Grammatical-Evolution-based parameterized Model Predictive Control for urban traffic networks

Jeschke, Joost; Sun, Dingshan; Jamshidnejad, Anahita; De Schutter, Bart

DOI
10.1016/j.conengprac.2022.105431

Publication date
2023

Document Version
Final published version

Published in
Control Engineering Practice

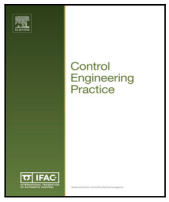
Citation (APA)
Jeschke, J., Sun, D., Jamshidnejad, A., & De Schutter, B. (2023). Grammatical-Evolution-based parameterized Model Predictive Control for urban traffic networks. *Control Engineering Practice*, *132*, Article 105431. https://doi.org/10.1016/j.conengprac.2022.105431

Control Engineering Practice 132 (2023) 105431

Contents lists available at ScienceDirect

Control Engineering Practice

journal homepage: www.elsevier.com/locate/conengprac



Grammatical-Evolution-based parameterized Model Predictive Control for urban traffic networks ☆

Joost Jeschke [a,b,1], Dingshan Sun [a,*,1], Anahita Jamshidnejad [c], Bart De Schutter [a]

[a] Delft Center for Systems and Control, Delft University of Technology, Mekelweg 2, Delft, 2628 CD, The Netherlands
[b] CGI Nederland, George Hintzenweg 89, Rotterdam, 3068 AX, The Netherlands
[c] Department of Control and Operations, Delft University of Technology, Kluyverweg 1, Delft, 2629 HS, The Netherlands

ARTICLE INFO

Keywords:
Model Predictive Control
Parameterized controller
Urban traffic control
Grammatical Evolution

ABSTRACT

While Model Predictive Control (MPC) is a promising approach for network-wide control of urban traffic, the computational complexity of the, often nonlinear, online optimization procedure is too high for real-time implementations. In order to make MPC computationally efficient, this paper introduces a *parameterized* MPC (PMPC) approach for urban traffic networks that uses Grammatical Evolution to construct continuous parameterized control laws using an effective simulation-based training framework. Furthermore, a projection-based method is proposed to remove the nonlinear constraints that are imposed on the parameters of the parameterized control laws and to guarantee the feasibility of the solution of the MPC optimization problem. The performance and computational efficiency of the constructed parameterized control laws are compared to those of a conventional MPC controller in an extensive simulation-based case study. The results show that the parameterized control laws, which are automatically constructed using Grammatical Evolution, decrease the computational complexity of the online optimization problem by more than 80% with a decrease in performance by less than 10%.

1. Introduction

Over the past decades, a growing demand for urban mobility has led to congested urban areas. Various control strategies have been proposed to meet this growing demand and to increase the traffic flow in urban traffic networks. Next we briefly discuss about traffic signal control and MPC for urban traffic networks.

1.1. Traffic signal control

Traffic signal control has evolved over the years. Webster (1958) proposed one of the first traffic signal control methods for minimizing the delay per vehicle. From there, different controllers were designed for single intersections, while they did not interact with adjacent intersections. This resulted in optimized control strategies for single intersections, where it could lead to congestion in other intersections in the traffic network. To address this issue, fixed-time strategies (Little et al., 1981; Robertson, 1986) were proposed to control multiple intersections at the same time. Fixed-time control strategies determine the control inputs offline based on historical traffic flow data. One of the disadvantages of fixed-time controllers is that they do not respond to real-time traffic fluctuations, e.g., when an accident occurs. To tackle this issue, traffic-responsive controllers were introduced (Hunt et al., 1982; Sims, 1979). Such controllers take the current traffic conditions that are measured by loop detectors into account and change the green times of the traffic lights accordingly.

Model-based control strategies (Gartner, 1983; Henry et al., 1984; Mirchandani & Head, 2001) are traffic-responsive methods that use a mathematical model to predict future traffic conditions and to calculate an optimized control input sequence for the traffic network. Model-based control approaches consist of a prediction model, an online optimization procedure, and a rolling horizon principle. By using future predictions, non-myopic control inputs can be obtained. Model Predictive Control (MPC) (Rawlings & Mayne, 2009) is a model-based control method that is widely used in different industrial areas (Afram & Janabi-Sharifi, 2014; Qin & Badgwell, 2003), and it has shown to be promising for urban traffic signal control (Ye et al., 2019).

☆ 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, China (CSC Grant No. 201806230254).

* Corresponding author.
E-mail addresses: joostjeschke@gmail.com (J. Jeschke), d.sun-1@tudelft.nl (D. Sun), a.jamshidnejad@tudelft.nl (A. Jamshidnejad), b.deschutter@tudelft.nl (B. De Schutter).

[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.conengprac.2022.105431
Received 13 May 2022; Received in revised form 3 December 2022; Accepted 29 December 2022
Available online 12 January 2023

1.2. Model Predictive Control for urban traffic networks

In MPC, a mathematical model is used to predict future states of the controlled traffic network over a prediction horizon of size N_p and to calculate an optimized control sequence at every control time step within the prediction window. MPC can simultaneously optimize multiple control objectives, e.g., the total time spent by the vehicles in the traffic network and the total emissions of the vehicles. Moreover, due to its rolling horizon approach, MPC can work based on real-time feedback from the traffic network, and thus quickly respond to changing traffic demands. Additionally, queue lengths on the roads and green times of the traffic lights can be constrained since MPC takes input and state constraints explicitly into account. Finally, the prediction model of MPC can easily be updated or replaced by another model in order to provide a desired trade-off between accuracy of the predictions and complexity of computing them. On the one hand, a more precise model will in general be computationally more complex (due to considering more state variables or incorporating nonlinearities), which results in a more complex online optimization problem that may be intractable in real time. On the other hand, while a less accurate model is computationally more efficient, the corresponding predictions are prone to larger errors. This can result in significant cumulative errors across the MPC prediction horizon, and thus in a degraded performance for the controlled system.

A major drawback of MPC especially for urban traffic networks, is the need for performing an online optimization procedure per control time step. Due to the nonlinear behavior of traffic and thus the need for nonlinear prediction models (Jamshidnejad et al., 2019; Lin et al., 2012; Lin & Xi, 2008; Ye et al., 2015), the resulting MPC optimization problem is nonlinear and nonconvex, with a large number of optimization variables (which correspond to the number of traffic signals/intersection within the traffic network). Therefore, a computationally complex optimization problem should be solved online, which makes MPC intractable for real-time implementation for urban traffic networks.

Different approaches have been proposed to lower the computational complexity of the online MPC optimization problem for urban traffic. In Ye et al. (2016, 2015) for instance, the traffic network is divided into multiple subnetworks, each corresponding to one local optimization problem that takes the interactions with the neighboring subnetworks into account. In Remmerswaal et al. (2022), MPC is combined with reinforcement learning for urban traffic signal control. Since reinforcement learning can deal with uncertainties and provide extra optimality, MPC can operate with a less accurate model or act at a lower control frequency, in order to reduce computational complexity. Lin et al. (2011) reformulate the nonlinear and nonconvex MPC optimization problem as a computationally efficient mixed-integer linear optimization problem (MILP). In this paper, we focus on reducing the computational complexity by parameterizing the decision variables of the MPC optimization problem.

In PMPC the decision variables are parameterized, which results in fewer decision variables and potentially lower computation time for the online optimization problem (Goulart et al., 2006; Lofberg, 2003; Pippia et al., 2018; Zegeye et al., 2012). PMPC has shown promising results regarding computational efficiency in control of urban and freeway traffic networks (Jeschke & De Schutter, 2021; van Kooten et al., 2017; Zegeye et al., 2012), via substantially reducing the number of optimization decision variables with limited decrease in the performance. However, the parameterized control laws in Jeschke and De Schutter (2021), van Kooten et al. (2017), and Zegeye et al. (2012) are handcrafted based on expert knowledge and experiences and are therefore difficult to design.

1.3. Main contributions

The main contributions of this paper include:

1. We use Grammatical Evolution (GE) to automatically construct continuous parameterized control laws based only on limited knowledge of the system. More specifically, two training frameworks, called Framework-1 and Framework-2, are proposed and investigated to automatically generate the parameterized control laws. While Framework-1 is similar to the one used in Jeschke and De Schutter (2021), the newly proposed Framework-2, which is shown to outperform Framework-1 in terms of performance measurements and training efficiency, is able to train multiple parameterized control laws at the same time.
2. An effective projection-based method is proposed to remove nonlinear constraints on the parameters of the PMPC problem in order to guarantee the feasibility and to reduce the computational complexity of the optimization problem.
3. We show the effectiveness of the GE-based PMPC controllers in a case study and compared to a conventional MPC controller, a fixed-time controller, and the handcrafted parameterized control laws from Jeschke and De Schutter (2021).

This paper significantly extends the work of Jeschke and De Schutter (2021) by proposing a new training framework for parameterized control laws that improves the performance, and by proposing a projection-based method to more efficiently deal with the constraints in PMPC.

1.4. Outline of the paper

The remainder of this paper is organized as follows. First, in Section 2 we discuss the principles behind conventional MPC and PMPC and provide the necessary background on the urban traffic model that is used. In Section 3, an overview of Grammatical Evolution (GE) is presented, followed by the newly proposed training frameworks for the parameterized control laws with GE and the projection-based method in Section 4. In Section 5 we present, compare, and discuss the effectiveness of the proposed training frameworks and the resulting parameterized control laws. Finally, in Section 6 we draw final conclusions and provide some suggestions for future work.

2. Parameterized model predictive urban traffic control

In this section we describe the principles behind PMPC in urban traffic control, as well as the mathematical constraints in model-based urban traffic control. Moreover, we shortly discuss the baseline parameterized control law (for the PMPC controller) that will be used in the case study. First, we introduce the mathematical urban traffic model that is used in the MPC controllers.

2.1. Urban traffic prediction model

We use the S-model (Jamshidnejad et al., 2019; Lin et al., 2012) as the prediction model for the PMPC controllers since this model provides a suitable balance between accuracy and computational complexity. The S-model is a macroscopic, nonlinear, and discrete-time urban traffic model that considers the cycle time of the downstream intersection of a link to update the traffic states, i.e., the number of vehicles and the queue lengths of that link. We give only the main equations of the model that are needed to understand the remainder of the paper. For more details, we refer the reader to Lin et al. (2012) and Jamshidnejad et al. (2019).

