MULTI-LEVEL AND LEARNING-BASED MODEL PREDICTIVE CONTROL FOR TRAFFIC MANAGEMENT

MULTI-LEVEL AND LEARNING-BASED MODEL PREDICTIVE CONTROL FOR TRAFFIC MANAGEMENT

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. dr. ir. T.H.J.J. van den Hagen,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op
dinsdag 3 oktober 2023 om 12:30 uur

door

Dingshan SUN

Master of Science in Aeronautical and Astronautical Science and Technology,
Shanghai Jiao Tong University, China,
geboren te Xiantao, Hubei, China.

Dit proefschrift is goedgekeurd door de

promotor: Prof. dr. ir. B. De Schutter
copromotor: Dr. A. Jamshidnejad

Samenstelling promotiecommissie:

Rector Magnificus,	voorzitter
Prof. dr. ir. B. De Schutter	Technische Universiteit Delft
Dr. A. Jamshidnejad	Technische Universiteit Delft
Onafhankelijke leden:	
Prof. dr. A. Ferrara	University of Pavia
Prof. dr. C. De Persis	Rijksuniversiteit Groningen
Prof. dr. ir. H. van Lint	Technische Universiteit Delft
Prof. dr. ir. M. M. de Weerdt	Technische Universiteit Delft
Prof. dr. N. Geroliminis	École Polytechnique Fédérale de Lausanne

The research described in this thesis was supported by China Scholarship Council under grant No. 201806230254.

TRAIL Thesis Series T2023/16, The Netherlands TRAIL Research School
P.O. Box 5017
2600 GA Delft, The Netherlands
Tel: +31 (0) 15 278 6046
E-mail: info@rstrail.nl

Cover designed by: P. Sun
Published and distributed by: D. Sun
E-mail: dingshan_1994@163.com

Keywords: Traffic Management, Model Predictive Control, Learning-Based MPC, Multi-Level MPC, Reinforcement Learning

ISBN 978-90-5584-335-0

Copyright © 2023 by D. Sun

All research data and code supporting the findings described in this dissertation are available in 4TU.Centre for Research Data at:
`https://doi.org/10.4121/97751b97-e421-426b-b09a-d7b975e0f2b0.v1`.

An electronic version of this dissertation is available at
`http://repository.tudelft.nl/`.

World peace

CONTENTS

Summary **xi**

Samenvatting **xiii**

Preface **xvii**

1 Introduction **1**
- 1.1 Background 2
  - 1.1.1 Urban traffic control 2
  - 1.1.2 Freeway traffic control 3
- 1.2 Challenges 4
- 1.3 Contributions 6
- 1.4 Outline 7

2 A Novel Bi-Level Temporally-Distributed MPC Approach **9**
- 2.1 Introduction and motivations 10
  - 2.1.1 Model predictive control (MPC) 10
  - 2.1.2 Current challenges of MPC for green urban mobility 11
  - 2.1.3 Contributions & structure of this chapter 12
  - 2.1.4 Related work 14
- 2.2 Green urban mobility based on an annual MPC schedule 15
  - 2.2.1 Cumulative Constraints 15
  - 2.2.2 Problem formulation 17
  - 2.2.3 Characteristics of the optimization problem 18
- 2.3 Proposed methodology for tackling the optimization complexity resulting from various temporal scales 19
  - 2.3.1 Bi-level temporal distribution of the problem 19
- 2.4 Case study 25
  - 2.4.1 Setup for case study 1 26
  - 2.4.2 Controllers 28
  - 2.4.3 Setup for case study 2 32
  - 2.4.4 Results and Discussion 32
- 2.5 Conclusions 34

3 GE-Based Parameterized MPC for Urban Traffic Networks **39**
- 3.1 Introduction 40
  - 3.1.1 Traffic signal control 40
  - 3.1.2 Model Predictive Control for urban traffic networks 40
  - 3.1.3 Main contributions 41
  - 3.1.4 Outline of the chapter 42

3.2 Parameterized model predictive urban traffic control 42
  3.2.1 Urban traffic prediction model 42
  3.2.2 Parameterized Model Predictive Control 44
3.3 Grammatical Evolution 46
  3.3.1 Genetic programming 46
  3.3.2 GE with context-free grammars 47
  3.3.3 Training of parameterized control laws 48
  3.3.4 Continuous grammars 49
3.4 GE-based parameterized control laws 49
  3.4.1 Score-based green time allocation 49
  3.4.2 Framework-1: MPC-mimicking 51
  3.4.3 Framework-2: Prediction-based learning 53
  3.4.4 Projection-based method for constraint satisfaction 55
3.5 Case study 57
  3.5.1 Setup 57
  3.5.2 Controllers 60
  3.5.3 Results and discussion 63
3.6 Conclusions 64

4 Combined MPC and DRL framework for Freeway Control **67**
4.1 Introduction 68
4.2 Related work 70
  4.2.1 MPC for freeway traffic control 70
  4.2.2 DRL for freeway traffic control 72
  4.2.3 Combined MPC and DRL methods 74
4.3 Combined MPC-DRL framework 77
  4.3.1 MPC-DRL framework 78
  4.3.2 Detailed description of the framework 80
  4.3.3 Algorithm for training the framework 83
4.4 Case study 85
  4.4.1 Setup 86
  4.4.2 Controllers 87
  4.4.3 Results for the learning process 89
  4.4.4 Results for the implementations 89
4.5 Conclusions 93

