This paper has proposed a novel synthesis framework for PMPC that integrates an extended PMPC scheme and an RL agent, in order to deal with changing environments and disturbances. The resulting RL-based adaptive PMPC (RL-PMPC) framework is not only computationally efficient, but it can also adapt to model mismatches and environmental uncertainties. The novel framework allows to adjust multiple components of the PMPC scheme by the RL agent, thus providing more flexibility to deal with uncertainties. Five cases of the synthesis framework have been presented corresponding to adjusting different components of the PMPC scheme. The framework embeds existing adaptive MPC methods, and further broadens adaptive MPC by proposing several new adaption strategies. We have illustrated the operation of the RL-PMPC scheme via a simulation-based case study for a freeway network that suffers from model mismatches and changing weather conditions. The simulation results show that the proposed RL-based adaptive PMPC framework outperforms the standalone PMPC and the standalone RL controllers in terms of total time spent, and can provide comparable control performance to the conventional MPC controller with a significantly reduced computation time, under a changing environment and in the presence of disturbances.

Future research can be conducted to address the scalability issue of the proposed framework, since the computational complexity will in general increase rapidly with the size of the networks. One potential solution is to extend the framework to a multi-agent variant by integrating distributed PMPC and multi-agent RL and dividing the large network into several smaller sub-networks. In addition, the stability and recursive feasibility of the novel control framework can be investigated, and the performance can be compared with existing robust MPC methods. Furthermore, more advanced optimization algorithms can be considered, such as the adaptive polyploid memetic algorithm (Dulebenets, 2021) and the diffused memetic optimization method (Dulebenets, 2023) to address disruptions, the fast fireworks algorithm method of Chen and Tan (2023) to deal with a large-scale optimization problem, the heuristic algorithm of Singh et al. (2022) to address multiple objectives, or the ant-based optimization techniques introduced in Singh and Pillay (2022). It would also be an interesting topic to apply and extend all these methods to other application fields next to traffic management, such as energy management, smart buildings, and health. Moreover, A comparison study can be carried out between the proposed method and the algorithms from the references.

References

Adetola, V., DeHaan, D., Guay, M., 2009. Adaptive model predictive control for constrained nonlinear systems. Systems & Control Letters 58, 320–326.

Akpan, V.A., Hassapis, G.D., 2011. Nonlinear model identification and adaptive model predictive control using neural networks. ISA Transactions 50, 177–194.

Alessio, A., Bemporad, A., 2009. A survey on explicit model predictive control, in: Magni, L. (Ed.), Nonlinear Model Predictive Control: Towards New Challenging Applications. Springer-Verlag, Berlin Heidelberg, pp. 345–369.

Arroyo, J., Manna, C., Spiessens, F., Helsen, L., 2022. Reinforced model predictive control (RL-MPC) for building energy management. Applied Energy 309, 118346.

Arulkumaran, K., Deisenroth, M.P., Brundage, M., Bharath, A.A., 2017. Deep reinforcement learning: A brief survey. IEEE Signal Processing Magazine 34, 26–38.

Bansal, S., Calandra, R., Xiao, T., Levine, S., Tomlin, C.J., 2017. Goal-driven dynamics learning via bayesian optimization, in: 2017 IEEE 56th Annual Conference on Decision and Control (CDC), IEEE. pp. 5168–5173.

Bemporad, A., Morari, M., 2007. Robust model predictive control: A survey, in: Robustness in Identification and Control. Springer, pp. 207–226.

Boggs, P.T., Tolle, J.W., 1995. Sequential quadratic programming. Acta Numerica 4, 1–51.

Brunner, F.D., Lazar, M., Allgöwer, F., 2015. Stabilizing model predictive control: On the enlargement of the terminal set. International Journal of Robust and Nonlinear Control 25, 2646–2670.

Cagienard, R., Grieder, P., Kerrigan, E.C., Morari, M., 2007. Move blocking strategies in receding horizon control. Journal of Process Control 17, 563–570.

Camacho, E.F., Alba, C.B., 2013. Model Predictive Control. Springer Science & Business Media.

Chen, M., Tan, Y., 2023. SF-FWA: A self-adaptive fast fireworks algorithm for effective large-scale optimization. Swarm and Evolutionary Computation 80, 101314.

Dulac-Arnold, G., Levine, N., Mankowitz, D.J., Li, J., Paduraru, C., Gowal, S., Hester, T., 2021. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. Machine Learning 110, 2419–2468.

Dulebenets, M.A., 2021. An adaptive polyploid memetic algorithm for scheduling trucks at a cross-docking terminal. Information Sciences 565, 390–421.

