Combined MPC and reinforcement learning for traffic signal control in urban traffic networks

Remmerswaal, Willemijn; Sun, Dingshan; Jamshidnejad, Anahita; De Schutter, Bart

DOI
10.1109/ICSTCC55426.2022.9931771

Publication date
2022

Document Version
Final published version

Published in
Proceedings of the 26th International Conference on System Theory, Control and Computing, ICSTCC 2022

Citation (APA)
Remmerswaal, W., Sun, D., Jamshidnejad, A., & De Schutter, B. (2022). Combined MPC and reinforcement learning for traffic signal control in urban traffic networks. In M. Barbu, & R. Solea (Eds.), *Proceedings of the 26th International Conference on System Theory, Control and Computing, ICSTCC 2022* (pp. 432-439). IEEE. https://doi.org/10.1109/ICSTCC55426.2022.9931771

Combined MPC and reinforcement learning for traffic signal control in urban traffic networks*

Willemijn Remmerswaal[1], Dingshan Sun[1,**], Anahita Jamshidnejad[2] and Bart De Schutter,[1] *Fellow, IEEE*

***Abstract*— In general, the performance of model-based controllers cannot be guaranteed under model uncertainties or disturbances, while learning-based controllers require an extensively sufficient training process to perform well. These issues especially hold for large-scale nonlinear systems such as urban traffic networks. In this paper, a new framework is proposed by combining model predictive control (MPC) and reinforcement learning (RL) to provide desired performance for urban traffic networks even during the learning process, despite model uncertainties and disturbances. MPC and RL complement each other very well, since MPC provides a sub-optimal and constraint-satisfying control input while RL provides adaptive control laws and can handle uncertainties and disturbances. The resulting combined framework is applied for traffic signal control (TSC) of an urban traffic network. A case study is carried out to compare the performance of the proposed framework and other baseline controllers. Results show that the proposed combined framework outperforms conventional control methods under system uncertainties, in terms of reducing traffic congestion.**

2022 26th International Conference on System Theory, Control and Computing (ICSTCC) | 978-1-6654-6746-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICSTCC55426.2022.9931771

I. Introduction

Traffic congestion has a negative impact in terms of environmental, economic, and societal effects. Through congestion, the emission of greenhouse gasses increases, as cars will spend more time on the road and often have non-smooth deceleration and acceleration patterns [1]. This last issue is especially appearing in urban areas where cars are regularly slowed down by traffic lights. In 2019, 19% of the emission of greenhouse gasses in the Netherlands was due to road traffic [2]. In the EU, 40% of the CO_2 emission and 70% of other pollutants caused by road traffic are due to traffic in urban areas [3]. These issues can majorly be resolved by efficient traffic management strategies [1].

In order to minimize traffic congestion and thus its negative effects, proper control strategies should be designed and utilized for optimal use of the available infrastructure. The most used control measure for urban traffic is via traffic signal control (TSC) [4]. Nowadays with the increasing traffic demands, the inappropriate utilization of traffic signals may cause urban traffic congestion [5]. The earlier controllers for urban traffic signal control are fixed-time, i.e. their control strategy is determined offline based on historical data. More efficient strategies include traffic-responsive controllers, which are able to react to traffic situations as they use real-time data. Traffic-responsive controllers also include model-based controllers, one of which is MPC [6].

MPC controllers have been used for TSC by De Schutter and De Moor [7], where they are shown to be very effective, due to their flexible nature and their capability of handling constraints explicitly. However, model-based control methods heavily rely on the accuracy of the nominal model, and this also holds for MPC. The performance of MPC deteriorates if model uncertainties or external disturbances are present. This issue can be relieved by using a more accurate model, or by using robust or stochastic MPC. However, this significantly increases the online computational complexity, and can make the problem of controlling traffic networks intractable in real time [8], [9]. In recent years the use of data-driven reinforcement learning (RL) for urban traffic control has gained lots of interest [10]. While model-free RL algorithms do not need a model to obtain a well-performing control law, they are adaptive, i.e., they change the control inputs according to changing traffic situations. This makes RL-algorithms capable of dealing with disturbances. However, RL algorithms require a sufficiently extensive training process, and they lack performance guarantees, especially during the learning process.

Based on these facts, this paper contributes to the state-of-the-art by: 1) for the first time proposing a combined MPC-RL framework such that the advantages of both methods are exploited; 2) improving the performance of MPC under model uncertainty or external disturbances; 3) sustainably giving acceptable performance compared with the standalone RL-based controller, even during the learning process.

The paper is organized as follows. Section II introduces the relevant background knowledge, including MPC-based and RL-based TSC. The proposed control method that combines MPC and RL in one overall framework is presented in Section III, followed by a case study in Section IV. Finally, Section V concludes and closes the paper.

II. Background

In this section, we introduce MPC and RL-based signal control for urban traffic networks, as well as the model-reference framework that has inspired the authors.

A. MPC for urban traffic signal control

In the current paper, MPC is used as a baseline controller to provide the essential control inputs. The control sampling time T_c of MPC is considered to be the cycle time T_{cyc} of the intersections in the traffic network, which, for the sake of simplicity are assumed to be identical and fixed to 1 min. The mathematical model for urban traffic networks used in this paper as the MPC prediction model is the BLX model

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

[1]Willemijn Remmerswaal, Dingshan Sun and Bart De Schutter are with the Delft Center for Systems and Control, Delft University of Technology, The Netherlands.