5 Adaptive Parameterized MPC based on RL: A Synthesis Framework **95**
5.1 Introduction & Motivation 96
5.2 Related work 97
  5.2.1 PMPC 97
  5.2.2 Learning-based adaptive MPC 98
  5.2.3 RL-based adaptive MPC 99
5.3 The synthesis framework of RL-based adaptive PMPC 101
  5.3.1 Extended PMPC scheme 101
  5.3.2 The synthesis framework 103

5.4 Case study 110
5.4.1 Freeway network 110
5.4.2 Parameterized freeway traffic control laws 111
5.4.3 Controllers 112
5.4.4 Results and discussion 115
5.5 Conclusions 116

6 Conclusions and recommendations 123
6.1 Conclusions 124
6.2 Impacts of this thesis 125
6.2.1 Social impacts 125
6.2.2 Scientific and technical impacts 125
6.3 Recommendations for future research 126
6.3.1 Recommendations in terms of applications 126
6.3.2 Recommendations in terms of Theory 127

Bibliography 129

Curriculum Vitæ 145

TRAIL Thesis Series 147

SUMMARY

Traffic networks are a major component of modern society, contributing to more efficient traveling and transportation, which significantly facilitates social productivity and well-being. However, as more vehicles are appearing on the roads, traffic volumes exceed the maximal road capacity and traffic congestion occurs easily, especially during rush hours. This leads to serious negative impacts, including noise, air pollution, waste of fuel, and increased travel time. Therefore, developing efficient traffic management approaches is of great importance and urgency.

Model predictive control (MPC) as a mature optimization-based control method has been applied successfully in industry, as well as the field of traffic management. However, in order to guarantee control performance, MPC requires a sufficiently accurate model, which is often not available for traffic networks. In addition, the computational complexity of MPC makes it hard to implement in real time, especially for large-scale traffic networks. Therefore, current MPC methods can still be further improved in terms of control performance and computational efficiency. Reinforcement learning (RL) has recently been widely used in control problems, including traffic management, since it can naturally address uncertainties and changing environments with negligible online computational resources. However, RL also suffers from its own shortcomings, such as low sample efficiency and safety issues. The features of MPC and RL complement each other very well. Therefore, part of this thesis considers combining MPC and RL to further improve the control performance.

This thesis focuses on management and control of traffic networks, including urban networks and freeway networks. More specifically, we aim to reduce traffic congestion by minimizing the total time spent of all the vehicles in the network, and also consider green mobility by minimizing the total emissions produced by the vehicles. In this thesis, we utilize MPC-based and learning-based approaches to achieve this goal, which can be further divided into four parts as presented below.

In the first part, we consider MPC for a long-term green urban mobility problem with cumulative emission constraints, where the optimization horizon of the MPC problem is significantly larger than the control sampling time. As a result, the number of the variables that should be optimized per control time step becomes very large. This results in optimization problems that are computationally intractable in real time. To address this issue, we propose a novel bi-level temporally distributed MPC structure, which includes a short-term and a long-term MPC formulation with small and large control sampling times that will be solved jointly, instead of the original complex optimization problem. The resulting bi-level control approach is used to solve the MPC problem online for real-time control of urban traffic networks with the objective of long-term green mobility. Simulation on an urban traffic network shows that the proposed bi-level MPC approach outperforms other conventional control methods, in terms of the total time spent, total emissions of CO_2, and computation time.

In the second part, in order to reduce the computational complexity of MPC, we propose to use parameterized MPC (PMPC) for traffic signal control of urban networks. PMPC is an efficient MPC approach that reduces the number of optimization variables by using a parameterized control law instead of directly optimizing the control inputs. But the design of the parameterized control law in general requires expert knowledge. In this part, we propose to use grammatical evolution to construct continuous parameterized control laws automatically, 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 training framework is implemented on an urban traffic network, and a parameterized control law is generated. SUMO-based simulation results show that the resulting PMPC controller reduces the online computation time significantly, and achieves a performance that is comparable with that of the conventional MPC controller.

In the third part, we consider control of freeway networks by combining MPC and deep reinforcement learning (DRL), in order to deal with uncertainties and disturbances. We propose a novel framework for integrating MPC and DRL methods for freeway traffic control that is different from existing MPC-RL methods. More specifically, the proposed framework adopts a hierarchical structure, where a high-level efficient MPC component works at a low frequency with a large control sampling time to provide a baseline control input, while the DRL component works at a high frequency to modify online the control input generated by MPC. The control framework, therefore, needs only limited online computational resources and is able to handle uncertainties and external disturbances after proper learning with enough training data. The proposed framework is implemented on a benchmark freeway network in order to coordinate ramp metering and variable speed limits. Simulation results show that the proposed framework outperforms both standard MPC and DRL approaches, in terms of the total time spent and constraint satisfaction, despite model uncertainties and external disturbances.

In the fourth part, the PMPC scheme is combined with RL to address the two main issues of conventional MPC simultaneously. We propose a novel framework that uses RL to adapt the PMPC scheme online, which integrates all the possible strategies to adjust different components of PMPC (e.g., objective function, state-feedback control law, optimization settings, and system model). This results in a synthesis framework for RL-based adaptive PMPC. Even the existing adaptive (P)MPC approaches can be embedded in this synthesis framework. The resulting combined PMPC-DRL framework provides an efficient MPC approach that can deal with model mismatches. The proposed framework is applied for freeway traffic control, and simulation results that the RL-based adaptive PMPC achieves a better performance than standalone RL-based controller and conventional MPC, in terms of total time spent and computation time.