The S-model represents an urban traffic network by a set of nodes N, a set of links L, and a set of controlled intersections $J \subseteq N$ (see Fig. 1). A link $(u, d) \in L$ is defined by its upstream node $u \in N$ and downstream node $d \in N$, and corresponds to a set $I_{u,d}$ of input nodes

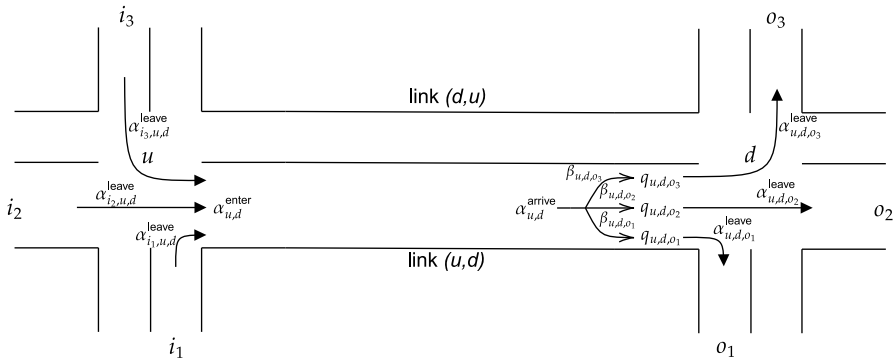


Fig. 1. A link in the S-model connecting two traffic-signal-controlled intersections, based on Lin et al. (2012).

and a set $O_{u,d}$ of output nodes. The cycle times of the upstream and downstream node are given by c_u and c_d, respectively. For simplicity, in this paper, the cycle time for all the intersections are considered equal. Furthermore, the control time interval and simulation time interval of the network are equal, resulting in one common time step counter k.

The state variables of the S-model are the total number of vehicles $n_{u,d}(k)$ and the queue length $q_{u,d}(k)$ on each link (u,d) per simulation time step k. The queue lengths can be further divided into queue lengths $q_{u,d,o}(k)$ corresponding to vehicles that move towards a specific output node $o \in O_{u,d}$. The number of vehicles and the queue lengths are updated every simulation time step k by

$$n_{u,d}(k+1) = n_{u,d}(k) + \left(\alpha^{\text{ent}}_{u,d}(k) - \alpha^{\text{leave}}_{u,d}(k)\right) \cdot c_d, \tag{1}$$

$$q_{u,d,o}(k+1) = q_{u,d,o}(k) + \left(\alpha^{\text{arr}}_{u,d,o}(k) - \alpha^{\text{leave}}_{u,d,o}(k)\right) \cdot c_d, \tag{2}$$

$$q_{u,d}(k) = \sum_{o \in O_{u,d}} q_{u,d,o}(k), \tag{3}$$

where $\alpha^{\text{ent}}_{u,d}(k)$ and $\alpha^{\text{leave}}_{u,d}(k)$ are the average entering and leaving flow rates of link (u,d), $\alpha^{\text{arr}}_{u,d,o}(k)$ is the average arriving flow rate at the tail of the queue on link (u,d) that intends to move towards node o, and $\alpha^{\text{leave}}_{u,d,o}(k)$ is the average leaving flow rate of the sub-stream on link (u,d) that intends to move towards node o, during the time interval $[kc_d, (k+1)c_d)$. The leaving flow rates are nonlinear functions of the states and the green time of the traffic lights, i.e.,

$$\alpha^{\text{leave}}_{u,d,o}(k) = h(\boldsymbol{x}(k), g_{u,d,o}(k)), \tag{4}$$

with $h(\cdot,\cdot)$ a nonlinear function taking different traffic conditions into account (see Lin et al., 2012 for more details), $g_{u,d,o}(k)$ the green time duration for the vehicles on link (u,d) that intend to turn towards node o during the time interval $[kc_d, (k+1)c_d)$, and $\boldsymbol{x}(k)$ a column vector containing $n_{u,d}(k)$ and $q_{u,d}(k)$ for all $(u,d) \in L$ (i.e., the number of vehicles and queue lengths of all the links).

To prevent collisions and to regulate the traffic, the cycle time per intersection is divided into phases for which certain lanes have right-of-way (i.e., a green light). For example, during one phase, two perpendicular incoming lanes do not have right-of-way to go straight over the intersection in the same phase. The green time duration for the individual lanes is linked to the phase times, while grouping the green times of the individual traffic lights into phases reduces the number of inputs that should be processed by the model per simulation time step, it imposes a constraint, i.e., the phase times at an intersection should add up to the cycle time of that intersection minus the yellow time of the traffic lights. Thus, for every simulation time step k we should have:

$$c_d = y_d + \sum_{i=1}^{N^{\text{ph}}_d} g_{d,i}(k), \tag{5}$$

in which y_d is the total yellow time for intersection d during a cycle, $g_{d,i}(k)$ is the green time of phase i at intersection d for simulation time step k, and N^{ph}_d is the number of phases for intersection d.

2.2. Parameterized Model Predictive Control

The MPC optimization problem that is solved at every control time step for an urban traffic network is given by:

$$\min_{\boldsymbol{g}(k)} \left(w_{\text{TTS}} J_{\text{TTS}}(k) + w_{\mathcal{D}} \mathcal{D}(\boldsymbol{g}(k)) + w_{\mathcal{Q}} \mathcal{Q}(k)\right) \tag{6}$$

$$\text{s.t.} \quad \boldsymbol{x}(k+j+1) = f(\boldsymbol{x}(k+j), \boldsymbol{g}(k+j)),$$
$$\boldsymbol{g}_{d,\min} \le \boldsymbol{g}_d(k) \le \boldsymbol{g}_{d,\max} \quad \forall d \in \boldsymbol{J},$$
$$(5),$$

holds for $j = 0, \dots, N_{\text{p}} - 1$ and where $J_{\text{TTS}}(k)$ stands for the total time spent (by the vehicles in the traffic network) predicted within the prediction window of size N_{p} for control time step k. Moreover, $\mathcal{D}(\boldsymbol{g}(k))$ and $\mathcal{Q}(k)$ represent, respectively, a cost on the control input increments computed within the entire prediction window of size N_{p} to prevent high fluctuations in consecutive control time steps, and a cost to take the longest queue per intersection at every control time step into account in order to avoid long queues that congest the downstream intersections. The formulations of these terms are specified in more detail in Section 5.1. Furthermore, $f(\cdot,\cdot)$ is the prediction model (i.e., the S-model explained in Section 2.1), $\boldsymbol{x}(k)$ is the state vector of the model as defined in Section 2.1, $\boldsymbol{g}_d(k)$ contains the phase times at intersection d at time step k, $\boldsymbol{g}(k)$ is a column vector containing $\boldsymbol{g}_d(k), \boldsymbol{g}_d(k+1), \dots, \boldsymbol{g}_d(k+N_{\text{p}}-1)$ for all $d \in \boldsymbol{J}$, $\boldsymbol{g}_{d,\min}$ and $\boldsymbol{g}_{d,\max}$ vectors of appropriate size with the minimum and maximum green times of the phase times at intersection d, respectively, for which '$\le$' is considered element-wise, w_{TTS}, $w_{\mathcal{D}}$, and $w_{\mathcal{Q}}$ the weights that describe the importance of the different control objectives, and (5) is the equality constraint on the phase times.

In order to reformulate (6) as a PMPC problem, the original control inputs are replaced by a parameterized control law that is added to the constraints of the MPC optimization problem, and the parameters of this control law are then optimized. We have:

$$\min_{\boldsymbol{\theta}} \left(w_{\text{TTS}} J_{\text{TTS}}(k) + w_{\mathcal{D}} \mathcal{D}(\boldsymbol{g}(k,\boldsymbol{\theta})) + w_{\mathcal{Q}} \mathcal{Q}(k)\right). \tag{7}$$

which in addition to the constraints given by (6) is also subjected to the parameterized control law that calculates the phase times, i.e., for $j = 0, \dots, N_{\text{p}} - 1$:

$$\boldsymbol{g}_d(k+j, \boldsymbol{\theta}_d) = \mu_d(\boldsymbol{x}(k+j), \boldsymbol{\theta}_d) \tag{8}$$

where $\mu_d(\cdot,\cdot)$ is the parameterized control law of intersection d and $\boldsymbol{\theta}_d$ is a vector that includes the parameters of that control law. While multiple intersections can use the same parameterized control law, the parameters for every intersection are independent. Moreover, $\boldsymbol{\theta}_d = \left[\theta_1, \dots, \theta_{N^{\theta}_d}\right]^{\top}$, where N^{θ}_d is the number of parameters for the control law of intersection d and $\boldsymbol{\theta}$ is a column vector containing $\boldsymbol{\theta}_d$ for all $d \in \boldsymbol{J}$.

If the number of parameters is lower than the original number of decision variables, the optimization should in general run faster. There

are two main challenges in PMPC. The first one is finding a parameterized control law that results in faster optimization without degrading the performance significantly. Secondly, as the parameterized control law is added to the constraints of the PMPC optimization problem, one need to ensure that the solution to this optimization problem remains feasible with the extra constraints.

Please note that since the control inputs become a function of the states in PMPC, the parameters do not necessarily have to be updated every control time step, as future control inputs can be calculated with future states and parameters from the previous control time step. Keeping θ constant over the prediction horizon reduces the number of decision variables in the optimization problem. However, one could make use of time-dependent parameters or use the idea of move-blocking MPC (Cagienard et al., 2007) in which the decision variables are held constant over several time steps to reduce the number of decision variables. This yields a trade-off between performance and computational complexity.

2.2.1. Relative queue lengths parameterized control law

Later on, in the case study, we will use the best performing parameterized control law of Jeschke and De Schutter (2021), to show the effectiveness of this approach on a larger traffic network and to use it as a baseline for the other PMPC controllers considered in this paper. This control law was designed using expert knowledge of the system. Here, we will only give the formulation of the parameterized control law for the clarity of this paper. For more details, we refer the reader to Jeschke and De Schutter (2021).

The parameterized control law uses the mean queue length $q_{d,j}^{\text{ph}}(k)$ and the mean arriving flow rate $\alpha_{d,j}^{\text{ph,arr}}(k)$ on the lanes that have right-of-way in the jth phase at intersection d at control time step k. The mean of the mean queue lengths of all the phases at intersection d is denoted by $\overline{q}_d^{\text{ph}}(k)$ and the mean of the mean arriving flow rates of all the phases at intersection d is denoted by $\overline{\alpha}_d^{\text{ph,arr}}(k)$. The parameterized control law that calculates the green time $g_{d,j}(k)$ of phase j at intersection d is given by

$$g_{d,j} = \overline{g}_d + \frac{q_{d,j}^{\text{ph}} - \overline{q}_d^{\text{ph}}}{\sum_{i=1}^{N_d^{\text{ph}}} q_{d,i}^{\text{ph}} + \kappa_q} \cdot \theta_{d,1} + \frac{\alpha_{d,j}^{\text{ph,arr}} - \overline{\alpha}_d^{\text{ph,arr}}}{\sum_{i=1}^{N_d^{\text{ph}}} \alpha_{d,i}^{\text{ph,arr}} + \kappa_\alpha} \cdot \theta_{d,2} \tag{9}$$

where $\overline{g}_d$ is the mean green time during one cycle at intersection d, N_d^{ph} is the number of phases at intersection d, κ_q and κ_α are small positive values to prevent division by zero, and $\theta_{d,1}$ and $\theta_{d,2}$ are independent parameters for intersection d.

3. Grammatical Evolution

Grammatical Evolution is a form of genetic programming that produces functions based on a user-defined context-free grammar (O'Neill & Ryan, 2001) and an evolutionary algorithm for evolving the functions (Nicolau & Agapitos, 2018).