Dulebenets, M.A., 2023. A diffused memetic optimizer for reactive berth allocation and scheduling at marine container terminals in response to disruptions. Swarm and Evolutionary Computation 80, 101334.
Görges, D., 2017. Relations between model predictive control and reinforcement learning. IFAC-PapersOnLine 50, 4920–4928.
Goulart, P.J., Kerrigan, E.C., Maciejowski, J.M., 2006. Optimization over state feedback policies for robust control with constraints. Automatica 42, 523–533.
Gros, S., Zanon, M., 2019. Data-driven economic NMPC using reinforcement learning. IEEE Transactions on Automatic Control 65, 636–648.
Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al., 2018. Soft actor-critic algorithms and applications. arXiv preprint arXiv:1812.05905 .
Hegyi, A., De Schutter, B., Hellendoorn, H., 2005. Model predictive control for optimal coordination of ramp metering and variable speed limits. Transportation Research Part C: Emerging Technologies 13, 185–209.
Heirung, T.A.N., Ydstie, B.E., Foss, B., 2017. Dual adaptive model predictive control. Automatica 80, 340–348.
Hewing, L., Wabersich, K.P., Menner, M., Zeilinger, M.N., 2020. Learning-based model predictive control: Toward safe learning in control. Annual Review of Control, Robotics, and Autonomous Systems 3, 269–296.
Hosseini, M.M., Rodriguez-Garcia, L., Parvania, M., 2023. Hierarchical combination of deep reinforcement learning and quadratic programming for distribution system restoration. IEEE Transactions on Sustainable Energy 14, 1088–1098.
Hu, J., Wang, Q., Ye, Y., Tang, Y., 2022. Toward online power system model identification: A deep reinforcement learning approach. IEEE Transactions on Power Systems .
Jeschke, J., De Schutter, B., 2021. Parametrized model predictive control approaches for urban traffic networks. IFAC-PapersOnLine 54, 284–291.
Jeschke, J., Sun, D., Jamshidnejad, A., De Schutter, B., 2023. Grammatical-evolution-based parameterized model predictive control for urban traffic networks. Control Engineering Practice 132, 105431.
Jiang, B., Chen, S., Wang, B., Luo, B., 2022. Mglnn: Semi-supervised learning via multiple graph cooperative learning neural networks. Neural Networks 153, 204–214.
Karamanakos, P., Geyer, T., Kennel, R., 2015. A computationally efficient model predictive control strategy for linear systems with integer inputs. IEEE Transactions on Control Systems Technology 24, 1463–1471.
Köhler, J., Kötting, P., Soloperto, R., Allgöwer, F., Müller, M.A., 2021. A robust adaptive model predictive control framework for nonlinear uncertain systems. International Journal of Robust and Nonlinear Control 31, 8725–8749.
Kotsialos, A., Papageorgiou, M., Diakaki, C., Pavlis, Y., Middelham, F., 2002. Traffic flow modeling of large-scale motorway networks using the macroscopic modeling tool METANET. IEEE Transactions on Intelligent Transportation Systems 3, 282–292.
Kumpati, S.N., Kannan, P., et al., 1990. Identification and control of dynamical systems using neural networks. IEEE Transactions on Neural Networks 1, 4–27.
Li, Y., Hua, K., Cao, Y., 2022. Using stochastic programming to train neural network approximation of nonlinear MPC laws. Automatica 146, 110665.
Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D., 2015. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 .
Liu, S., Sadowska, A., De Schutter, B., 2022. A scenario-based distributed model predictive control approach for freeway networks. Transportation Research Part C: Emerging Technologies 136, 103261.
Lofberg, J., 2003. Approximations of closed-loop minimax MPC, in: 42nd IEEE International Conference on Decision and Control, IEEE. pp. 1438–1442.
Lorenzen, M., Allgöwer, F., Cannon, M., 2017. Adaptive model predictive control with robust constraint satisfaction. IFAC-PapersOnLine 50, 3313–3318.
Marco, A., Hennig, P., Bohg, J., Schaal, S., Trimpe, S., 2016. Automatic LQR tuning based on Gaussian process global optimization, in: 2016 IEEE International Conference on Robotics and Automation (ICRA), IEEE. pp. 270–277.
Mesbah, A., 2016. Stochastic model predictive control: An overview and perspectives for future research. IEEE Control Systems Magazine 36, 30–44.
Messner, A., Papageorgiou, M., 1990. METANET: A macroscopic simulation program for motorway networks. Traffic Engineering & Control 31, 466–470.
Micchelli, C.A., 1984. Interpolation of scattered data: Distance matrices and conditionally positive definite functions. Springer.
Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K., 2016. Asynchronous methods for deep reinforcement learning, in: International conference on machine learning, PMLR. pp. 1928–1937.
Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M., 2013. Playing Atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 .
Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al., 2015. Human-level control through deep reinforcement learning. Nature 518, 529–533.
Morari, M., Lee, J.H., 1999. Model predictive control: Past, present and future. Computers & Chemical Engineering 23, 667–682.
Negenborn, R.R., De Schutter, B., Wiering, M.A., Hellendoorn, H., 2005. Learning-based model predictive control for Markov decision processes. IFAC Proceedings Volumes 38, 354–359.
Pannocchia, G., Rawlings, J.B., Wright, S.J., 2011. Conditions under which suboptimal nonlinear MPC is inherently robust. Systems & Control Letters 60, 747–755.
Papageorgiou, M., Hadj-Salem, H., Blosseville, J.M., et al., 1991. ALINEA: A local feedback control law for on-ramp metering. Transportation Research Record 1320, 58–67.
Park, J., Sandberg, I.W., 1991. Universal approximation using radial-basis-function networks. Neural Computation 3, 246–257.
Perrusquía, A., Yu, W., 2021. Identification and optimal control of nonlinear systems using recurrent neural networks and reinforcement learning: An overview. Neurocomputing 438, 145–154.
Piga, D., Forgione, M., Formentin, S., Bemporad, A., 2019. Performance-oriented model learning for data-driven MPC design. IEEE Control Systems Letters 3, 577–582.