In summary, this thesis overcomes the shortcomings of conventional MPC by proposing several frameworks, including the bi-level temporally-distributed MPC, grammatical evolution-based PMPC, combined DRL-MPC, and RL-based adaptive PMPC frameworks. These frameworks are applied for traffic management of urban networks and freeway networks and achieve improved performance compared to conventional methods, under disturbances and uncertainties in the environment.

SAMENVATTING

Verkeersnetwerken zijn een belangrijk onderdeel van de moderne samenleving en dragen bij tot efficiënter reizen en vervoer, wat de maatschappelijke productiviteit en het welzijn aanzienlijk bevordert. Naarmate er echter meer voertuigen op de wegen verschijnen, overschrijdt het verkeersvolume de maximale wegcapaciteit en ontstaan er gemakkelijk files, vooral tijdens de spitsuren. Dit leidt tot ernstige negatieve gevolgen, zoals lawaai, luchtverontreiniging, brandstofverspilling en langere reistijden. Daarom is het van groot belang en dringend noodzakelijk om een efficiënte aanpak van het verkeersbeheer te ontwikkelen.

Model predictive control (MPC) als een volwassen optimalisatie-gebaseerde regelmethode is met succes toegepast in de industrie en op het gebied van verkeersmanagement. Om de regelprestaties te garanderen, vereist MPC echter een voldoende nauwkeurig model, dat vaak niet beschikbaar is voor verkeersnetwerken. Bovendien maakt de computationele complexiteit van MPC het moeilijk om het in real time uit te voeren, vooral voor grootschalige verkeersnetwerken. Daarom kunnen de huidige MPC-methoden nog worden verbeterd in termen van regelprestaties en computerefficiëntie. Reinforcement learning (RL) is onlangs op grote schaal gebruikt voor regelproblemen, waaronder verkeersmanagement, omdat het op natuurlijke wijze onzekerheden en veranderende omgevingen kan aanpakken met verwaarloosbare online computermiddelen. RL heeft echter ook zijn eigen tekortkomingen, zoals een lage steekproefefficiëntie en veiligheidsproblemen. De kenmerken van MPC en RL vullen elkaar zeer goed aan. Daarom wordt in een deel van dit proefschrift overwogen MPC en RL te combineren om de regelprestaties verder te verbeteren.

Dit proefschrift richt zich op beheer en controle van verkeersnetwerken, waaronder stedelijke netwerken en snelwegennetwerken. Meer specifiek streven we naar vermindering van verkeersopstoppingen door het minimaliseren van de totale tijd die alle voertuigen in het netwerk doorbrengen, en overwegen we ook groene mobiliteit door het minimaliseren van de totale emissies die door de voertuigen worden geproduceerd. In dit proefschrift gebruiken we MPC-gebaseerde en leergebaseerde benaderingen om dit doel te bereiken, dat verder kan worden onderverdeeld in vier delen, zoals hieronder gepresenteerd.

In het eerste deel beschouwen wij MPC voor een groen stedelijk mobiliteitsprobleem op lange termijn met cumulatieve emissiebeperkingen, waarbij de optimalisatiehorizon van het MPC-probleem aanzienlijk groter is dan de regeltijd. Daardoor wordt het aantal variabelen dat per regeltijdstap moet worden geoptimaliseerd zeer groot. Dit leidt tot optimalisatieproblemen die in real time computationeel onuitvoerbaar zijn. Om dit probleem aan te pakken, stellen wij een nieuwe bi-level temporally distributed MPC-structuur voor, die een kortetermijn- en een langetermijn-MPC-formulering omvat met kleine en grote regeltijdstappen, die gezamenlijk worden opgelost in plaats van het oorspronkelijke complexe optimalisatieprobleem. De resulterende bi-level re-

gelaanpak wordt gebruikt om het MPC-probleem online op te lossen voor real-time regeling van stedelijke verkeersnetwerken met als doel groene mobiliteit op lange termijn. Uit simulatie op een stedelijk verkeersnetwerk blijkt dat de voorgestelde bi-level MPC-aanpak beter presteert dan andere conventionele regelmethoden, in termen van totale tijdsbesteding, totale uitstoot van CO_2 en rekentijd.

In het tweede deel stellen wij, om de rekenkundige complexiteit van MPC te verminderen, voor om geparametriseerde MPC (PMPC) te gebruiken voor de verkeerssignaalregeling van stedelijke netwerken. PMPC is een efficiënte MPC-aanpak die het aantal optimalisatievariabelen vermindert door een geparametriseerde regelwet te gebruiken in plaats van de regelingangen rechtstreeks te optimaliseren. Maar het ontwerp van de geparametriseerde regelwet vereist in het algemeen deskundige kennis. In dit deel stellen wij voor grammaticale evolutie te gebruiken om continue geparametriseerde regelwetten automatisch te construeren, met behulp van een effectief simulatiegebaseerd trainingskader. Bovendien wordt een projectiemethode voorgesteld om de niet-lineaire beperkingen die worden opgelegd aan de parameters van de geparametriseerde regelwetten te verwijderen en de haalbaarheid van de oplossing van het MPC-optimalisatieprobleem te garanderen. Het trainingskader wordt toegepast op een stedelijk verkeersnetwerk en een geparametriseerde regelwet wordt gegenereerd. Simulatieresultaten op basis van SUMO tonen aan dat de resulterende PMPC-regelaar de online rekentijd aanzienlijk vermindert en een prestatie bereikt die vergelijkbaar is met die van de conventionele MPC-regelaar.