[2]Anahita Jamshidnejad is with the Department of Control and Operations, Faculty of Aerospace Engineering, Delft University of Technology, The Netherlands.

**The corresponding author is Dingshan Sun: d.sun-1@tudelft.nl

[11]. Note that the proposed framework in this paper also applies to other traffic models.

The control inputs determined by MPC are the green time percentages $\pi_{\text{green},d}$, expressed as a percentage of the cycle time for all intersections $d \in J$ where J is the set of all the controlled intersections in the network. As the network considered in this paper for illustration purposes uses a two-phased cycle, the green time percentage consists of one variable per cycle per intersection. The cycle counter is denoted as $k_c \in \{1, ..., N_{cyc}\}$, where N_{cyc} is the total number of cycles within the simulation window. An optimization problem is solved by MPC at every controller time step k_c over a prediction window of length N_p (called the prediction horizon) to determine the optimal control inputs. We have

$$\min_{\boldsymbol{u}_{N_p}(k_c)} \sum_{k=0}^{N_p-1} \sum_{(u,d)\in L} \text{TTS}_{(u,d)}(k_c+k+1), \tag{1}$$

$$\text{s.t. } x(k_c+k+1) = f(x(k_c+k), u(k_c+k)), \tag{2}$$

$$u_{\min}(k_c) \le \boldsymbol{u}_{N_p}(k_c) \le u_{\max}, \tag{3}$$

where $\boldsymbol{u}_{N_p}(k_c) = [\boldsymbol{u}^\top(k_c), \ldots, \boldsymbol{u}^\top(k_c+N_p-1)]^\top$, and where $\boldsymbol{u}(k_c)$ contains the green time percentage $\pi_{\text{green},d}$ for each intersection $d \in J$ at controller time step k_c. The function $f(\cdot)$ is the prediction model of the MPC controller, e.g., the BLX model. The stage objective function, $\sum_{(u,d)\in L} \text{TTS}_{(u,d)}(k_c)$ is the total time spent (TTS) of all vehicles on all links of the traffic network during the k_c^{th} control cycle, with L the set of all the links. The decision variables of the optimization problem have an upper and a lower bound, where $u_{\max}$ and $u_{\min}$ are vectors with the maximum and minimum green time percentage of appropriate size, respectively. The inequalities in (3) are defined element-wise.

B. RL for urban traffic signal control

To implement RL in urban traffic signal control, a Markov decision process (MDP) should be utilized to describe the model [12], [13]. In the context of MDP, states, actions, as well as rewards need to be defined for the traffic network. Initially, conventional Q-learning algorithm was proposed for a single-intersection traffic network [14]. The queue length is one of the most common state definitions used in RL-based TSC [10]. As for the actions, the RL-based controller for each traffic light can choose to either remain with the current phase or change to another phase. In RL-based TSC, cost functions used in MPC-TSC are usually taken as reward functions, such as TTS, average junction waiting time, fuel conservation, or average number of trip stops [10].

Due to the curse of dimensionality, standard RL algorithms such as Q-learning cannot handle large state and action spaces. With the advancements in deep RL algorithms, it is nowadays possible to apply RL on larger traffic networks. Haydari and Yilmaz in [15] present an extensive survey on deep RL for TSC. For more information, the interested reader is referred to that paper and the references therein.

C. Model reference framework

The combined MPC-RL framework for TSC proposed in the current paper is inspired by a control framework that combines model-reference control with model-free deep RL and was proposed in [16] for control of autonomous surface vehicles. Thus we briefly explain that framework first. As shown in Figure 1, baseline control laws of two conventional controllers are injected into the real system and a nominal model of the system. The nominal model generates a reference trajectory that the vehicle should follow; meanwhile the output of the nominal model and the output of the real vehicle are compared, the error of which is fed to the RL controller.

TABLE I
COMPARISON OF RL AND MPC CHARACTERISTICS

	MPC	RL
Model needed	Yes	No
Mature stability theory	Yes	No
Mature feasibility theory	Yes	No
Constraints handling	Yes	No
Adaptivity	No	Yes
Online computational complexity	High	Low
Offline computation time	Low	High

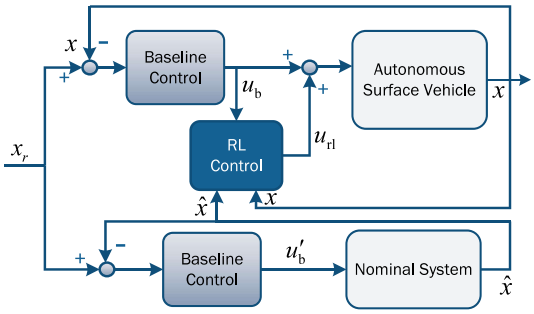

Fig. 1. Model-reference RL control block diagram for control of autonomous surface vehicles [16]

The objective is to steer the vehicle to the desired reference trajectory. The overall control input of the vehicle is the combination of the baseline control u_b and the RL control u_l as denoted in the figure, where u_b is employed to ensure some basic performance, (i.e., local stability), and u_l is introduced to compensate for the system uncertainties or disturbances. With the real states x and the desired state $\hat{x}$ and control input u_b fed into the RL control block, the RL controller can be trained to minimize the error between x and $\hat{x}$ by designing a proper reward function.