Pippia, T., Sijs, J., De Schutter, B., 2018. A parametrized model predictive control approach for microgrids, in: 2018 IEEE Conference on Decision and Control (CDC), IEEE. pp. 3171–3176.

Qin, S.J., Badgwell, T.A., 2003. A survey of industrial model predictive control technology. Control Engineering Practice 11, 733–764.

Raković, S.V., Kouvaritakis, B., Findeisen, R., Cannon, M., 2012. Homothetic tube model predictive control. Automatica 48, 1631–1638.

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

Rosolia, U., Borrelli, F., 2017. Learning model predictive control for iterative tasks: A data-driven control framework. IEEE Transactions on Automatic Control 63, 1883–1896.

Rosolia, U., Zhang, X., Borrelli, F., 2017. Robust learning model predictive control for iterative tasks: Learning from experience, in: 2017 IEEE 56th Annual Conference on Decision and Control (CDC), IEEE. pp. 1157–1162.

Singh, A., Raj, K., Kumar, T., Verma, S., Roy, A.M., 2023. Deep learning-based cost-effective and responsive robot for autism treatment. Drones 7, 81.

Singh, E., Pillay, N., 2022. A study of ant-based pheromone spaces for generation constructive hyper-heuristics. Swarm and Evolutionary Computation 72, 101095.

Singh, P., Pasha, J., Moses, R., Sobanjo, J., Ozguven, E.E., Dulebenets, M.A., 2022. Development of exact and heuristic optimization methods for safety improvement projects at level crossings under conflicting objectives. Reliability Engineering & System Safety 220, 108296.

Sun, D., Jamshidnejad, A., De Schutter, B., 2023. Adaptive parameterized control for coordinated traffic management using reinforcement learning. IFAC-PapersOnLine 56, 5463–5468.

Sun, D., Jamshidnejad, A., De Schutter, B., 2024. A novel framework combining mpc and deep reinforcement learning with application to freeway traffic control. IEEE Transactions on Intelligent Transportation Systems .

Sutton, R.S., Barto, A.G., 2018. Reinforcement learning: An introduction. MIT press.

Tanaskovic, M., Fagiano, L., Gligorovski, V., 2019. Adaptive model predictive control for linear time varying MIMO systems. Automatica 105, 237–245.

Tóth, R., 2010. Modeling and identification of linear parameter-varying systems. Springer–Verlag, Berlin, Heidelberg.

Van Kooten, R., Imhof, P., Brummelhuis, K., Van Pampus, M., Jamshidnejad, A., De Schutter, B., 2017. ART-UTC: An adaptive real-time urban traffic control strategy, in: 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), IEEE. pp. 1–6.

Wang, S., Diao, R., Xu, C., Shi, D., Wang, Z., 2020. On multi-event co-calibration of dynamic model parameters using soft actor-critic. IEEE Transactions on Power Systems 36, 521–524.

Zegeye, S.K., De Schutter, B., Hellendoorn, J., Breunesse, E.A., Hegyi, A., 2012. A predictive traffic controller for sustainable mobility using parameterized control policies. IEEE Transactions on Intelligent Transportation Systems 13, 1420–1429.

Zhang, H., Li, S., Zheng, Y., 2020. Q-learning-based model predictive control for nonlinear continuous-time systems. Industrial & Engineering Chemistry Research 59, 17987–17999.

Zhang, K., Shi, Y., 2020. Adaptive model predictive control for a class of constrained linear systems with parametric uncertainties. Automatica 117, 108974.

Zhang, Q., Pan, W., Reppa, V., 2021. Model-reference reinforcement learning for collision-free tracking control of autonomous surface vehicles. IEEE Transactions on Intelligent Transportation Systems 23, 8770–8781.

Zhong, M., Johnson, M., Tassa, Y., Erez, T., Todorov, E., 2013. Value function approximation and model predictive control, in: 2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL), IEEE. pp. 100–107.