In het derde deel beschouwen we de regeling van snelwegennetwerken door MPC en deep reinforcement learning (DRL) te combineren, om met onzekerheden en storingen om te gaan. Wij stellen een nieuw kader voor de integratie van MPC en DRL-methoden voor verkeersregeling op snelwegen voor, dat verschilt van de bestaande MPC -RL-methoden. Het voorgestelde kader heeft een hiërarchische structuur, waarbij een hoogwaardige efficiënte MPC-component met een lage frequentie en een grote bemonsteringstijd werkt om een basisregelinput te leveren, terwijl de DRL-component met een hoge frequentie werkt om de door MPC gegenereerde regelinput online te wijzigen. Het regelkader heeft daarom slechts beperkte online computermiddelen nodig en kan omgaan met onzekerheden en externe storingen na een goed leerproces met voldoende trainingsgegevens. Het voorgestelde kader wordt toegepast op een benchmarksnelwegnetwerk voor de coördinatie van toeritdosering en variabele snelheidsbeperkingen. Simulatieresultaten tonen aan dat het voorgestelde raamwerk beter presteert dan de standaard MPC- en DRL-benaderingen, in termen van totale tijdsbesteding en beperkingstevredenheid, ondanks modelonzekerheden en externe verstoringen.

In het vierde deel wordt de PMPC-regeling gecombineerd met RL om de twee belangrijkste problemen van conventionele MPC tegelijkertijd aan te pakken. Wij stellen een nieuw kader voor dat RL gebruikt om de PMPC-regeling online aan te passen, waarbij alle mogelijke strategieën om verschillende onderdelen van PMPC aan te passen (bv. de doelfunctie, de state-feedback-regelwet, de optimalisatie-instellingen en het systeemmodel) worden geïntegreerd. Dit resulteert in een syntheseraamwerk voor RL-gebaseerde adaptieve PMPC. Zelfs de bestaande adaptieve (P)MPC-benaderingen kunnen in dit syntheseraamwerk worden opgenomen. Het resulterende gecombineerde PMPC-DRL kader biedt een efficiënte MPC benadering die kan omgaan met model mis-

matches. Het voorgestelde kader wordt toegepast voor verkeersregeling op snelwegen, en uit simulaties blijkt dat de RL-gebaseerde adaptieve PMPC beter presteert dan een standalone RL-gebaseerde regelaar en conventionele MPC, in termen van totale tijdsbesteding en rekentijd.

Kortom, dit proefschrift overwint de tekortkomingen van conventionele MPC door verschillende raamwerken voor te stellen, waaronder bi-level temporally-distributed MPC, grammatical evolution-based PMPC, combined DRL-MPC, en RL-based adaptive PMPC frameworks. Deze raamwerken worden toegepast voor verkeersbeheer van stedelijke netwerken en snelwegen en leveren betere prestaties dan conventionele methoden, bij verstoringen en onzekerheden in de omgeving.

PREFACE

Time flies. The memory of the morning four years ago when I first arrived in The Netherlands is still fresh like yesterday. When I look back, I find this four-year journey was not easy, full of joys and sorrows. So many things happened. From the beginning when I was curious about everything and tried to explore the research directions that were new to me, to the middle when we encountered the pandemic and everything was more difficult, including life and research. Now, we have got through it and everything gets back to normal, and I have also finished my PhD successfully, with the help of many people whom I owe thanks to.

First, I want to express my sincere gratitude to my supervisors, Prof. Bart De Schutter and Dr. Anahita Jamshidnejad. Without your support and guidance, I would never make it. Bart is definitely the perfect supervisor that everyone would love to work with. He is always patient and encouraging when I was anxious and stressed with my research, and provides his timely and wise suggestions when I was lost and slow with the progress. He is like the warm light that guide me to the right direction when I was in darkness. On the other hand, Bart has a very strict standard with research quality and academic ethics. Under his influence, I have gradually grown as a mature and qualified researcher. I also want to thank Anahita for her very careful and valuable instructions. I will never forget the draft that is full of her careful hand-written comments, which greatly improved the quality of our papers. I will also miss the days when we had joint meetings to share insightful thoughts and to have brainstorms to come up with interesting ideas. The only downside is that I have developed a fastidious taste for papers due to your high standards, which makes me very critical or even nitpicky when reviewing manuscripts, leaving no typos or inconsistencies unscathed. So, beware, fellow authors! These words are far from enough to express my gratefulness. The best thing I can do to return is passing down their valuable guidance to my future students with my own actions during the rest of my career.

Besides, I want to thank our secretaries, Marieke, Hellen, Francy, and Marsha, for their efforts to create such a comfortable environment in our department, and their efforts to organize so many interesting events. I have got precious and unforgettable memories for these moments. Here I want to thank Francy specially for her kind assistance, especially during the last a few months of my stay in TU Delft.