III. COMBINED MPC-RL CONTROL FRAMEWORK FOR TRAFFIC NETWORKS

MPC and RL methods exceptionally complement each other (see Table I). Therefore, we propose a framework that combines MPC and RL inspired by [16], in which the advantages of both methods are exploited.

A. Combined MPC-RL control framework

In Figure 2 the combined MPC-RL control framework is illustrated. This framework can potentially mitigate the drawbacks of both MPC and RL control. The adaptive framework reduces the negative effects that uncertainties have on the performance of MPC. Furthermore, the adaptive RL controller also reduces the computational complexity in comparison to conventional MPC. With this combined framework, reduction in the performance of the MPC module, e.g. by premature termination of the optimization process or

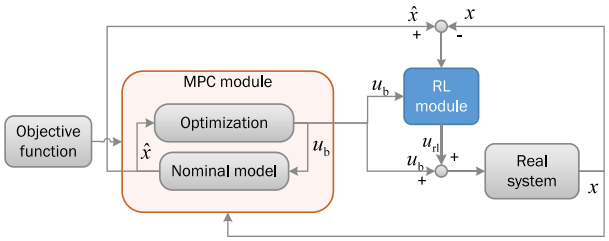

Fig. 2. The combined MPC-RL control framework.

by considering a less accurate model, is affordable since the adaptive control inputs by the RL module are potentially able to compensate for the performance loss. Note that although reduction of the online computation time with the proposed combined control framework compared to conventional MPC is an added benefit of the proposed combined control framework, it is not investigated in the case study of this paper. More specifically, improvement of the control performance in presence of uncertainties is the most important aspect of the proposed control framework compared to regular MPC controllers for urban traffic signal control.

Compared to conventional RL controllers, in the proposed combined framework the MPC offers a baseline control law that sustainably provides acceptable performance during the training process of the RL module. This also makes the framework more sample-efficient, because the RL controller has a relatively smaller action and state space, and the baseline MPC input provide high-quality initial data points for the RL controller to start with. Compared to the model-reference RL control scheme in [16], only one baseline controller (i.e., the MPC module) is used in our proposed framework. Thus the same baseline control input is injected into both the nominal and the real system at every control time step. In addition, the real state x is also fed to the nominal system to update the desired state $\hat{x}$ every cycle. Therefore, the error between the real and desired states varies within a certain range, which makes the learning easier for the RL controller.

As shown in Figure 2 the control law for the urban traffic network (i.e., the real system) is given by:

$$u = u_{\mathrm{b}} + u_{\mathrm{rl}}, \tag{4}$$

where u_{b} is the baseline control input that is the first element of $\boldsymbol{u}_{N_{\mathrm{p}}}(k_{\mathrm{c}})$ for every cycle k_{c} as described in Section II-A, and u_{rl} is the control input generated by the deep RL adaptive control law. In order to cooperate with each other, the MPC and the RL controllers are synchronized and have the same control sampling time. The design of the MPC controller for the computation of the baseline control law is identical to the one introduced in Section II-A. Next we will discuss how to obtain the adaptive RL control law.

B. RL-based control module

To obtain the RL-based adaptive law, the urban traffic network is represented as a Markov decision process (MDP) denoted by tuple $\langle S, A, R, T\rangle$. The state $s(k_{\mathrm{c}}) \in S$, the action $u_{\mathrm{rl}}(k_{\mathrm{c}}) \in A$, and the immediate reward $R(k_{\mathrm{c}}) = R(s(k_{\mathrm{c}}), u_{\mathrm{rl}}(k_{\mathrm{c}}))$ are defined in this section, with k_{c} the cycle counter.

1) State: The state-space of the RL controller contains a vector with the difference between the state of the reference model (x_{m}) and that of the real system (x) at the current time step. This is the difference in the number of vehicles on each link of the network. The elements of the state vector of the RL controller at control time step k_{c} are given by:

$$\delta_n(k_{\mathrm{c}}) = \hat{n}_{u,d}(k_{\mathrm{c}}) - n_{u,d}(k_{\mathrm{c}}) \quad \forall (u,d) \in L, \tag{5}$$

where $\hat{n}_{u,d}(k_{\mathrm{c}})$ and $n_{u,d}(k_{\mathrm{c}})$ denote the desired and the real number of vehicles on each link[1], respectively. Furthermore, the control input of the baseline controller is also provided to the controller. Thus, the state $s(k_{\mathrm{c}}) \in S$ is defined as:

$$s(k_{\mathrm{c}}) = [\delta_n^{\top}(k_{\mathrm{c}}), u_{\mathrm{b}}^{\top}(k_{\mathrm{c}})]^{\top}. \tag{6}$$