3.1. Genetic programming

Techniques that are used for the evolvement of functions with evolutionary algorithms are called genetic programming (Koza, 1994). Genetic programming uses a (derivation) tree-based structure to represent the functions, where these trees can be evaluated in a recursive manner (see Fig. 2). The resulting functions can consist of complex programming languages or can be more simple curve-fitting models or symbolic regressions (Poli et al., 2008). For our application, the produced functions are parameterized control laws.

The basic genetic programming algorithm (Poli et al., 2008) works with a function set and a terminal set. The function set often consists of mathematical operators, logical operators, and user-defined functions, and the terminal set consists of the operands, e.g. the dependent variables or constants. The genetic programming algorithm is initialized

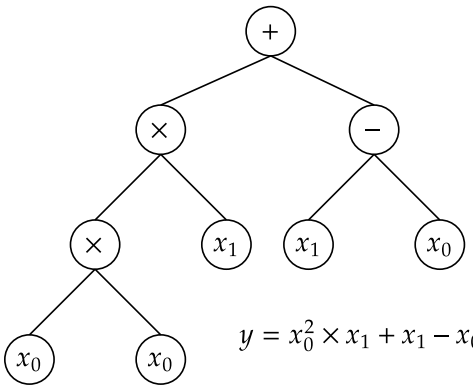


Fig. 2. Tree representation of genetic programming for a symbolic regression problem. Here, the function set contains the mathematical operators and the terminal set contains x_0 and x_1.

with an initial population of functions. Then a selection process is performed to choose the best a few functions according to a given criterion, such as a fitness function that evaluates the performance (see Section 3.3). Based on the selected functions, a new population of functions is generated using sub-tree crossover and sub-tree mutation. In sub-tree crossover, two functions are combined to create two new functions by interchanging parts of the trees. In sub-tree mutation, single nodes in the tree are replaced by other nodes from their respective set (i.e. the function and terminal set). In particular, in each tree a node is selected, and the successive branches of these nodes are interchanged.

Genetic programming is especially useful when the exact form of the function is not known in advance (Poli et al., 2008) as no constraints are set on the output of the algorithm. However, in PMPC of urban traffic networks, there is some information about the solution of the optimization problem, i.e., the sum of the green times for an intersection should add up to the cycle time of the intersection minus the yellow time (see (5)). When genetic programming is used to find a function that generates the phase times of an intersection, it is very unlikely that these phase times add up to the cycle time of the intersection. Thus the algorithm may not return a valid function due to the phase time constraint. Moreover, since the search space of genetic programming is generally very large, it is helpful to steer the algorithm in the right direction.

To reduce the search space and to guarantee the feasibility of the created programs, we can use Grammatical Evolution (GE) with a context-free grammar to generate the functions. The grammar describes how the functions should be constructed and imposes constraints on the search space.

3.2. GE with context-free grammars

Context-free grammars have a recursive notation and describe how functions can be constructed from an existing list of variables and functions (Hopcroft et al., 2006). Context-free grammars are often defined in Backus–Naur form Ryan et al. (2018), considering four basic components: a finite set of terminals, a finite set of non-terminals, a start symbol, and a finite set of production rules. The production rules represent the recursive behavior of the context-free grammar and define how a non-terminal evaluates to another non-terminal, a terminal, or a combination of the two. The non-terminals give structure to the grammar and the terminals end up in the resulting function. A set of production rules for a context-free grammar that could be used for symbolic regression is shown in Fig. 3. The top production rule contains the start symbol S, which is replaced by an expression. All the non-terminals are defined between angle brackets and all the terminals (i.e.,

Rule			Rule number
S	::=	<expr>	(0)
<expr>	::=	(<expr> <op> <expr>)	(0)
		| <func> (<expr>)	(1)
		| <terminal>	(2)
<op>	::=	+ | - | * | /	(0-3)
<func>	::=	sin | cos | exp | log	(0-3)
<terminal>	::=	<xlist>	(0)
		| <digitlist>.<digitlist>	(1)
<xlist>	::=	x1 | x2 | x3	(0-2)
<digitlist>	::=	<digit>	(0)
		| <digit><digitlist>	(1)
<digit>	::=	0 | 1 | 2 | ... | 9	(0-9)

Fig. 3. The production rules of a context-free grammar in Backus–Naur form, based on Tsoulos et al. (2008), where the words enclosed by angle brackets are the non-terminals, ::= indicates replacement of the left-hand side symbol with the right-hand side expression, and the vertical bar is used for separation of the different options a non-terminal can evaluate to. Furthermore, the resulting function will consist of the terminals sin, cos, exp and log, representing the sinus, cosine, exponential and logarithmic functions, respectively, and the variables $x1$, $x2$, $x3$ and the numbers 0 to 9.

the arithmetic operators, the variables x, and the numbers) are in the production rules of some of the non-terminals.

The grammar in Fig. 3 can be used for symbolic regression purposes, but more advanced grammars can be used for, e.g., solving the Sante Fe ant trail problem (Nicolau et al., 2012), the control of femtocell network coverage (Hemberg et al., 2013), and finance and economics (Brabazon, 2018). In these grammars, user-defined "if" statements are used to check which action should be taken and "then" statements are used to perform specific actions. For example, in the Sante Fe ant trail problem (in which artificial ants try to find pallets of food), user-defined actions, such as turn left and turn right, are used if the ants sense the food.

A grammar is called context-free if the non-terminals can be mapped using the production rules no matter the expressions around it. For example, if we use a grammar to create a mathematical function and we have two production rules that map a non-terminal to a single opening or closing parentheses, we could create functions where the parenthesis do not come in pairs (i.e. more opening than closing parenthesis or vice versa), and therefore the grammar is not context-free. One could make this grammar context-free by creating production rules that map to expressions between a pair of parentheses (e.g. a rule that evaluates to a non-terminal between parentheses: (*non-terminal*)). This is also done in the second production rule of the grammar in Fig. 3.

One of the main advantages of GE over conventional genetic programming is that the search space can be restricted and knowledge of the system can be incorporated into the production rules of the context-free grammar. For example, in Jeschke and De Schutter (2021) the production rules are used to ensure that the parameters of the parameterized control laws appear in the conditions of if-statements, which leads to an efficient optimization step during the training of the parameterized control laws.

3.3. Training of parameterized control laws

GE uses a set of input–output pairs to train the functions (i.e. the parameterized control laws) that map the inputs to outputs. In Jeschke and De Schutter (2021), input and output data is generated by simulating an MPC-controlled urban traffic network in a traffic simulator. The input data are the states of the prediction model per control time step and the outputs are the optimal phase times for every intersection and every control time step as determined by the MPC controller.

The training step of one generation of parameterized control laws consists of two parts. First, a new generation of parameterized control laws is generated using genetic operators (i.e., crossover and mutation), and secondly, for every parameterized control law, the parameters in that control law should be optimized on the training data. This means that for every data point and every parameterized control law, we need to optimize the parameters of the control law to estimate the output as closely as possible to the control inputs generated by conventional MPC. Note that this optimization process can be performed in a computationally efficient way, by selecting a faster solver or approximating the global optimum roughly. After the parameters are optimized for every parameterized control law, the fitness of the parameterized control laws is calculated based on an error between the outputs (based on the data collected via the traffic simulator) and the outputs computed via the parameterized control law. Based on this fitness, a percentage of the best-performing control laws are kept and new control laws are created for the rest of the population using genetic operators.

The most used genetic operators in GE are crossover and mutation (Ahmadizar et al., 2015; O'Neill & Ryan, 2001). Since the parameterized control laws generated with GE have an underlying decision tree representation, standard tree-based crossover and mutation from conventional genetic programming are used to evolve the population (Fenton et al., 2017).

3.4. Continuous grammars

In the grammar of Jeschke and De Schutter (2021), the parameters of the parameterized control laws appear in the if- and else-statements of the grammar, resulting in a discontinuous parameterized control law. In the second part of the training step (i.e./ optimizing the parameters in the control law for every data point), a simple grid search can then be used as only a limited number of combinations of the parameters are possible, resulting in different outcomes of the control law. The parameterized control law could thus be simply evaluated for every combination. A downside of the discontinuous control laws is that it also results in discontinuities in the optimization step of the parameterized MPC controller.

Therefore, in this work, we propose a grammar that determines parameterized control laws that are continuous functions of the traffic states and parameters. Defining the grammar as a set of production rules that determine a continuous parameterized control law can have negative consequences for the training time of the algorithm as a grid search will probably not result in the optimal performance of the constructed control laws. For every data point, a more complex optimization problem has to be solved to calculate the performance for that data point, which will most likely increase the training time. However, since the training process is offline and PMPC lowers the number of optimization variables, the optimization problem that has to be solved for every data point is expected to be computationally tractable. In addition, the optimization process can be accelerated as mentioned in previous section. Furthermore, as all the parameterized control laws in a generation are independent of each other, the fitness of a whole generation can be calculated in parallel. In the next section, we propose a grammar to create the continuous control laws and two training frameworks to train them.

4. GE-based parameterized control laws

In this section, the process of constructing a parameterized control law using GE is described. In particular, a grammar for creating continuous parameterized control laws and two training frameworks are proposed. Moreover, a projection-based saturation method is proposed to guarantee the constraints on the control inputs (i.e., constraints on the phase times of the intersection).

$$
\begin{aligned}
\langle y_{d,j} \rangle &::= \langle \text{expr} \rangle \\
\langle \text{expr} \rangle &::= \langle \text{expr} \rangle \langle \text{op} \rangle \left(\langle \text{expr} \rangle \right) \Big| \left(\langle \text{expr} \rangle \langle \text{op} \rangle \left(\langle \text{expr} \rangle \right) \right) \\
&\quad \Big| \langle \text{func} \rangle \left(\langle \text{expr} \rangle \right) \Big| \langle \text{var} \rangle \Big| \langle \text{theta} \rangle \langle \text{expr} \rangle \\
\langle \text{var} \rangle &::= n_{d,j}^{\text{ph}} \Big| q_{d,j}^{\text{ph}} \Big| \alpha_{d,j}^{\text{ph,ent}} \Big| \alpha_{d,j}^{\text{ph,arr}} \\
\langle \text{theta} \rangle &::= \theta_{d,1} \Big| \theta_{d,2} \\
\langle \text{op} \rangle &::= + \Big| - \Big| \times \Big| \text{div} \\
\langle \text{func} \rangle &::= x^2 \Big| \sqrt[3]{x} \Big| e^x
\end{aligned}
$$

Fig. 4. The GE-based grammar used to obtain the score of each phase based on (11).