Next, warm and big thanks to my lovely friends. Thank Jianing for teaching me to play guitar and basketball and having fun together during the tough times. Thanks for the good times spending together with Jianing, Xiaohui, DingDing, Zhaorui, Jiaxiang, Jianzhang, Ziyu, Keer. Thank you for always being there making me never feel lonely, and you are always willing to offer me a hand when I am in trouble. Your companionship has made my life more colorful. It is really nice to know you and I really like you guys. I also want to express my thanks to Xiaoyu, Jingwei, Yun, Kanghui, Luyao, Shengling, Zhixin, Changrui, for the daily lunch time together, the inspiring discussions, and the happy weekend parties, especially during the last year of my PhD. I will miss the numerous nights when we left the department together before the building was closed. Thanks to Xiaoyu for accommodating the parties for cooking and karaoke. I can always relieve my stress in these happy events. Thanks to Yun who can always provide very helpful suggestions with his profound knowledge whenever I raise a question. Thanks to Jingwei for cheering up each other when we were in a hard time. I also want to thank the ones who have left: Maolong, Jianfeng, Tian, Ximan, Yichao. The time I have spent with you will be always cherished. All of you are very excellent researchers and I can always learn a lot from you. Thanks to the basketball team: Jianing, Jingwei, Suad, Jiaxiang, and Changrui. It is really joyful to play with you, and I have also improved my skills a lot by learning from you. Thank you!

Thanks to my colleagues. Suad, you are the FMVP, and it is really interesting to play basketball with you. Thanks to Fillipo for the interesting and insightful discussions, and I will never forget the happy time I have together with Fillipo, Xiaoyu, Yun, Shengling, Francesco, and Azita in Japan during IFAC 2023. In addition, I want to thank Dr. Azita Darib for giving me the chance to co-supervise a MSc student with her.

Then I want to thank Prof. Yueying Wang, who guided me to the academic career and helped me without reservation. Thank Prof. Li Chen for supervising my master thesis and lead me to the door of academia. Thank Dr. Weixiang Zhou and Dr. Yan Wei for sharing your experiences with me and providing many valuable suggestions. Thanks to my friends in China: Xin Wang, Ye Li, Zhao Zhang, Kailun Huang, Grace, Linhong, and many other names in my heart. I also want to mention my girlfriend, Xiaoxin. Thank you for your remote accompany for almost six years. It has been difficult for both of us to keep together with such a distance in spatial and temporal. I believe we will reach the happy end very soon.

Last but not least, I want to thank my parents and sister for your selfless love to me and your unconditional support during the last four years of my PhD study. Thank you for your understanding and for your always standing behind me with unwavering support.

Dingshan Sun
Delft, September 2023

1

INTRODUCTION

In this chapter, we first provide a brief overview of traffic management by introducing a background for the control of urban and freeway traffic networks. We then discuss the open challenges in this area, where this information provides context and motivation for this PhD thesis. In particular, we highlight the main contributions of the thesis. Finally, we present an outline of the thesis to guide the reader through the upcoming chapters.

1.1. BACKGROUND

With the rising number of vehicles on the roads, traffic congestion has become increasingly prevalent, resulting in detrimental effects on individuals and society as a whole. Traffic congestion on the roads not only hampers the smooth flow of vehicles but also has adverse implications for air quality and environmental noise, which pose a threat to ecosystems and biodiversity. According to the European Court of Auditors (ECA), road transport is already one of the primary sources of air pollution in the European Union (EU) and the European Economic Area (EEA), particularly with regards to CO_2 emissions [44] where the ECA has reported that air pollution is the most significant environmental health risk in Europe [38].

A reduction in travel time by 10% is projected to yield an increase in economic productivity of 2.9%. However, in areas heavily affected by congestion, mitigating traffic congestion can lead to even more substantial economic productivity gains, reaching up to 30%[44]. The ECA estimates the costs associated with road congestion in EU to be approximately €110 billion annually, which is over 1% of the EU's gross domestic product [44]. Therefore, it is of significant benefit and urgency to reduce the congestion by improving the traffic efficiency. This poses an unprecedented challenge to the current traffic infrastructures and traffic management systems. In fact, improving the existing road infrastructures can take effect immediately, for example via increasing the road capacities by widening the existing roads or by constructing new roads. However, a larger traffic capacity has been reported to attract more traffic demands, which is known as induced demand, and may actually result in heavier congestion [35]. Additionally, constructing new roads is expensive and time-consuming. In some cases, it is even impossible due to space restrictions or financial limitations. Another cheaper way to improve the traffic efficiency is to install traffic signs to control and manage the traffic flows. For different types of traffic networks (e.g., urban traffic network and freeway traffic network), various traffic control measures can be implemented. Next, we briefly discuss both urban and freeway traffic control methods.

1.1.1. URBAN TRAFFIC CONTROL

Since the development and installation of the first electric traffic light in the United States in 1912, ensuring safe and organized passage for both vehicles and pedestrians in urban areas, especially at crossroads, has become a necessity. Traffic signals typically use cycles, with each cycle divided into several phases, including the green light phase and the red light phase. Traditional traffic lights usually have fixed phases that cannot adapt to varying traffic situations, nor can they coordinate with each other. This can lead to traffic congestion when traffic demands are high.

Over the past four decades, many traffic signal control methods have been proposed to improve the traffic efficiency. These methods include OPAC [56], GreenWave [164], Maxband [116], SCATS [172], and SCOOT [84], which have improved the traffic efficiency to some extent, by adapting the phases of the traffic lights to different traffic conditions or by coordinating various traffic signals. A comprehensive overview of traffic signal control approaches is presented in [41], where the control approaches are categorized according to the network types, types of road users (e.g., cars and pedestrians), real-time strategies, control objectives, and constraints.