2) Action: The RL-based adaptive law should adapt the control input that is computed by the MPC controller according to the disturbances and uncertainties. The control input includes the time length value to extend or reduce the green time during the current cycle. For simplicity, the action $a_d(k_{\mathrm{c}})$ for intersection d will be an integer value between some chosen bounds, where the set of actions is not necessarily evenly distributed over the range. The Deep Q-network (DQN) [17], [18] is used as the RL algorithm, since it uses a discrete action space and a continuous state space, which suit our case. The action space A consists of all possible combinations of actions for all intersections in the network. The action $u_{\mathrm{rl}}(k_{\mathrm{c}})$ for the RL-based controller at control time step k_{c} is defined as:

$$\begin{aligned} &u_{\mathrm{rl}}(k_{\mathrm{c}}) = [a_1(k_{\mathrm{c}}) \dots a_d(k_{\mathrm{c}}) \dots a_{N_d}(k_{\mathrm{c}})]^{\top}, \\ &a_d(k_{\mathrm{c}}) \in \mathbb{Z}, a_{\min}, \leq a_d(k_{\mathrm{c}}) \leq a_{\max}, \ \forall d \in J, \end{aligned} \tag{7}$$

where $a_{\min}$ and $a_{\max}$ are the minimum and maximum value for the action, J is the set of all intersections, and N_J is the size of set J.

3) Reward: Within the combined MPC-RL control framework, it is the goal of the RL controller to follow the desired state determined by the MPC controller. Thus, the reward is determined based on tracking the predicted number of vehicles $\hat{n}_{u,d}$ in the network. As it is not possible to model constraints for the RL controller explicitly, input constraints can be added implicitly in the reward function. More specifically, whenever an infeasible action is chosen by the RL controller, a reward of $R_{\mathrm{c}} = -10$ is added to the immediate reward $R(k_{\mathrm{c}})$, which is:

$$R(k_{\mathrm{c}}) = \sum_{(u,d)\in L} -K|n_{u,d}(k_{\mathrm{c}}) - \hat{n}_{u,d}(k_{\mathrm{c}})| + R_{\mathrm{c}}, \tag{8}$$

where K is a positive gain on the state error, and L is the set of all links in the network. No extra reward is considered when the terminal state is reached.

4) Training: Deep Q-network (DQN) [17], [18] is a deep RL algorithm that has many successful applications, due to the following two key characteristics: experience replay and a target network. A target network is updated periodically, while experience replay destroys the correlations among the data points, which can deal with the instability issue of traditional RL algorithms. The training process of DQN

[1]For simplicity, only the number of vehicles on each link is taken into account in the state definition of the RL-based control module.

is illustrated in Figure 3. At each time step the sample $(s(k_c), u_{rl}(k_c), R(k_c), s(k_c+1))$ is obtained and stored in the replay buffer. These samples are randomly sampled in a mini-batch to train the Q-network. The target network $(\hat{Q}(\theta))$ is updated every T_{targ} controller time steps. The

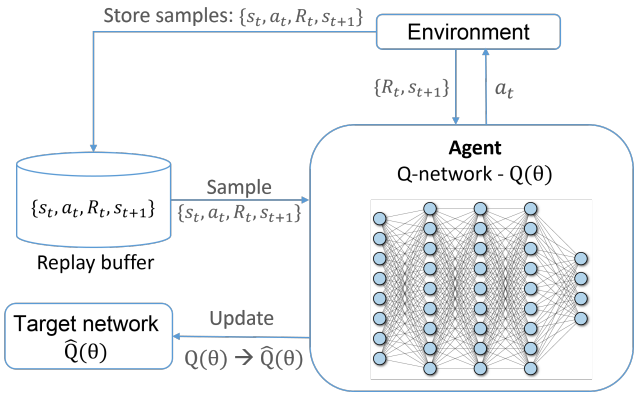

Fig. 3. The training process of the deep reinforcement learning using DQN.

training algorithm including the MPC module is presented in Algorithm 1. The ϵ-greedy parameter ϵ is reduced with a decay rate δ_ϵ by:

$$\epsilon(k_c+1) = \epsilon(k_c)(1-\delta_\epsilon). \tag{9}$$

The Q-network is chosen to be a fully connected multiple layer perceptron with rectified linear unit (ReLU) nonlinearities as activation functions, since hidden layers composed of ReLU typically train faster than the ones composed of other activation functions, such as a sigmoid function [19].

IV. Case Study

In this section, a simulation-based case study is conducted on an urban traffic network to validate the effectiveness of the proposed combined framework. The control performances are measured by the TTS of the vehicles in the network within the simulation window. Several traffic demand scenarios are considered to test the performance of different control methods.