4.1. Score-based green time allocation

As it was discussed earlier, several constraints should be enforced when applying the GE-based controller for traffic signal control. We propose a score-based structure to allocate the green length of each phase for control time step k, such that constraint (5) can be implicitly satisfied. We have:

$$g_{d,j}(k) = \frac{y_{d,j}}{\sum_{j=1}^{N_d^{\text{ph}}} y_{d,j}} \cdot g_{d,\text{tot}}(k), \tag{10}$$

where $y_{d,j}$ is a score for the jth phase of intersection d that describes the importance of a certain phase with respect to the other phases, and $g_{d,\text{tot}}$ is the total green time length of intersection d at each cycle (i.e., the cycle time minus the total yellow time). GE-based grammars are used to construct a, generally nonlinear, function to evaluate the score $y_{d,j}$, such that:

$$y_{d,j} = f_{\text{s}}\left(n_{d,j}^{\text{ph}}, q_{d,j}^{\text{ph}}, \alpha_{d,j}^{\text{ph,ent}}, \alpha_{d,j}^{\text{ph,arr}}, \theta_{d,1}, \theta_{d,2}\right) + \kappa_y, \tag{11}$$

where κ_y is a positive constant used to avoid a zero score, $n_{d,j}^{\text{ph}}, q_{d,j}^{\text{ph}}$, $\alpha_{d,j}^{\text{ph,ent}}, \alpha_{d,j}^{\text{ph,arr}}$ are respectively the total number of vehicles, queue lengths, and number of entering vehicles and arriving vehicles on the lanes that correspond to the jth phase of intersection d. In addition, $\theta_{d,1}$ and $\theta_{d,2}$ are the parameters that are present in the parameterized control law for corresponding intersection d. Here a maximum number of two parameters are used in the GE-based control law, but the number of parameters can be adjusted if needed. Note that the state values in (11) are the sums of the states corresponding to all the links for the same phase, instead of the average values used in Jeschke and De Schutter (2021). This is because the numbers of lanes of the different phases are not necessarily the same, and comparing the mean value of states may underestimate the congestion degree of the phase with more lanes.

The grammar shown in Fig. 4 is designed to train the score function. In the grammar, the starting tree indicates that the outcome of this grammar is a generally nonlinear function represented by ⟨expr⟩, and the production rule of ⟨expr⟩ has recursive elements, which enable a flexible structure of the outcome and a larger solution space. Thus the outcome function can either be a complex expression or a simple function. The non-terminal ⟨var⟩ includes all the available traffic states that are used to generate function $f_{\text{s}}(\cdot)$ in (11), while the non-terminal ⟨theta⟩ represents the parameters in the parameterized control law. The elements of the non-terminal ⟨op⟩ are the basic operators to construct the function, where div(·) is a modified division function that is used to avoid a zero division. We have

$$\text{div}(x) = \frac{1}{x+1}, \tag{12}$$

where x is assumed to be non-negative. The entries of the non-terminal ⟨func⟩ are used to introduce nonlinearity to the score function (11).

In contrast to the grammar in Jeschke and De Schutter (2021) that also exploits the score-based structure to decide green time length, the grammar used in this paper introduces more information, i.e., more states of the links, and the grammar can choose the states that are useful to construct the function. For example, any combination of the state variables $n_{d,j}^{\text{ph}}, q_{d,j}^{\text{ph}}, \alpha_{d,j}^{\text{ph,ent}}, \alpha_{d,j}^{\text{ph,arr}}$ can be realized by the grammar due to the second recursive rule in Fig. 4. Moreover, the parameters $\theta_{d,1}$ and $\theta_{d,2}$ are also present in the grammar, which means that the number of parameters (maximum 2 here) and the position of the parameters in the final expression are all adjustable during the learning process. These changes make the grammar in this paper more flexible than the one in Jeschke and De Schutter (2021) and allow for more diversity in the formulation of the score function, and thus a higher chance to generate a well-performing parameterized control law. To obtain a good result, the fitness function used to evaluate the parameterized control laws during training is also important and should be designed properly. Based on different formulations of the fitness function, two different training frameworks are proposed.

4.2. Framework-1: MPC-mimicking

The scheme of Framework-1 is shown in Fig. 5. The framework consists of two modules: the MPC module and the training module. The main idea is to train a parameterized control law that generates the control inputs that are as close as possible to the MPC controller. This is similar to the idea in Jeschke and De Schutter (2021). To start the training process, conventional MPC is first implemented on the target traffic network to generate an extensive data set. Each data point of the data set consists of a traffic state vector $\boldsymbol{x}_d^{\text{train}}(k)$ and a green time vector $\boldsymbol{g}_d^{\text{train}}(k)$ of a single intersection d at time step k, which are defined as:

$$\boldsymbol{x}_d^{\text{train}}(k) = \left[\boldsymbol{n}_d^{\text{ph}}(k)^\top, \boldsymbol{q}_d^{\text{ph}}(k)^\top, \boldsymbol{\alpha}_d^{\text{ph,ent}}(k)^\top, \boldsymbol{\alpha}_d^{\text{ph,arr}}(k)^\top\right]^\top, \tag{13}$$

$$\boldsymbol{g}_d^{\text{train}}(k) = \left[g_{d,1}(k), \ldots, g_{d,N_d^{\text{ph}}}(k)\right]^\top, \tag{14}$$

where $\boldsymbol{n}_d^{\text{ph}}(k) = \left[n_{d,1}^{\text{ph}}(k), \ldots, n_{d,N_d^{\text{ph}}}^{\text{ph}}(k)\right]^\top$, $\boldsymbol{q}_d^{\text{ph}}(k) = \left[q_{d,1}^{\text{ph}}(k), \ldots, q_{d,N_d^{\text{ph}}}^{\text{ph}}(k)\right]^\top$, $\boldsymbol{\alpha}_d^{\text{ph,ent}}(k) = \left[\alpha_{d,1}^{\text{ph,ent}}(k), \ldots, \alpha_{d,N_d^{\text{ph}}}^{\text{ph,ent}}(k)\right]^\top$, $\boldsymbol{\alpha}_d^{\text{ph,arr}}(k) = \left[\alpha_{d,1}^{\text{ph,arr}}(k), \ldots, \alpha_{d,N_d^{\text{ph}}}^{\text{ph,arr}}(k)\right]^\top$, and N_d^{ph} is the number of phases of intersection d. A data point $\boldsymbol{x}_d^{\text{train}}(k)$ is extracted from an intersection d at control time step k, and $\boldsymbol{g}_d^{\text{train}}(k)$ includes the corresponding green phase times generated by the conventional MPC controller. The data points corresponding to all the intersections in the traffic network during a simulation interval $[0, k_{\text{s}}t_{\text{s}})$ form the training data set that is expressed as:

$$\begin{aligned} S = \Big\{ &\left(\boldsymbol{x}_1^{\text{train}}(1), \boldsymbol{g}_1^{\text{train}}(1)\right), \ldots, \left(\boldsymbol{x}_1^{\text{train}}(k_{\text{s}}), \boldsymbol{g}_1^{\text{train}}(k_{\text{s}})\right), \ldots, \\ &\left(\boldsymbol{x}_{|J|}^{\text{train}}(1), \boldsymbol{g}_{|J|}^{\text{train}}(1)\right), \ldots, \left(\boldsymbol{x}_{|J|}^{\text{train}}(k_{\text{s}}), \boldsymbol{g}_{|J|}^{\text{train}}(k_{\text{s}})\right) \Big\}, \end{aligned} \tag{15}$$

where t_{s} is the simulation sampling time and k_{s} corresponds to the last control time step. Note that the dimensions of the data point vectors $\boldsymbol{x}_d^{\text{train}}(k)$ and $\boldsymbol{g}_d^{\text{train}}(k)$ may vary per intersection d if the intersections have different number of phases. This means different types of intersections should be trained separately, i.e., different types of intersections have different parameterized control laws.

Remark 1. Note that the data points in the training data set are assumed to be independent from each other for the training process, i.e., traffic at other intersections is assumed not to contribute to the phase times of the intersection considered in the data point. However, this is unrealistic in a real-world traffic network where the intersections have to coordinate and communicate with their neighbors to obtain a globally optimal performance. Training the control laws with only local data points may in general not result in a global optimum for the entire

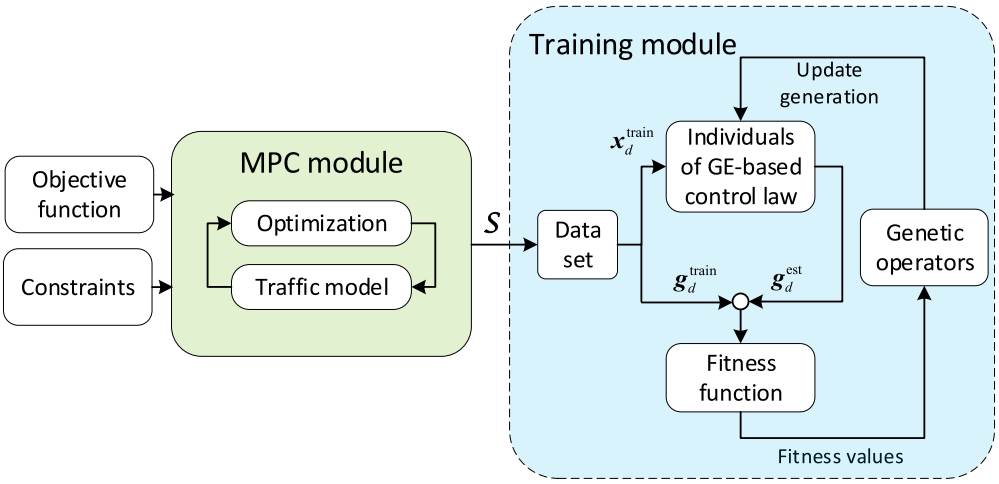


Fig. 5. The diagram of Framework-1.

traffic network. Therefore, it is necessary to include parameters in the local control laws, such that the parameters can be optimized to capture global information during the training process. This makes it possible to train the local controllers independently while mimicking the behavior of the centralized MPC controller.

The training data set is then given to the training module (see Fig. 5). First the state vector $\boldsymbol{x}_d^{\text{train}}(k)$ is fed into the GE-based controller block, where the score function is used to evaluate each phase based on the states and to determine the green time length according to (10). The output of this block is the estimated green time vector $\boldsymbol{g}_d^{\text{est}}(k) = [g_{d,1}^{\text{est}}(k), \dots, g_{d,N_d^{\text{ph}}}^{\text{est}}(k)]^\top$. The fitness function is defined as the mean square error between the true green time vector $\boldsymbol{g}_d^{\text{train}}(k)$ and the estimated green time vector $\boldsymbol{g}_d^{\text{est}}(k)$. As mentioned in Remark 1, the parameters in the control law need to be optimized to minimize the fitness value for every single data point, where the fitness value is given by:

$$\frac{1}{N_{\text{t}}} \sum_{d\in J} \sum_{k=0}^{k_{\text{s}}-1} \left[\min_{\theta_d} \left\| \boldsymbol{g}_d^{\text{train}}(k) - \boldsymbol{g}_d^{\text{est}}(k) \right\|_2^2 \right], \tag{16}$$

where $\boldsymbol{\theta}_d = [\theta_{d,1}, \theta_{d,2}]^\top$ that is included in $\boldsymbol{g}_d^{\text{est}}(k)$ implicitly, and N_{t} is the total number of data points. The mean of the fitness values of all the data points is the fitness value of the parameterized control laws. Note that multi-start points are used to find the global optimum due to the nonconvexity of the optimization problem.

The remaining training process is similar to genetic programming as described in Section 3. The evaluation and selection will continue until the last generation of parameterized control laws is generated, and the best parameterized control law will be selected. If there are multiple types of intersections (i.e., the intersections with different number of phases), the training process is repeated for different intersections with different training data sets.