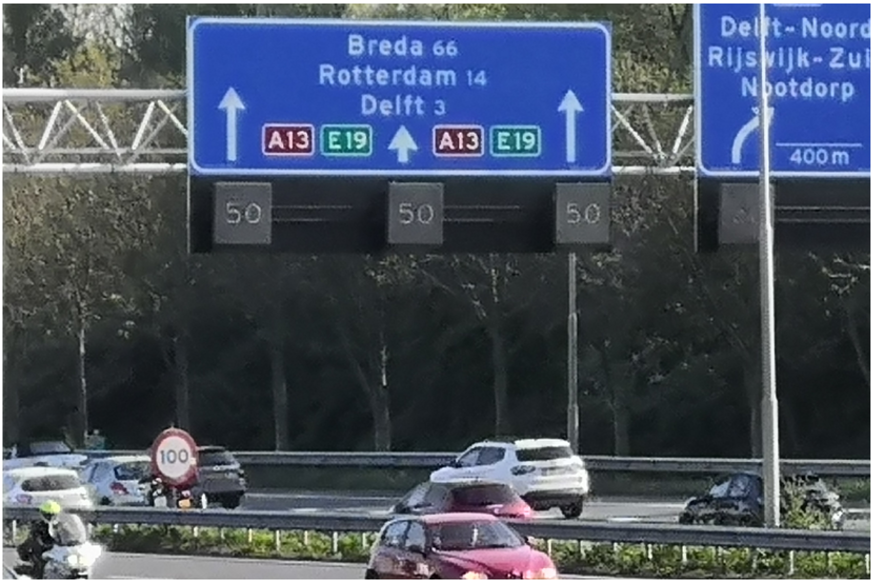

Figure 1.1: Variable speed limits signs along the A13 freeway, Delft, The Netherlands.

1.1.2. FREEWAY TRAFFIC CONTROL

Freeways, also known as highways or motorways, are a fundamental part of modern traffic networks that contribute substantially to social and economic developments of societies. Freeways can reduce the travel time, and thus enable people to travel efficiently for business, trade, or pleasure, and provide an economical trade route for goods. Accordingly, freeway congestion can have serious consequences both environmentally and economically.

To reduce congestion and guarantee traffic efficiency in freeways, several freeway control measures have been developed. The most common strategies are variable speed limits (VSLs) and ramp metering (RM). Unlike conventional fixed speed limits, VSLs are more flexible and allow the maximum speed limit to change according to the traffic conditions, which is usually displayed on an electronic traffic sign (see Figure 1.1). Thus, VSLs are more effective than fixed speed limits in reducing congestion, increasing safety, and providing clear guidance for motorists, especially when accidents or poor weather conditions occur. RM is basically a traffic signal controller that regulates the traffic flows that enter a freeway. RM is usually implemented in order to control the inflows to the ramps, and to avoid the congestion that is caused by the lane drop (see Figure 1.2). Route guidance is another system that assists drivers in choosing their routes by displaying traffic information on variable message signs, when there are multiple choices to reach their destination. In general, VSLs and RM are often integrated in order to achieve a better control performance [73]. An overview of freeway control methods is presented in [173].

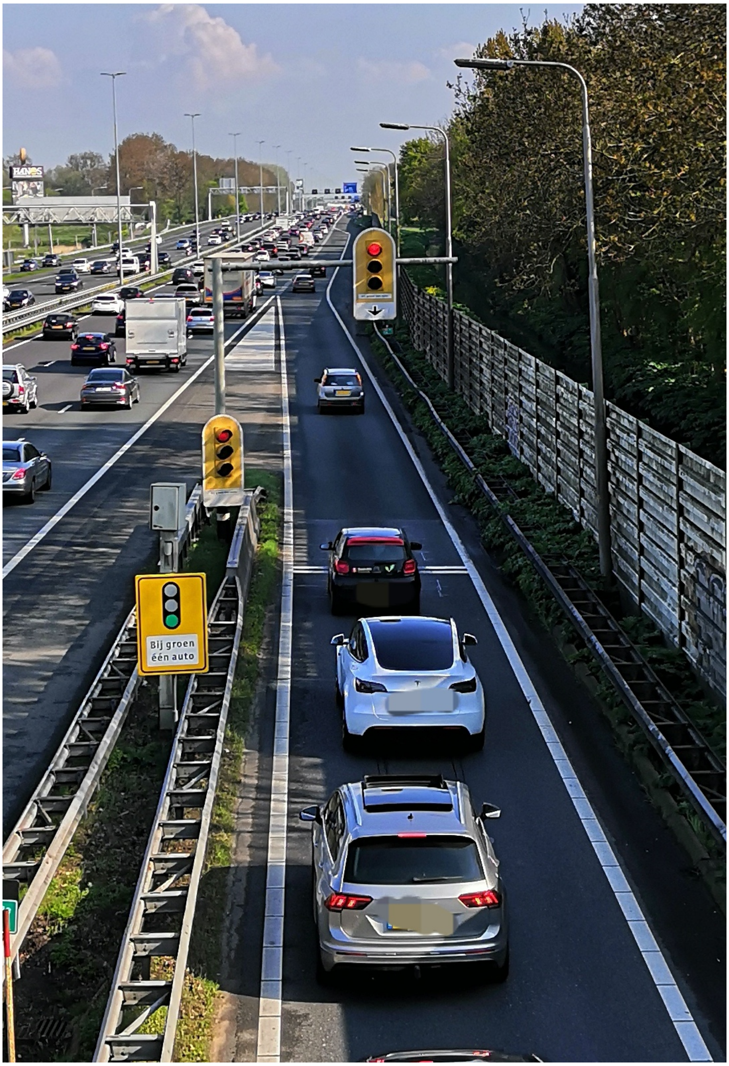

Figure 1.2: Ramp metering at the A13 freeway, Delft, The Netherlands.