A. Network setup

1) Traffic network structure: In order to analyze the performance of the combined MPC-RL model-reference framework, compared to the standalone MPC controller and RL-based controller, a small grid-based urban traffic network is considered. The layout of the network is shown in Figure 4, which consists of two intersections with controllable traffic signals, each with four connected two-way roads. The network is mathematically described using the BLX model [11]. There are seven two-way roads in total, where each road has three lanes dedicated to right, straight, and left directions. The traffic network has 6 source and 6 sink nodes. All roads have a length of 500 m with the exception of road numbers 7 and 8, which have a length of 550 m. Note that we consider a small traffic network, which is sufficient to gain insight on the performance of the proposed control framework and to evaluate its performance. The small size of the network allows for using discrete actions for the RL-based controller,

Algorithm 1 The combined MPC-RL control algorithm

procedure TRAIN
  Initialize experience buffer
  Initialize network Q with random parameters ϕ
  Initialize target network $\hat{Q}$ with $\hat{\phi} \leftarrow \phi$
  Initialize $\epsilon \leftarrow 1$
  loop
    $s \leftarrow s_0$
    $t \leftarrow 0$
    while $t < t_{end}$ **do**
      MPC: Solve optimization problem (1)-(3) to find $\pi_{green,d}$
      Chose action a using ϵ-greedy
      if Action is infeasible **then**
        Add penalty to reward
        Change action accordingly
      end if
      Take action a, observe next state s' and reward r
      Store (s, a, r, s') in experience buffer
      Select random data points (s, a, r, s') from experience buffer with the size of a mini-batch
      Train network Q by the selected samples experience
      if $t = k \cdot T_{targ}, k \in \mathbb{N}$ **then**
        Update target network $\hat{Q} \leftarrow Q$
      end if
      Update ϵ based on (9)
      $s \leftarrow s'$
      $t \leftarrow t+1$
      MPC: update active constraints and environment states
    end while
  end loop
end procedure

implementing DQN, as well as centralized MPC with little computational effort.

The free-flow speed of the vehicles in the traffic network is 13.9 m/s and the average length of each vehicle is 7 m. The vehicles enter the network via the source links 1, 3, 6, 9, 12 and 14. Each intersection has a controllable traffic signal with a fixed traffic cycle. Each traffic cycle consists of two phases as shown in Figure 5. The first phase has green light for all traffic arriving from the north or south directions, and in the other phase there is a green light for all traffic arriving from the east and west directions. The yellow time at the beginning of each phase is 5 s for all situations and controller types.

2) Predicted demand: The amount of traffic on each road and link in the traffic network is determined by two components: (1) the number of vehicles entering the network on the six entering roads (namely, link 1, 3, 6, 9, 12, and 14), and (2) the turn ratio of each link. For each link, the turn ratios of its three roads sum up to one. The values of the turn ratios for each road are given in Table II. Three demand patterns are designed: one with constant high demand and two with fluctuating demands. The demand profiles of different lanes under the three scenarios are depicted in Figure 6.

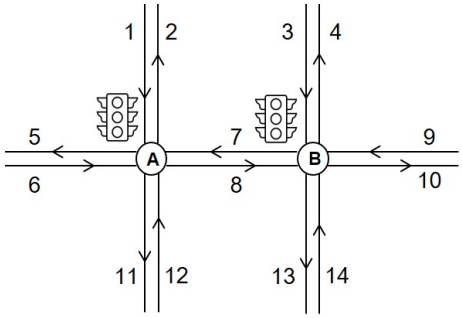

Fig. 4. The studied grid-based traffic network. The numbers present in the network are the link IDs, the arrows in each line represent the driving direction.

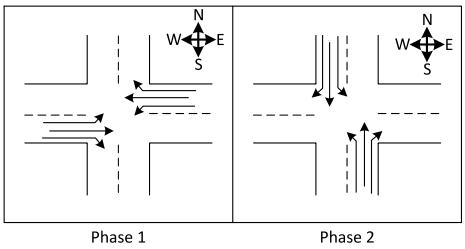

Fig. 5. The two traffic phases for the controller. In phase 1 all traffic arriving from the east and the west direction has right of way, in phase 2 all traffic arriving from north and south direction has right of way.

Before each simulation or training episode, the states of the traffic network are initialized by inserting a low traffic into the traffic network for 300 seconds considering fixed-time traffic signal control for the interactions (see Section IV-B for details).

3) Real demand: The combined MPC-RL control framework is expected to improve the control performance in the presence of uncertainties. Therefore in this case study, we consider disturbances in the predicted traffic demand that enters via source links 1, 3, 6, 9, 12, and 14 (see Figure 4). The demand is disturbed with a sinusoidal wave as shown in Figure 7.

B. Controllers

The proposed combined control framework will be compared to a fixed-time controller, a standalone MPC controller, and a standalone RL-based controller. In this section, the design of the benchmark controllers is discussed, and the parameters for the RL controller are given.

1) Fixed-time controller: For the fixed-time controller, the cycle time is fixed to 60 s and the green time percentage is set to the mean green time percentage of the intersection (i.e. 50% for the two-phased network). With a yellow time length of 5 s for each phase, this means that each phase has 25 s of green time.

2) Standalone MPC controller: The standalone MPC controller used to compare the performance of the framework is the same as the one used within the framework. The design of the controller is identical to the one in Section II-A. Moreover, the prediction horizon of the MPC controller in the framework and the standalone MPC controller is $N_{\mathrm{p}} = 3$ (i.e. 3 cycles), which provides enough time for a vehicle to leave the traffic network, whichever route is chosen. The control horizon is equal to the prediction horizon.