The trained parameterized control laws will be used in the PMPC formulation (7), where only parameters that appear in the obtained GE-based control laws need to be optimized online by the PMPC controller. This training framework basically involves a regression problem, where the parameterized control laws are trained to mimic the behavior of a conventional MPC controller. Thus the training process is sensitive to the data set, which should be extensive enough to cover all kinds of traffic scenarios, but also compact enough to avoid a computationally expensive training.

Since the sum constraint (5) on the green time length is implicitly satisfied by the score-based allocation function (10), only the lower bound and upper bound constraints on the phase time lengths will be considered by the online PMPC controller. Note that during the training in Framework-1, no constraints on the control inputs are enforced. As long as the final mean square errors between $\boldsymbol{g}_d^{\text{train}}(k)$ and $\boldsymbol{g}_d^{\text{est}}(k)$ $\forall d \in J$ and $\forall k \in \{0, 1, \dots, k_{\text{s}} - 1\}$ are small enough, it is considered that the bound constraints on the estimated control inputs are satisfied. The constraints on control inputs during implementation of the PMPC controller can be enforced when solving optimization problem (7).

4.3. Framework-2: Prediction-based learning

Instead of mimicking the behavior of conventional MPC and training the control laws of different intersections independently, in Framework-2 we propose to directly optimize the global objective function for the entire traffic network during the training. The scheme of Framework-2 is shown in Fig. 6. In contrast to Framework-1, Framework-2 requires a data set including the state information only. The data set contains data points $\tilde{\boldsymbol{x}}^{\text{train}}(k)$ that is a column vector containing $\boldsymbol{n}_d^{\text{ph}}(k), \boldsymbol{q}_d^{\text{ph}}(k)$, $\boldsymbol{\alpha}_d^{\text{ph,ent}}(k), \boldsymbol{\alpha}_d^{\text{ph,arr}}(k)$ for all $d \in J$ and the control time step k, i.e.

$$\begin{aligned}\tilde{\boldsymbol{x}}^{\text{train}}(k) = \Big[&\boldsymbol{n}_1^{\text{ph}}(k)^\top, \boldsymbol{q}_1^{\text{ph}}(k)^\top, \boldsymbol{\alpha}_1^{\text{ph,ent}}(k)^\top, \boldsymbol{\alpha}_1^{\text{ph,arr}}(k)^\top, \dots, \\ &\boldsymbol{n}_{|J|}^{\text{ph}}(k)^\top, \boldsymbol{q}_{|J|}^{\text{ph}}(k)^\top, \boldsymbol{\alpha}_{|J|}^{\text{ph,ent}}(k)^\top, \boldsymbol{\alpha}_{|J|}^{\text{ph,arr}}(k)^\top, k \Big]^\top.\end{aligned} \tag{17}$$

The data points can be extracted from an open-loop simulation of the traffic network model, for the simulation interval $[0, k_{\text{s}} t_{\text{s}})$. Since each data point is used as the initial state of the traffic network during the training, any controller can be used to generate the data set. Note that the state information of all the intersections in the traffic network per control time step is included in each data point, and the parameterized control laws for all the intersections are trained simultaneously. The corresponding grammar should be slightly adjusted here. For example, if there are two types of intersections in a network, the starting tree needs to be changed to $S ::= \langle \text{expr} \rangle; \langle \text{expr} \rangle$, which implies that the output of the grammar includes two functions, i.e., two parameterized control laws are trained at the same time.

Each data point $\tilde{\boldsymbol{x}}^{\text{train}}$ is given to the inner iteration module in Fig. 6 as the initial state, and an MPC procedure is conducted based on the GE-based parameterized control law and the traffic model. The objective value is calculated over the prediction horizon N_{p} according to (7). Then this objective function is optimized for each data point, and used as the fitness value. Fitness value for each parameterized control law is given by:

$$\frac{1}{k_{\text{s}}} \sum_{k=0}^{k_{\text{s}}-1} \left[\min_{\boldsymbol{\theta}} \left(w_{\text{TTS}} J_{\text{TTS}}(k) + w_{\mathcal{D}} \mathcal{D}(\boldsymbol{g}(k, \boldsymbol{\theta})) + w_{\mathcal{Q}} \mathcal{Q}(k) \right) \right]. \tag{18}$$

In addition to the fact that the data points in Framework-2 contain more information, the built-in state evolution process within the inner

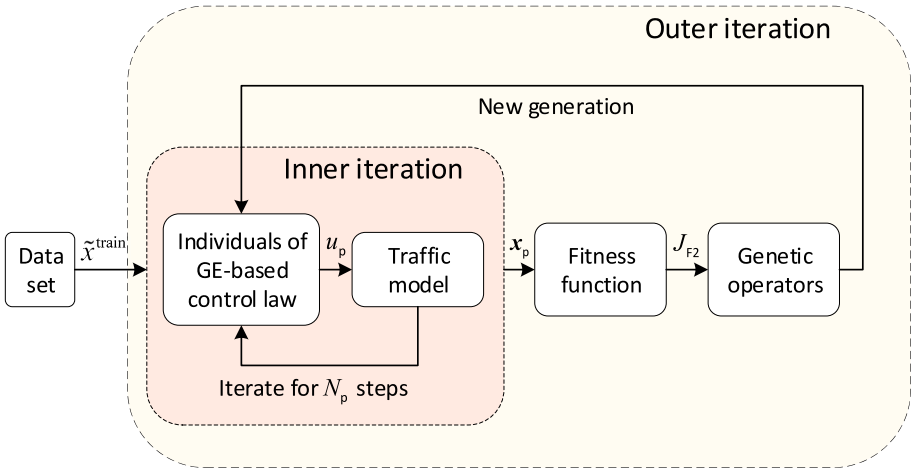


Fig. 6. The diagram of Framework-2, where u_p is the control input generated by the parameterized control law, $\boldsymbol{x}_p$ is the vector containing all the state information and control inputs that are used in the objective function (18), and J_{F2} is the optimized objective value of (18).

iteration module allows an initial state to generate a sequence of successive states, which makes Framework-2 more data-efficient and requiring a lower amount of training data points.

Remark 2. The MPC procedure in Framework-2 can be implemented in a rough way instead of a detailed way, in order to accelerate the training process without influencing the evaluation of the parameterized control laws. For example, the maximum number of iterations in the optimization can be set to a smaller value, or the number of multi-start points can be reduced for the nonconvex optimization problem.

Remark 3. Note that the number of parameters in the parameterized control law is adjustable for both Framework-1 and Framework-2. A larger number of parameters may lead to a better fitness value, while resulting in a higher computational complexity. In practice, it is recommended to choose a suitable number of parameters so that a balance between the complexity of the grammar and the fitness value is reached.

During the MPC procedure of the training process, the parameterized control laws for all intersections work together in a centralized way. However, optimizing all the parameters together also means a more complex optimization problem, which results in a longer training time. So the training process can be accelerated as mentioned in Remark 2. In addition, since the GE-based parameterized control laws are implemented directly on the traffic model, it is necessary to consider the lower bound and upper bound constraints on the control inputs explicitly. Therefore, in the next section, we propose a projection-based method to guarantee the bound constraints on the phase times.

4.4. Projection-based method for constraint satisfaction

Unlike conventional MPC, where bound constraints on the phase time can be addressed directly during optimization, in PMPC the bound constraint on the phase time introduces nonlinear constraint functions on the parameters $\boldsymbol{\theta}$. This increases the computational complexity of the resulting optimization problem. Therefore a projection-based method is used together with the phase time allocation function (10). After the phase times are calculated through (10), all the phase time values are projected into the feasible region where the constraints are ensured. Thus the phase times generated by the parameterized control law satisfy the bound constraints inherently, and the nonlinear constraint function is eliminated in the optimization problem. Therefore, the parameterized control law with projection-based method can be considered as a modified parameterized control law, for which the constraints on control inputs are always satisfied by construction.

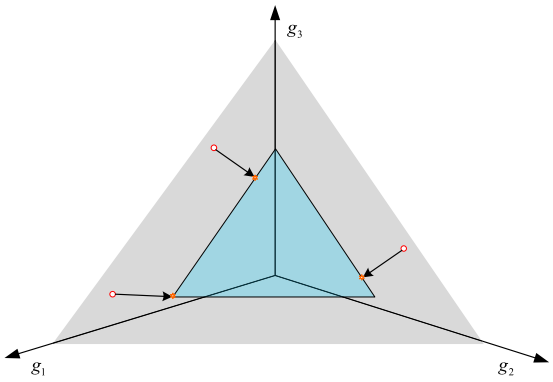


Fig. 7. Illustration of the projection-based method for a three-phase intersection to guarantee the phase time constraint: the gray plane is the area where the sum constraint is satisfied, while the blue region is where both the bound constraints and the sum constraint are satisfied. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

To illustrate the projection-based method, we consider a three-phase intersection. Assume the total green time length within one cycle is g_{tot}, then the calculated phase times g_1, g_2, g_3 satisfy the sum constraint:

$$g_1 + g_2 + g_3 = g_{\text{tot}}. \tag{19}$$

To further enforce the bound constraints on the phase time, the points (g_1, g_2, g_3) are projected to the feasible region as shown in Fig. 7 where the three axes represent g_1, g_2, g_3 respectively, and the gray plane consists of all the points that satisfy the sum constraint (19). The blue region represents the feasible region where lower and upper bound constraints on g_1, g_2, g_3 are satisfied. While all the calculated points (g_1, g_2, g_3) are in the gray plane, they are not necessarily within the blue region, such as the circle points shown in Fig. 7. Therefore, these points are projected to the blue region (see the star points shown in Fig. 7), where the points are projected to the corresponding closest points within the feasible region. In addition, the Euclidean distance between the original point and the projected point is also collected and used as a penalty on the constraint violation in the fitness function (18). With a given original point (g_1, g_2, g_3), both the projected point and the projection distance can be calculated in an analytical way. So the computational complexity of the projection method is low. Note that this method also applies for one intersection with more than three phases.

With the projection-based method, the explicit constraints on θ in the PMPC optimization problem (7) can therefore be removed. This accelerates the optimization process and also the training process of the parameterized control law. In addition, the projection-based method better than the explicit constraints in problem (7). This is illustrated next.

Proposition 4.1. *For the PMPC optimization problem (7), replacing the explicit constraints on parameters θ with the projection-based method results in equivalent or better performance in terms of the objective function.*

Proof. Let R_1 be the set of parameters θ that yield a feasible solution of the PMPC problem (7) with explicit constraints. Considering the projection-based method for a given form of the parameterized control law with parameters θ is in fact equivalent to considering a new modified control law with the same parameters θ for which the explicit constraints are always satisfied by construction (due to the projection). As such, the set R_2 of parameters θ for which the modified projection-based control law will yield a feasible solution of (7) will be a superset of R_1. As a result, optimizing over R_2 will result in an optimal performance that is not worse than the optimal performance over R_1. This proves the proposition. □

Note that this projection-based method can be applied for all PMPC control laws, including the relative-queue-length parameterized control law, the control law generated by Framework-1 or Framework-2, and also the training process of Framework-2. In addition, the proposed schemes can be applied and adjusted easily for the traffic signal control of any urban network with various layouts and phase settings. The schemes are designed for centralized control of the whole network (i.e., coordinating the local traffic signal controllers) to minimize the total time spent by the vehicles on the entire network. Nonetheless, the schemes can also be used for the training of a single intersection or a set of intersections of a sub-network, as well as for other performance criteria (e.g., minimization of emission or fuel/energy consumption). Furthermore, the idea behind the proposed methods, using GE to learn a state-feedback function for parameterized MPC, can be generalized to many other applications besides the field of traffic control.