1.2. CHALLENGES

Despite the emergence of effective traffic control measures, road congestion continues to persist due to ongoing urbanization and continually increasing traffic demands. Even a minor traffic accident can cause local congestion that may spread throughout the entire traffic network, resulting in a global traffic jam. This vulnerability is particularly pronounced during the rush hours when the traffic demand can easily surpass the road capacity. As a result, it is crucial to consider the coordination of various local traffic controllers in order to achieve an optimal performance for a traffic network. Additionally, predicting future traffic behavior based on estimated traffic demands and road conditions can enhance decision-making and improve both traffic efficiency and resilience.

Model predictive control (MPC) [158] is a model-based and optimization-based control method that meets the above requirements very well (i.e., coordinating local controllers and predicting future behaviors). At every control step, MPC solves an optimization problem to minimize a given objective function by predicting the future system behaviors using a prediction model. The optimized control inputs are then implemented on the real system, and the process is repeated at the next control step. MPC has been widely used in practice [156], since it can handle systems with multiple inputs and outputs and can explicitly incorporate both state and input constraints into its decision-making. MPC can also coordinate multiple local controllers, either in a centralized or distributed way [125], in order to optimize a global objective. These characteristics of MPC meet the requirements of traffic management very well, and hence has led to many successful applications of MPC in the field of traffic network control, for both urban and freeway networks [73], [115]. A survey on the use of MPC for traffic signal control is given in [205], and many notable studies of MPC for freeway traffic control are presented in [48], [50], [52], [53], [74]. Although MPC has achieved considerable success in traffic management, it still suffers from the following two main issues:

1. Computational complexity is a major challenge for MPC in traffic management. The complexity of the optimization problem that should be solved at each control step mainly depends mainly on the nominal model and on the size of the prediction horizon. In traffic networks, the mathematical model that represents the dynamics is typically nonlinear in order to accurately capture various traffic phenomena. As a consequence, the resulting optimization problem is also nonlinear and often nonconvex. Additionally, traffic networks are usually large-scale and have many states and inputs, which further complicates the MPC optimization problem. Finally, due to the inherent time-delayed nature of traffic networks, the prediction horizon needs to be large enough to capture the future effects of the current control inputs for the entire traffic network. These factors make MPC computationally too intensive to be implemented in real time, especially for large-scale traffic networks.

2. The performance of MPC is highly dependent on the accuracy of the prediction model. However, obtaining an accurate model for complex real-world systems, particularly for traffic networks, is often challenging. Additionally, traffic models used in MPC should typically have a limited level of details, in order to prevent that the computations become intractable. Thus, the mismatches between the traffic network and the prediction model are inevitable, and may worsen the performance of MPC. Moreover, external disturbances and unpredictable traffic conditions can exacerbate the issue.

Therefore, this thesis will address the above two main issues of MPC for traffic management, i.e., computational complexity and model mismatches. Chapter 2 and Chapter 3 focus on the implementation of MPC for urban traffic networks, and propose solutions to tackle the computational complexity of MPC. Chapter 4 and Chapter 5 focus on the MPC problem for freeway networks, where both computational complexity and model mismatches are addressed. It is worth noting that the approaches that are developed in these chapters can be applied to both urban and freeway traffic networks.

1.3. CONTRIBUTIONS

The main contributions of this thesis are presented below:

- We propose a novel bi-level temporally-distributed MPC approach to address green urban mobility with a long-term cumulative emission constraint. When considering constraints for a long term (e.g., the maximum allowed emissions for one year), a very large prediction horizon is required; however, the MPC problem may become computationally intractable with a very large prediction horizon. The proposed methods circumvent this problem by adopting a two-level MPC framework: A long-term MPC controller with a less detailed prediction model works at the high level to govern the long-term constraints and to assign maximum allowed emissions to the low level of control, in which a short-term MPC controller with a detailed prediction model operates in real time to provide network-wide traffic signal control inputs.

- We propose a parameterized MPC (PMPC) method for urban signal control based on grammatical evolution. PMPC can reduce the computational complexity of conventional MPC, since the number of the optimization variables is usually reduced by using a parameterized control law. We propose two training frameworks to generate the parameterized control law by employing grammatical evolution, where these methods are illustrated via a case study that involves traffic signal control for an urban network.

- We propose a novel framework that combines MPC and deep reinforcement learning (DRL) with application to freeway networks. The combined MPC-DRL framework is able to address the two main issues of conventional MPC (i.e. computational complexity and model mismatches) simultaneously. The high-level MPC controller works at a low frequency, and, therefore, is real-time implementable; the low-level DRL controller works at a high frequency, aiming to compensate for model mismatches and external disturbances. In the case study, VSLs and RM are integrated within this framework for control of a freeway network.

- We propose a reinforcement learning (RL) based adaptive PMPC scheme, in which all components of PMPC can be parameterized and adapted by the high-level RL agent. These components include the objective function, the prediction model, the control law, etc. The proposed synthesis framework can adapt to model mismatches and to changing environments, while the embedded PMPC controller is computationally more efficient than a conventional MPC controller. The proposed framework is illustrated via a case study, in which the RL-based adaptive PMPC controller is implemented for RM control of a freeway network.