TABLE II

TURN RATIOS FOR ALL ORIGIN LINKS OF THE NETWORK

Direction	β_1	β_3	β_6	β_7	β_8	β_9	β_{12}	β_{14}
Right	0.12	0.44	0.44	0.35	0.35	0.12	0.35	0.12
Straight	0.50	0.44	0.44	0.35	0.35	0.48	0.35	0.44
Left	0.38	0.12	0.12	0.30	0.30	0.40	0.30	0.44

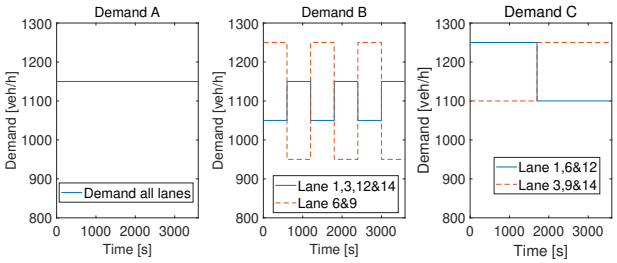

Fig. 6. The demand profiles for different scenarios. Left-hand side plot called demand A: the high-demand case, the middle plot called fluctuating demand B, and the right-hand side plot called fluctuating demand C.

3) Standalone RL-based controller: The RL-based controller is designed based on [20], which also utilizes the DQN algorithm and shares similar parameters and structure as the RL-based control module in the proposed combined framework. Note that the control sampling time of the RL-based controller is 1 s, and the cycle time is not necessarily fixed.

4) Combined MPC-RL controller: The parameters used for the deep RL algorithm (see Figure 3) are given here. The neural network consists of five fully connected layers: an input layer, three inner layers, and an output layer. The number of neurons in the inner layers are 256, 128 and 64, respectively. The number of neurons in the input layer is the number of states and the number of neurons in output layer is the number of actions. The parameters that are used by the DQN algorithm in the framework are presented in Table III, where α is the learning rate of the critic network (the agent network in Figure 3), γ is the discount factor used for computation of the return, ϵ_0 is the starting value of ϵ in the ϵ-greedy method for balancing exploration and exploitation, δ_ϵ is the ϵ decay rate and $\epsilon_{\min}$ is the minimum value of ϵ during training. The other information given in Table III include the experience replay batch size, the optimization method, the activation function of all the neurons, the maximum size of the experience replay buffer and all the dimensions of the network layers. For the actions per intersection, we have:

$$a_d(k_{\mathrm{c}}) \in \{-10, -5, -3, -1, 0, 1, 3, 8, 10\} \quad \forall d \in J. \quad (10)$$

Thus, there are nine actions per intersection, which means that the RL-based controller can choose from a total of 9^{N_d} action combination, where N_d is the number of intersections in the traffic network. In this case study two intersections are considered (see Figure 4), which means an action space of size 81.

C. RL training process

The combined MPC-RL framework will be compared to the standalone RL-based controller as regards their performance during the learning process. As by design these

TABLE III
SETTINGS FOR THE DQN ALGORITHM USED FOR THE MODEL-REFERENCE FRAMEWORK

Episodes	α [-]	γ [-]	ϵ_o [-]	δ_ϵ [-]	ϵ_{min} [-]	T_{targ}
1000	0.01	0.95	1	0.001	0.01	10

Batch size	Optimizer	Act. func.	Buffer size	Input dims.	Output dims.	No. layers
256	Adam	ReLU	$1 \cdot 10^6$	14	81	5

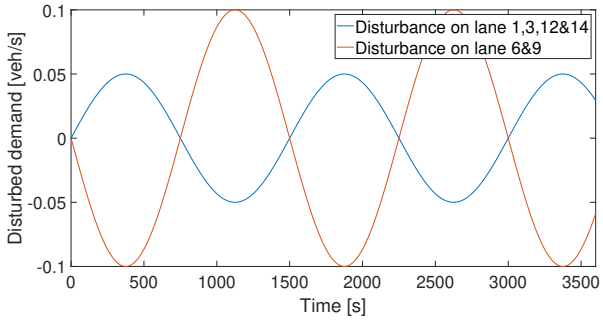

Fig. 7. The disturbance on the traffic demand for different source links in the traffic network. The blue and red curves represent, respectively, the disturbance for the demand on source links 1, 3, 12 and 14 and on source links 6 and 9.

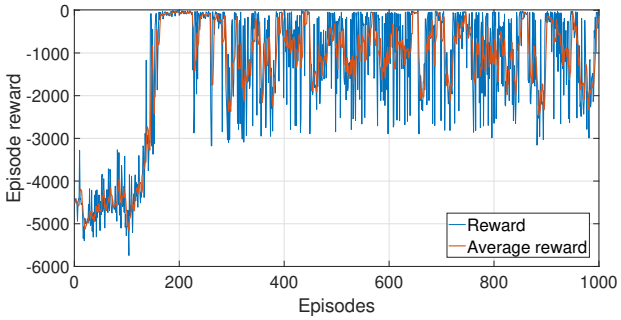

Fig. 8. The learning curve for the standalone RL-based controller

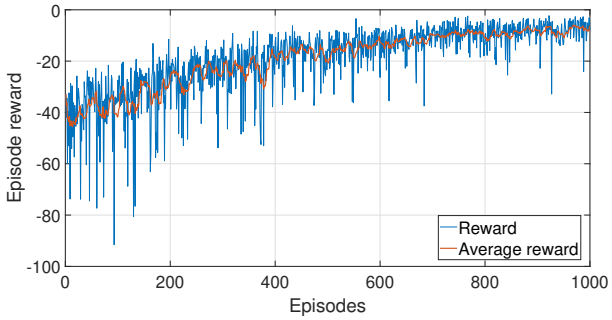

Fig. 9. The learning curve for the adaptive RL module of combined MPC-RL framework