5. Case study

In this section we compare the performance of the proposed GE-based PMPC controllers to the conventional MPC and the handcrafted PMPC controllers. The controllers are programmed using Matlab R2021a. The case study is carried out based on simulations in the traffic simulator SUMO (Lopez et al., 2018), and the interface TraCI (Wegener et al., 2008) is used to communicate between SUMO and Matlab. The measurements of the performance include the Total Time Spent (TTS) by the vehicles in the network and the computational complexity of the control methods. All the simulations run on a PC with an Intel Xeon Quad-Core E5- 1620 V3 CPU with a clock speed of 3.5 GHz.

5.1. Setup

An urban traffic network is considered as shown in Fig. 8 where the lengths of all the links are given. This network consists of 6 intersections denoted as A-E, and 6 source and sink nodes denoted by 1–6. There are two types of intersections: 2 (A and F) with 4 phases, and 4 (B,C,D,E) with 3 phases. The phases for different types of intersections are presented in Figs. 9 and 10. The cycle times for all the intersections are the same and set to 1 min. All 88 parameters of the traffic model, including the length and free-flow speed of each link and the saturation flow rate of each lane for all links are identified based on data collected from SUMO. The identified parameter values as well as the turning ratio values can be found in Appendix .

The traffic flow demand profiles are generated with SUMO's built-in route generator, which generates routes based on flow profiles and

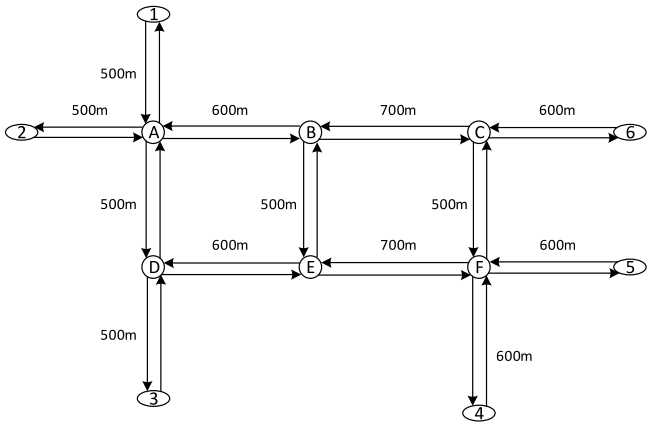


Fig. 8. The layout and the link lengths of the urban traffic network.

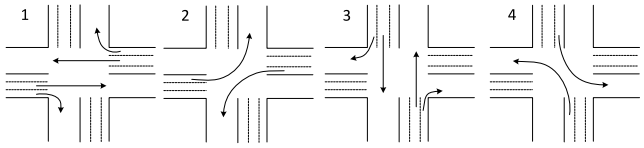


Fig. 9. The phases of 4-phase intersections.

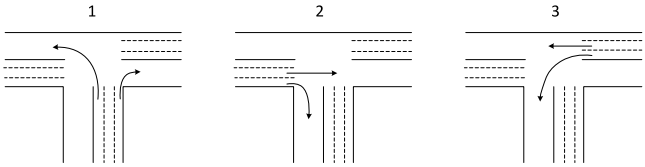


Fig. 10. The phases of 3-phase intersections.

turning rates. Three different demand profiles (see Fig. 11) are designed and applied in the traffic network to evaluate the performance of the controllers under various traffic conditions. All the three demand scenarios last for one hour. Before applying the demand profiles, the empty traffic network is initialized by running the network with a constant traffic flow of 1000 [veh/h] from all the source nodes for 30 min, in order to create a situation with heavy traffic with long queues on the lanes. The initial queue lengths of all the lanes are present in Table 1. The prediction horizon of all MPC controllers is 8 min, such that a vehicle that enters the network can leave the network within the prediction horizon considering the longest path and the red signal light. The control time step is 1 min. The cost functions for conventional MPC (6) and for PMPC (7) are identical, and the weights are $w_{\text{TTS}} = 1$, $w_{\mathcal{D}} = 1$, $w_{\mathcal{Q}} = 2$. The cost terms are defined as:

$$J_{\text{TTS}}(k) = \sum_{(u,d)\in L} \sum_{j=1}^{N_\text{p}} c_d \cdot n_{u,d}(k+j), \tag{20}$$

$$\mathcal{D}(\mathbf{g}(k)) = \Big\| \Big[(\mathbf{g}(k) - \mathbf{g}(k-1))^\top, (\mathbf{g}(k+1) - \mathbf{g}(k))^\top, \dots, (\mathbf{g}(k+N_\text{p}) - \mathbf{g}(k+N_\text{p}-1))^\top \Big]^\top \Big\|_2^2, \tag{21}$$

$$\mathcal{Q}(k) = \sum_{j=1}^{N_\text{p}} \sum_{d\in J} \max_{(u,d,o)\in L_d} q_{u,d,o}(k+j), \tag{22}$$

where L_d is the set of lanes that arrive at intersection d. For the PMPC controllers using projection-based method, an extra penalty term is added on the constraint violations of the control inputs:

$$\mathcal{V}(\mathbf{g}(k)) = d(\mathbf{g}(k)) + d(\mathbf{g}(k+1)) + \cdots + d(\mathbf{g}(k+N_\text{p}-1)), \tag{23}$$

where $d(\mathbf{g}(k)) = \sum_{d\in J} \left\| \mathbf{g}_d(k) - \mathbf{g}_d^{\text{pro}}(k) \right\|_2$ with $\mathbf{g}_d^{\text{pro}}(k)$ the projected phase times of $\mathbf{g}_d(k)$. The weight of the penalty term is $w_{\mathcal{V}} = 1$.

Table 1
The initialized queue lengths of all the lanes, where the queue values are given respectively in the order of left-turning, straight-going, and right-turning directions for each link, and the notation '–' means that term is not applicable for the corresponding direction.

	link(1,A)	link(2,A)	link(B,A)	link(D,A)	link(A,B)	link(C,B)	link(E,B)	link(B,C)	link(6,C)	link(F,C)
Queue [veh]	(35,39,44)	(40,35,44)	(45,53,52)	(27,40,46)	(–,79,63)	(61,81,–)	(47,–,21)	(–,51,34)	(16,27,–)	(12,–,4)
	link(A,D)	link(E,D)	link(3,D)	link(B,E)	link(D,E)	link(F,E)	link(C,F)	link(E,F)	link(4,F)	link(5,F)
Queue [veh]	(12,43,–)	(40,–,28)	(–,27,21)	(62,61,–)	(52,–,34)	(–,50,55)	(44,42,49)	(48,50,37)	(35,48,35)	(39,51,28)

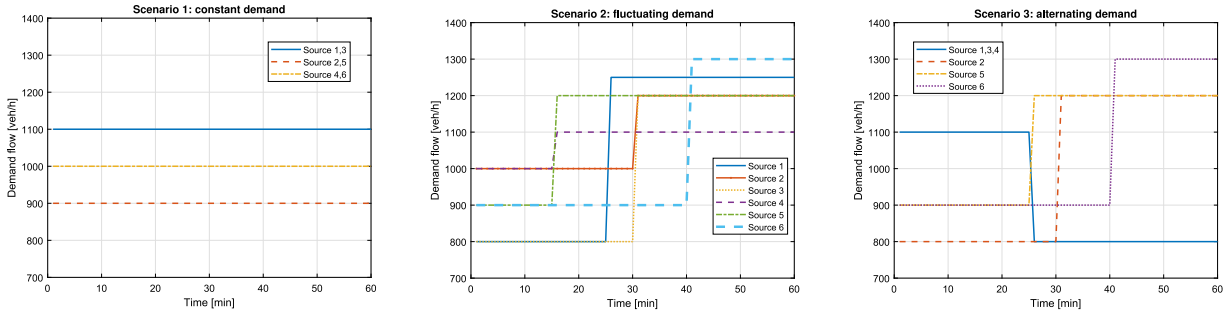


Fig. 11. Demand profiles of the three scenarios used in the case study.

5.2. Controllers

We compare the performance of five different controllers: fixed-time controller, conventional MPC controller, relative-queue-length (RQL) PMPC controller, GE-based Framework-1 (GE-F1) PMPC controller, and GE-based Framework-2 (GE-F2) PMPC controller. In addition, the RQL-PMPC controller is further divided into two methods: RQL-PMPC with explicit constraints in the optimization and RQL-PMPC with the projection-based method. With a cycle time of 1 min, the yellow time length after each green time phase is selected as 2 s. Therefore, the total green time within one cycle for the 3-phase intersections is 54 s, and the lower bound and upper bound on the phase times are 6 s and 42 s, respectively. For the 4-phase intersections, the total green time within one cycle is 52 s, and the lower bound and upper bound on the phase times are 6 s and 34 s.

Since the mathematical model of urban traffic networks are highly nonlinear and non-smooth, we found that the numerical solver `fmincon` function from Matlab optimization toolbox is more suitable to solve this specific problem. By comparing SQP and interior-point method, we found that SQP performs better in terms of convergence performance.[2] On the other hand, the main aim of this study is to show the relative improvement of the proposed methods in terms of computational efficiency, so we select single shooting to implement the nonlinear optimization for the sake of simplicity. Therefore, `fmincon` with SQP is used to solve the optimization problems for all the MPC controllers. Due to the non-convex optimization problem, multi-start optimization is used to approximate the global optimum. For this case study, our numerical experiments indicated that considering 10 starting points is enough to approximate the global optimum. So for all the optimization-based controllers, the optimization problem is solved 10 times with random starting points, and the best result is selected as the final solution. For the parameters of the convergence criteria of the solver, we have chosen to tune the tolerance of the objective function, the search step, and the constraints, respectively. The value 10^{-3} is chosen for all three parameters based on our experimental results such that a balance between computational efficiency and accuracy is reached.

[2] We also conducted experiments by using CasADi. Results showed that it provided comparable performance and CPU time to that of `fmincon` for this specific problem.

5.2.1. Fixed-time controller

For this controller, the total green time is distributed to each phase equally, and the phase times remain constant during the simulation interval. Therefore the phase times for the 4-phase and 3-phase intersections are 13 s and 18 s, respectively.

5.2.2. Conventional MPC controller

Since the sum of the phase times is fixed, the linear constraint (5) can be used to eliminate and substitute one of the phase times into the objective function. By using this strategy, the number of decision variables is reduced. Then there are 3 variables for each 4-phase intersection, and 2 variables for each 3-phase intersection. Since $N_{\mathrm{p}} = 8$, the number of parameters to be optimized every step of conventional MPC is reduced from 160 to 112. In addition, the lower bound and upper bound constraint on the phase times are included explicitly in the optimization.