controllers have different reward functions, it is not possible to directly compare the values of their reward function during training. To accomplish a fair comparison, the TTS of the vehicles in the system is documented during the training process and these values are compared. A total of 11 training demand sets are created, all with a length of one hour (3600 s). The training episode terminates when $t = 3600$ s. The system is initialized by inserting a low traffic into the traffic network for 300 s with fixed-time control signal input. Both RL-based controllers are trained using the same demand sets. The training lasts for a sufficient number of episodes until the RL-based controller converges to an optimal policy. In this case, the number of episode is 1000.

In Figure 8 the training process of the standalone RL-based controller is presented. Figure 9 shows the training process for the RL module of the proposed combined MPC-RL controller. The red curves in these figures represent the moving average of the last ten episodes. Note that due to the use of different demand sets during training, the reward value of both learning curves does not converge to a specific value, but will always fluctuate in all stages of the learning process. Despite this fluctuation, the episode rewards show that the RL policy converges to an optimum. The RL algorithm of the standalone RL-based controller learns a well-performing policy after approximately 200 episodes. After approximately 220 episodes the reward function deteriorates again, which is likely to be caused by a combination of catastrophic forgetting and overfitting of the neural network. This issue can be avoided by an early stop of the training process, and the best-performing RL-based controller during the training process is selected for comparison in the next section. The fact that the standalone RL-based controller converges faster than the combined MPC-RL framework can be explained by that the RL agent has a faster control sampling time, i.e., the RL-agent takes 3600 actions during one episode while the framework agent only takes 60 actions per episode.

Figure 10 shows the performances as regards TTS of the standalone RL-based controller and the combined MPC-RL controller during training over all episodes. The TTS curve shows that the combined MPC-RL framework performs well in all training episodes, already from the first episode when the TTS is low. This happens because the performance of the combined MPC-RL control framework is mainly dependent on the performance of the baseline MPC controller, which provides a well-performing input from the start of the learning process. The standalone RL-based controller, however, does not perform well up until around the 160^{th} episode. This is because that the RL-based controller has to learn the entire policy from scratch.

D. Post-training simulation results

After training the RL module of the combined MPC-RL controller and the standalone RL-based controller, the performance of these controllers is simulated and compared. The simulations are done with the demand pattern described in Section IV-A. Note that the value of ϵ in the exploration method is set to zero for both RL algorithms to achieve a deterministic policy.

All four controllers are simulated with all three demand data sets. In Table IV, V and VI the system performance in

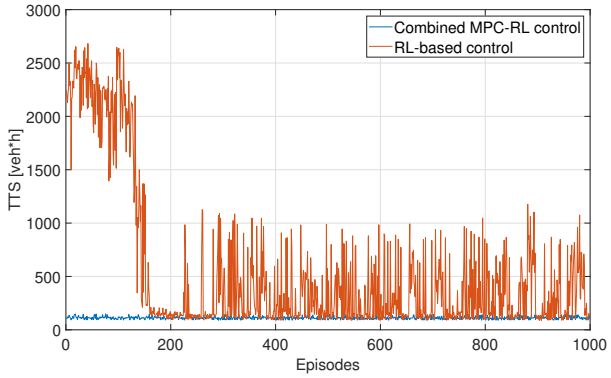

Fig. 10. The system performance is expressed in TTS during training for the standalone RL-based controller and the combined MPC-RL controller

terms of TTS and relative TTS of the different controllers for all three demand scenarios are shown and compared. The relative TTS is taken with respect to the fixed-time controller. In all three demand cases, the MPC controller, the RL-based controller, and the combined MPC-RL controller reduce the TTS with respect to the fixed-time controller. In the first two demand cases, the proposed combined control framework results in the lowest TTS; in the last demand case the RL-based controller performs the best. In general, the combined MPC-RL control framework performs better than the MPC controller in all cases. The TTS is reduced by 7.0%, 4.1% and 3.3% compared to the standalone MPC controller for the three demand profiles respectively. This performance improvement compared to the MPC controller is caused by the addition of the RL adaptive control law, which is designed to reject the disturbances in the traffic network. The standalone RL-based controller has a better performance than the MPC controller in all the demand cases and than the combined control framework in the third demand case. The fact that the RL-based controller performs better can be explained by the flexibility of the control input. For the MPC controller and the combined control framework the cycle time is set to 60 s. The RL-based controller, however, can adjust the phase length freely. By doing this, the controller has the potential to create a cycle offset between the two intersections, which can benefit the system performance.

V. Conclusions and Topics for Future Work

In this paper, we proposed a new combined control framework that exploits the benefits of model predictive control (MPC) and reinforcement learning (RL). The proposed combined controller was applied for signal control of an urban traffic network. In the presence of model uncertainties or disturbances, the proposed combined MPC-RL control framework outperforms standalone MPC, and has comparable performance with well-trained standalone RL-based controllers that have a higher control frequency and require lengthy training procedures before they perform well.

In the future, a more extensive comparison between the proposed combined controller and standalone RL-based controller will be carried out, by using the same control sampling time and flexible traffic cycle setting. Moreover, a more extensive assessment based on simulations with a more realistic traffic simulator such as SUMO [21] will be performed. In addition, the online computation of MPC in the framework can be further reduced by using parameterized MPC or a simplified traffic model. Thus the computational efficiency can be improved without compromise of performance.

TABLE IV

The system performance in terms of TTS and relative TTS for the different controllers for demand A

Controller	TTS [veh·h]	Relative TTS [%]
Fixed-time control	137.6	-
MPC	125.2	-8.5
RL-based control	122.5	-10.9
Combined MPC-RL control	117.1	-14.9

TABLE V

The system performance in terms of TTS and relative TTS for the different controllers for demand B

Controller	TTS [veh·h]	Relative TTS [%]
Fixed-time control	141.3	-
MPC	123.2	-12.8
RL-based control	121.6	-13.9
Combined MPC-RL control	118.1	-16.4

TABLE VI

The system performance in terms of TTS and relative TTS for the different controllers for demand C

Controller	TTS [veh·h]	Relative TTS [%]
Fixed-time control	157.0	-
MPC	145.1	-7.6
RL-based control	138.2	-12.0
Combined MPC-RL control	140.3	-10.7

Acknowledgment

The authors would like to thank Dr. Q. Zhang, the author of [16], for his valuable suggestions on the design and parameter tuning for the RL module within the proposed control framework.

References

[1] M. Barth and K. Boriboonsomsin, "Traffic congestion and greenhouse gases," *Access Magazine*, no. 35, pp. 2–9, 2009.
[2] CBS, "Welke sectoren stoten broeikasgassen uit?" https://www.cbs.nl/nl-nl/dossier/dossier-broeikasgassen, 2019.
[3] European Commission, "Urban Mobility," https://ec.europa.eu/transport/themes/urban/ urban_mobility_en, 2019.
[4] M. Papageorgiou, C. Kiakaki, V. Dinopoulou, A. Kotsialos, and Yibing Wang, "Review of road traffic control strategies," *Proceedings of the IEEE*, vol. 91, no. 12, pp. 2043–2067, dec 2003.
[5] Y. Du, W. Shangguan, D. Rong, and L. Chai, "RA-TSC: Learning adaptive traffic signal control strategy via deep reinforcement learning," *2019 IEEE Intelligent Transportation Systems Conference, ITSC 2019*, pp. 3275–3280, 2019.
[6] E. F. Camacho and C. Bordons, *Model Predictive Control*, second edition ed., ser. Advanced Textbooks in Control and Signal Processing. London: Springer London, 2007.
[7] B. De Schutter and B. De Moor, "Optimal traffic light control for a single intersection," *European Journal of Control*, vol. 4, pp. 260–276, 1998.
[8] A. Jamshidnejad, D. Sun, A. Ferrara, and B. De Schutter, "A novel bi-level temporally-distributed MPC: An application for persistent green urban mobility," *Manuscript submitted*, 2021.
[9] J. Jeschke, D. Sun, A. Jamshidnejad, and B. De Schutter, "Grammatical-evolution-based parameterized model predictive control for urban traffic networks," *Manuscript submitted*, 2022.

[10] P. Mannion, J. Duggan, and E. Howley, "An experimental review of reinforcement learning algorithms for adaptive traffic signal control," in *Autonomic Road Transport Support Systems*. Cham: Springer International Publishing, 2016, pp. 47–66. [Online]. Available: http://link.springer.com/10.1007/978-3-319-25808-9_4

[11] S. Lin and Y. Xi, "An efficient model for urban traffic network control," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 14 066–14 071, 2008.

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

[13] L. Buşoniu, R. Babuška, B. De Schutter, and D. Ernst, *Reinforcement Learning and Dynamic Programming Using Function Approximators*. CRC Press, 2010.

[14] B. Abdulhai, R. Pringle, and G. J. Karakoulas, "Reinforcement learning for true adaptive traffic signal control," *Journal of Transportation Engineering*, vol. 129, no. 3, pp. 278–285, 2003.

[15] A. Haydari and Y. Yılmaz, "Deep reinforcement learning for intelligent transportation systems: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 1, pp. 11–32, 2022.

[16] Q. Zhang, W. Pan, and V. Reppa, "Model-reference reinforcement learning for collision-free tracking control of autonomous surface vehicles," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[17] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv:1312.5602v1*, pp. 1–9, dec 2013.

[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[19] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 315–323.

[20] S. El-Tantawy, B. Abdulhai, and H. Abdelgawad, "Design of reinforcement learning parameters for seamless application of adaptive traffic signal control," *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations*, vol. 18, no. 3, pp. 227–245, 2014.

[21] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y. P. Flotterod, R. Hilbrich, L. Lucken, J. Rummel, P. Wagner, and E. Wiebner, "Microscopic traffic simulation using SUMO," in *IEEE Intelligent Transportation Systems Conference (ITSC)*, vol. 21. Maui, Hawaii: IEEE, 2018, pp. 2575–2582. [Online]. Available: https://elib.dlr.de/124092/