5.2.3. RQL-PMPC controller

Using the parameterized control law (9), the parameters for each intersection $d \in J$ are $\theta_{d,1}$ and $\theta_{d,2}$. Thus the total number of parameters to be optimized every step of RQL-PMPC is 12, which is reduced significantly compared with the conventional MPC. To evaluate the effectiveness of the projection-based method, a comparison between RQL-PMPC with an explicit constraint (RQL-PMPC-EC) and RQL-PMPC with the projection-based method (RQL-PMPC-PC) is considered. For the RQL-PMPC-EC controller, the lower and upper bound constraints on the generated phase times are included in the optimization, while no constraints appear in the optimization problem for the RQL-PMPC-PC controller.

5.2.4. GE-F1 PMPC controller

Since there are two types of intersections in the traffic network, two parameterized control laws are trained separately using Framework-1. Conventional MPC is implemented using the S-model to generate the training data set. Six scenarios covering various traffic situations are considered, each lasting for 1 h. Thus there are in total 360 control time steps for all the intersections, resulting in 720 data points for the 4-phase intersections and 1440 data points for the 3-phase intersections.

For the training of grammar-based parameterized control law, the toolbox PonyGE2 (Fenton et al., 2017), which is implemented based on Python and is user-friendly due to its scalability and comprehensive instruction document, is used. The parameters used for the genetic

programming are: maximum number of generations = 50, and population size = 300. Moreover subtree mutation and crossover are used as the genetic operators. Another important parameter is the maximum depth limit for a derivation tree, which decides how complex the generated grammar will be. In this case, the value is set 10 to avoid a complex expression of the score function. These parameters are tuned on the basis of the default values, and are adjusted for this case study according to the experiments such that satisfying results are obtained while the training time is acceptable. For more information about the parameters of GE training, the reader is referred to Fenton et al. (2017).

As mentioned before, the parameters of the parameterized control law are optimized for each data point to find the optimal fitness value. The gradient-based algorithm L-BFGS is utilized to solve the optimization problem for every single data point. With $\kappa_y = 1$ in (11), the final obtained score function for 3-phase intersections is:

$$y_{d,j}^{\text{3-ph}} = \theta_{d,1}\left(\left(n_{d,j}^{\text{ph}} - \alpha_{d,j}^{\text{ph,arr}}\right)\cdot\frac{n_{d,j}^{\text{ph}}}{2\alpha_{d,j}^{\text{ph,ent}}+1}\cdot\left(\alpha_{d,j}^{\text{ph,arr}} + \theta_{d,2}\alpha_{d,j}^{\text{ph,arr}}\right)\right) + \theta_{d,2}\left(\left(\alpha_{d,j}^{\text{ph,arr}}\right)^2 + \alpha_{d,j}^{\text{ph,arr}} - \theta_{d,2}\frac{\alpha_{d,j}^{\text{ph,arr}}}{\alpha_{d,j}^{\text{ph,arr}}+1}\right) + 1. \tag{24}$$

The training fitness value of this control law is $1.46\cdot10^{-8}$, and the fitness value for the test data set is $4.78\cdot10^{-2}$, which are low enough fitness values for the 3-phase intersections. As mentioned in Remark 3, the maximum number of parameters is adjustable. For training of the 4-phase intersections, an extra parameter $\theta_{d,3}$ is added in order to further reduce the fitness value. The obtained GE-based scored function is:

$$y_{d,j}^{\text{4-ph}} = \theta_{d,3}n_{d,j}^{\text{ph}} + \theta_{d,2}\alpha_{d,j}^{\text{ph,ent}} - \theta_{d,3}^2\frac{\sqrt[3]{n_{d,j}^{\text{ph}}}}{\alpha_{d,j}^{\text{ph,arr}} + n_{d,j}^{\text{ph}} - \sqrt[3]{\alpha_{d,j}^{\text{ph,ent}}}+1} - \theta_{d,1}\left(n_{d,j}^{\text{ph}} - q_{d,j}^{\text{ph}}\right) + 1. \tag{25}$$

The training fitness value of this control law is $1.03\cdot10^{-1}$, and the testing fitness value is $3.69\cdot10^{-2}$. The control law is well-trained for the 4-phase intersections since the fitness values are small enough. Note that in score function (24) not all the state information is included in the function, e.g., queue state $q_{d,j}^{\text{ph}}$ is not selected by the grammar. In score function (25), all the parameters are used by the grammar. Therefore, a total number of 16 parameters are present in the parameterized control law. The projection-based method is used together with this control law.

5.2.5. GE-F2 PMPC controller

The control laws for the two types of intersections are trained simultaneously within Framework-2. Thus only one data set is needed. Since, as mentioned before Framework-2 is more data-efficient than Framework-1, a limited number of initial points are enough to train a well-performing parameterized control law. A total number of 45 data points are included in the data set, which is extensive enough to cover most traffic situations. The training setting is similar to that of the GE-F1 PMPC, except that the population size is reduced to 50 to reduce the training time. Similar to the RQL-PMPC controller, at most two parameters are allowed in this grammar for both parameterized control laws. According to the experiments, Powell's method is most efficient among all the available solvers within the toolbox and thus is used to optimize the parameters in the control law for each data point. The projection-based method is employed to remove the constraints on the parameters during both training and implementation of the parameterized control laws.

The obtained score function for the 3-phase intersections is:

$$y_{d,j}^{\text{3-ph}} = \sqrt[3]{\left(n_{d,j}^{\text{ph}}q_{d,j}^{\text{ph}}\right)^2 + \alpha_{d,j}^{\text{ph,arr}} - \theta_{d,1}\alpha_{d,j}^{\text{ph,arr}}\alpha_{d,j}^{\text{ph,ent}} + \theta_{d,2}n_{d,j}^{\text{ph}}} + 1, \tag{26}$$

and the obtained score function for the 4-phase intersections is:

$$y_{d,j}^{\text{4-ph}} = \sqrt[3]{q_{d,j}^{\text{ph}} + \frac{\theta_{d,2}\alpha_{d,j}^{\text{ph,ent}}}{\alpha_{d,j}^{\text{ph,ent}}+1}} + 1. \tag{27}$$

It is worth mentioning that the score function (26) is simpler than (24), and (26) is even more compact with only two states $q_{d,j}^{\text{ph}}$ and $\alpha_{d,j}^{\text{ph,ent}}$ and one parameter $\theta_{d,2}$. The performance of these controllers is compared and the results are analyzed in the next section.

5.3. Results and discussion

In Table 2 the system performance and computational complexity of the different controllers are shown for the different demand scenarios. The performance measurements TTS^{rel} and CT^{rel} are the relative change of the TTS and of the *mean* computation time (CT) with respect to the conventional MPC controller, where the mean computation time is considered for all the control time steps over the simulation interval. First, for this case study the experiments indicate that compared to the conventional MPC controller all the PMPC controllers reduce the computational complexity of the online optimization with more than 80% for all scenarios, while the system performance decreases only up to about 1% except for the GE-F1 PMPC controller. Second, all the controllers provide a better TTS than the fixed-time case, except for GE-F1 PMPC in scenario 1, which yields a similar performance as the fixed-time controller. The reason why GE-F1 PMPC performs relatively worse is that the training of Framework-1 requires a high-quality training data set. A proper training data set is necessary to avoid overfitting and to guarantee the performance of the generated control laws for various scenarios. However, it is usually difficult to guarantee the quality of the training data in practice. In contrast, the GE-F2 PMPC controller performs better than the GE-F1 PMPC controller for all three scenarios, in terms of both system performance and computation time, with a much smaller training data set. Compared with conventional MPC, GE-F2 PMPC can provide a comparable performance with significantly less CPU time for all the considered scenarios. In view of the advantages of Framework-2 mentioned before, it is recommended to use GE-F2 PMPC for further applications, as long as enough training is allowed.

The RQL-PMPC controller using the projection-based method outperforms the one using the explicit constraint for all the scenarios in terms of both performance measures, which confirms the effectiveness of the proposed method. The RQL-PMPC controller performs slightly better than GE-F2 PMPC, because the former is constructed based on expert knowledge and has been fine tuned through experiments, whereas GE-F2 PMPC is created automatically with limited expert knowledge of the system. Therefore, the differences in the performance are acceptable. Note that the conventional MPC controller performs slightly worse than the PMPC controllers for scenario 3. This is most probably due to the mismatch between the prediction model and the simulator, where the conventional MPC controller seems to be more sensitive to model uncertainties. Another possible reason is that for conventional MPC it might be more difficult to obtain a good approximation of a globally optimal solution within the same computation time budget as PMPC for every single optimization process, as the large number of parameters of conventional MPC makes the corresponding optimization problem more complex.

6. Conclusions

In this paper efficient parameterized MPC (PMPC) approaches have been introduced for urban traffic signal control. In addition to the handcrafted relative-queue-length PMPC, grammatical-evolution (GE)-based PMPC has been proposed to generate the parameterized control laws automatically with limited expert knowledge. Therefore, it is possible to find well-performing parameterized control laws, which can easily be adjusted to meet specific requirements by changing the grammar structure. Two training frameworks for GE-based PMPC have been proposed and compared. Moreover, a projection-based method for removing the nonlinear constraints on the parameters of PMPC controllers has been introduced. The SUMO-based simulation results

Table 2
The TTS, mean and maximum computation time, and the relative change in TTS and mean computation time with respect to the conventional MPC controller for the different demand scenarios and controllers.

Demand scenario	Controller	TTS [veh h]	TTS^{rel} [%]	Mean computation time [s]	Max computation time [s]	CT^{rel} [%]
Scenario 1	Fixed-time	1284.6	12.21	–	–	–
	Conventional MPC	1144.8	–	21.35	26.91	–
	RQL-PMPC-EC	1146.1	0.11	3.33	5.34	−84.40
	RQL-PMPC-PC	1145.2	0.00	1.93	3.22	−90.96
	GE-F1 PMPC	1252.4	9.40	3.13	7.72	−85.34
	GE-F2 PMPC	1146.0	0.10	2.16	5.06	−89.88
Scenario 2	Fixed-time	1239.3	20.34	–	–	–
	Conventional MPC	1029.9	–	22.36	25.05	–
	RQL-PMPC-EC	1034.2	0.42	3.27	5.24	−85.38
	RQL-PMPC-PC	1033.4	0.34	1.89	3.08	−91.55
	GE-F1 PMPC	1064.8	3.39	3.74	6.31	−83.27
	GE-F2 PMPC	1040.4	1.02	2.23	4.26	−90.03
Scenario 3	Fixed-time	1192.3	11.88	–	–	–
	Conventional MPC	1065.7	–	22.37	29.81	–
	RQL-PMPC-EC	1051.3	−1.35	4.46	12.39	−80.06
	RQL-PMPC-PC	1049.5	−1.52	1.65	4.11	−92.62
	GE-F1 PMPC	1075.1	0.88	3.18	5.03	−85.78
	GE-F2 PMPC	1054.2	−1.08	2.00	5.02	−91.06

show that PMPC controllers reduce the online computation time significantly, and achieve a performance that is comparable with that of the conventional MPC controller. It is also demonstrated that the projection-based method further improves the computational efficiency of PMPC controllers. In our case study, the GE-based PMPC controllers perform as well as the handcrafted PMPC controller, which outperforms the GE-based PMPC controller of Jeschke and De Schutter (2021). Framework-2 proves to be better than Framework-1 by generating a more concise control law, which improves the computational efficiency and the system performance.

In the future, the proposed GE-based PMPC approach together with the projection-based method will also be adapted to more complex traffic networks that contain more intersection types and also considers more cost terms (e.g., emissions). In addition, distributed GE-based PMPC controller will be developed to deal with larger-scale networks. Learning-based approaches such as reinforcement learning can also be incorporated to deal with model uncertainties and external disturbances. Moreover, an in depth comparison study of different solution methods can be carried out.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix. Identified parameters

The calibrated parameter values of the urban traffic network as well as the given turning ratio values in the case study are presented in Table A.3.

Table A.3
The identified parameters of the traffic network in the case study, where the saturation flow rates and turning ratios are given in the order of left-turning, straight-going, and right-turning directions for each link, respectively, and the notation '–' means that term is not applicable for the corresponding direction.

	link(1,A)	link(2,A)	link(B,A)	link(D,A)
Free speed [m/s]	34.1358	10.1425	74.6448	54.4716
Link length [m]	570.414	277.594	567.736	374.167
Saturation flow rate [veh/h]	(2199.4,2390.3,1813.5)	(2032.4,2129.3,1851.5)	(2530.9,2332.6,1823.6)	(2120.2,1926.1,2269.7)
Turning ratio	(0.34,0.32,0.34)	(0.33,0.34,0.33)	(0.36,0.36,0.28)	(0.33,0.34,0.33)
	link(A,B)	link(C,B)	link(E,B)	link(B,C)
Free speed [m/s]	84.4651	62.0863	72.6384	83.5817
Link length [m]	667.691	732.577	454.815	598.724
Saturation flow rate [veh/h]	(-,2411.5,1906.6)	(2365.7,2435.9,-)	(2300.9,-,1886.6)	(-,2229.2,2026.9)
Turning ratio	(-,0.57,0.43)	(0.51,0.49,-)	(0.53,-,0.47)	(-,0.53,0.47)
	link(6,C)	link(F,C)	link(A,D)	link(E,D)
Free speed [m/s]	22.6800	69.6748	82.6973	78.4553
Link length [m]	757.118	563.404	709.551	785.791
Saturation flow rate [veh/h]	(2307.3,2269.0,-)	(2557.5,-,1700.8)	(2145.9,2402.4,-)	(2094.3,-,1860.9)
Turning ratio	(0.45,0.55,-)	(0.6,-,0.4)	(0.5,0.5,-)	(0.54,-,0.46)
	link(3,D)	link(D,E)	link(B,E)	link(F,E)
Free speed [m/s]	29.5055	59.9165	96.7607	61.3621
Link length [m]	447.165	521.882	605.750	647.780
Saturation flow rate [veh/h]	(-,2118.9,1880.4)	(2134.5,2462.8,-)	(2304.9,-,1914.9)	(-,2179.5,2075.1)
Turning ratio	(-,0.54,0.46)	(0.5,0.5,-)	(0.54,-,0.46)	(-,0.53,0.47)
	link(E,F)	link(C,F)	link(5,F)	link(4,F)
Free speed [m/s]	86.7538	81.4044	14.9533	94.6695
Link length [m]	636.439	666.894	501.306	608.948
Saturation flow rate [veh/h]	(2612.8,2111.7,1865.0)	(1979.0,2414.5,2221.8)	(2469.7,2219.0,1637.1)	(2131.2,2461.3,1790.7)
Turning ratio	(0.35,0.34,0.31)	(0.33,0.34,0.33)	(0.34,0.34,0.32)	(0.34,0.32,0.34)

References

Afram, A., & Janabi-Sharifi, F. (2014). Theory and applications of HVAC control systems–A review of model predictive control (MPC). *Building and Environment*, *72*, 343–355.

Ahmadizar, F., Soltanian, K., Akhlaghian Tab, F., & Tsoulos, I. (2015). Artificial neural network development by means of a novel combination of grammatical evolution and genetic algorithm. *Engineering Applications of Artificial Intelligence*, *39*, 1–13.

Brabazon, A. (2018). Grammatical-evolution in finance and economics: A survey. In *Handbook of grammatical evolution* (pp. 263–288). Springer.

Cagienard, R., Grieder, P., Kerrigan, E., & Morari, M. (2007). Move blocking strategies in receding horizon control. *Journal of Process Control*, *17*(6), 563–570.

Fenton, M., McDermott, J., Fagan, D., Forstenlechner, S., Hemberg, E., & O'Neill, M. (2017). Ponyge2: Grammatical evolution in python. In *Proceedings of the genetic and evolutionary computation conference companion* (pp. 1194–1201).

Gartner, N. (1983). *OPAC: A demand-responsive strategy for traffic signal control, no. 906*. Transportation Research Board.

Goulart, P., Kerrigan, E., & Maciejowski, J. (2006). Optimization over state feedback policies for robust control with constraints. *Automatica*, *42*(4), 523–533.

Hemberg, E., Ho, L., O'Neill, M., & Claussen, H. (2013). A comparison of grammatical genetic programming grammars for controlling femtocell network coverage. *Genetic Programming and Evolvable Machines*, *14*(1), 65–93.

Henry, J., Farges, J., & Tuffal, J. (1984). The PRODYN real time traffic algorithm. In *Control in transportation systems* (pp. 305–310). Elsevier.

Hopcroft, J., Motwani, R., & Ullman, J. (2006). *Introduction to automata theory, languages, and computation* (3rd ed.). Addison-Wesley.

Hunt, P., Robertson, D., Bretherton, R., & Royle, M. (1982). The SCOOT on-line traffic signal optimisation technique. *Traffic Engineering & Control*, *23*(4), 190–192.

Jamshidnejad, A., Lin, S., Xi, Y., & De Schutter, B. (2019). Corrections to "Integrated urban traffic control for the reduction of travel delays and emissions". *IEEE Transactions on Intelligent Transportation Systems*, *20*(5), 1978–1983.

Jeschke, J., & De Schutter, B. (2021). Parametrized model predictive control approaches for urban traffic networks. In *Proceedings of the 16th IFAC symposium on control in transportation systems*.

Koza, J. (1994). Genetic programming as a means for programming computers by natural selection. *Statistics and Computing*, *4*(2), 87–112.

Lin, S., De Schutter, B., Xi, Y., & Hellendoorn, H. (2011). Fast model predictive control for urban road networks via MILP. *IEEE Transactions on Intelligent Transportation Systems*, *12*(3), 846–856.

Lin, S., De Schutter, B., Xi, Y., & Hellendoorn, H. (2012). Efficient network-wide model-based predictive control for urban traffic networks. *Transportation Research Part C (Emerging Technologies)*, *24*, 122–140.

Lin, S., & Xi, Y. (2008). An efficient model for urban traffic network control. *IFAC Proceedings Volumes*, *41*(2), 14066–14071.

Little, J., Kelson, M., & Gartner, N. (1981). *MAXBAND: A versatile program for setting signals on arteries and triangular networks*. Sloan School of Management, Massachusetts Institute of Technology.

Lofberg, J. (2003). Approximations of closed-loop minimax MPC. In *42nd IEEE international conference on decision and control, vol. 2* (pp. 1438–1442). IEEE.

Lopez, P. A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flotterod, Y. P., Hilbrich, R., Lucken, L., Rummel, J., Wagner, P., & Wiebner, E. (2018). Microscopic traffic simulation using SUMO. In *IEEE intelligent transportation systems conference, vol. 21* (pp. 2575–2582). Maui, Hawaii: IEEE.

Mirchandani, P., & Head, L. (2001). A real-time traffic signal control system: architecture, algorithms, and analysis. *Transportation Research Part C (Emerging Technologies)*, *9*(6), 415–432.

Nicolau, M., & Agapitos, A. (2018). Understanding grammatical evolution: Grammar design. In *Handbook of grammatical evolution* (pp. 23–53). Springer.

Nicolau, M., O'Neill, M., & Brabazon, A. (2012). Termination in grammatical evolution: Grammar design, wrapping, and tails. In *IEEE congress on evolutionary computation* (pp. 1–8).

O'Neill, M., & Ryan, C. (2001). Grammatical evolution. *IEEE Transactions on Evolutionary Computation*, *5*(4), 349–358.

Pippia, T., Sijs, J., & De Schutter, B. (2018). A parametrized model predictive control approach for microgrids. In *Proceedings of the 57th IEEE conference on decision and control* (pp. 3171–3176).

Poli, R., Langdon, W. B., McPhee, N. F., & Koza, J. (2008). *A field guide to genetic programming*. Lulu.com.

Qin, S., & Badgwell, T. (2003). A survey of industrial model predictive control technology. *Control Engineering Practice*, *11*(7), 733–764.

Rawlings, J., & Mayne, D. (2009). *Model predictive control: Theory and design*. Nob Hill Publishing.

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* (pp. 432–439). IEEE.

Robertson, D. (1986). Research on the TRANSYT and SCOOT methods of signal coordination. *ITE Journal*, *56*(1), 36–40.

Ryan, C., O'Neill, M., & Collins, J. (2018). Introduction to 20 years of grammatical evolution. In *Handbook of grammatical evolution* (pp. 1–21). Springer.

Sims, A. (1979). The sydney coordinated adaptive traffic system. In *Engineering foundation conference on research directions in computer control of urban traffic systems*.

Tsoulos, I., Gavrilis, D., & Glavas, E. (2008). Neural network construction and training using grammatical evolution. *Neurocomputing*, *72*(1–3), 269–277.

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 *IEEE 20th international conference on intelligent transportation systems* (pp. 1–6). Yokohama, Japan: IEEE.

Webster, F. (1958). *Traffic signal settings*: *Technical report*, Road Research Laboratory.

Wegener, A., Piórkowski, M., Raya, M., Hellbrück, H., Fischer, S., & Hubaux, J. (2008). TraCI: an interface for coupling road traffic and network simulators. In *Proceedings of the 11th communications and networking simulation symposium* (pp. 155–163).

Ye, B., Wu, W., Li, L., & Mao, W. (2016). A hierarchical model predictive control approach for signal splits optimization in large-scale urban road networks. *IEEE Transactions on Intelligent Transportation Systems*, *17*(8), 2182–2192.

Ye, B., Wu, W., & Mao, W. (2015). Distributed model predictive control method for optimal coordination of signal splits in urban traffic networks. *Asian Journal of Control*, *17*(3), 775–790.

Ye, B., Wu, W., Ruan, K., Li, L., Chen, T., Gao, H., & Chen, Y. (2019). A survey of model predictive control methods for traffic signal control. *IEEE/CAA Journal of Automatica Sinica*, *6*(3), 623–640.

Zegeye, S., De Schutter, B., Hellendoorn, H., Breunesse, E., & Hegyi, A. (2012). A predictive traffic controller for sustainable mobility using parameterized control policies. *IEEE Transactions on Intelligent Transportation Systems*, *13*(3), 1420–1429.