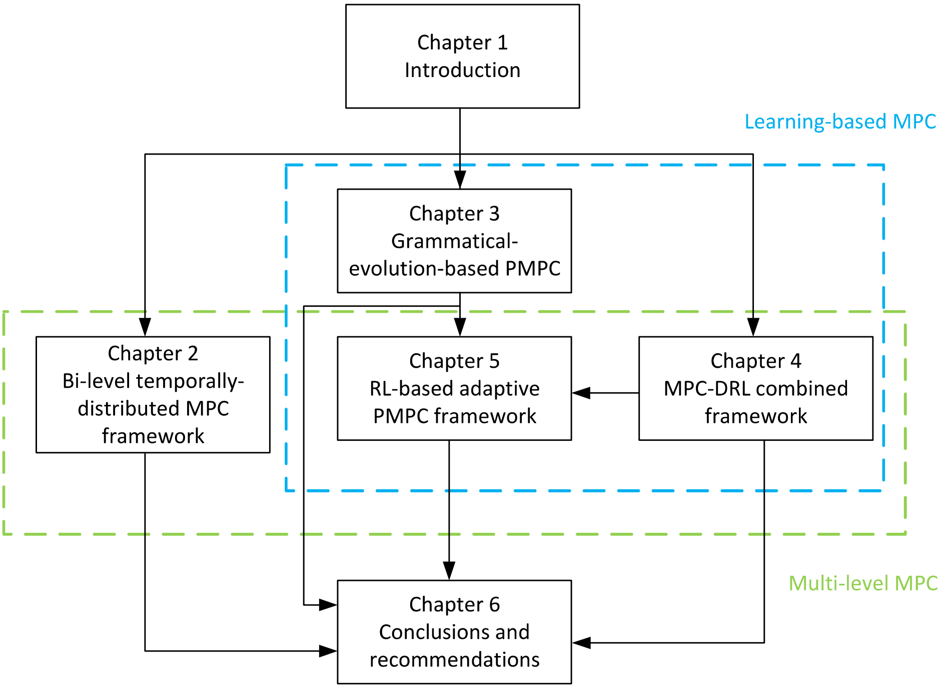
Chapter 1
Introduction
Learning-based MPC
Chapter 3
Grammatical-evolution-based PMPC
Chapter 2
Bi-level temporally-distributed MPC framework
Chapter 5
RL-based adaptive PMPC framework
Chapter 4
MPC-DRL combined framework
Multi-level MPC
Chapter 6
Conclusions and recommendations

Figure 1.3: Outline and structure of this thesis.

1.4. OUTLINE

The structure of this thesis is presented in Figure 1.3. Chapter 2 to Chapter 5 are a collection of papers that have either been published, or submitted to a journal, or are under preparation for a journal. Note that since each chapter develops different frameworks, the mathematical notations are defined for each chapter separately.

Chapter 2 develops a bi-level temporally-distributed MPC framework to deal with long-term cumulative emission constraints of urban traffic networks. Chapter 3 uses grammatical evolution to learn PMPC control laws for urban traffic networks. Chapter 4 develops a novel multi-level MPC-DRL combined framework for freeway traffic control. On the basis of Chapter 3 and Chapter 4, Chapter 5 establishes an RL-based adaptive PMPC framework that integrates existing adaptive MPC techniques and uses RL to adjust all the components of the PMPC scheme. Chapter 6 concludes the work of this thesis, and proposes recommendations for future research based on the work of each chapter.

2

A NOVEL BI-LEVEL TEMPORALLY-DISTRIBUTED MPC APPROACH: AN APPLICATION TO GREEN URBAN MOBILITY

Model predictive control (MPC) has been widely used for traffic management, such as for minimizing the total time spent or the total emissions of vehicles. When long-term green urban mobility is considered, the optimization horizon of the MPC problem is significantly larger than the control sampling time, and thus the number of the variables that should be optimized per control time step becomes very large. For systems with dynamics that involve nonlinear, non-convex, and non-smooth functions, including urban traffic networks, this results in optimization problems that are computationally intractable in real time. In this chapter, we propose a novel bi-level temporal distribution of such complex MPC optimization problems, and we develop two mathematically linked short-term and long-term MPC formulations with small and large control sampling times that will be solved together instead of the original complex optimization problem. The resulting bi-level control architecture is used to solve the two MPC formulations online for real-time control of urban traffic networks with the objective of long-term green mobility. In order to assess the performance of the bi-level control architecture, we perform a case study where a rough version of the model of the urban traffic flow, S-model, is used by the long-term MPC level to estimate the states of the urban traffic networks, and a detailed version of the model is used by the short-term MPC level. The results of the simulations prove the effectiveness (with respect to the objective of control, as well as computational efficiency) of the proposed bi-level MPC approach, compared to state-of-the-art control approaches.

Parts of this chapter have been published in [85], to which this PhD candidate contributes significantly.

2.1. INTRODUCTION AND MOTIVATIONS

One of the main long-term objectives of the European Climate Law [43] is to achieve climate neutrality by 2050, which means zero greenhouse gas emissions for all EU countries. The law, correspondingly, sets an intermediate target: to reduce the net amount of greenhouse gas emissions for, at least, 55% by 2030, compared to the levels in 1990. According to the European emissions gap report [187], transportation accounts for one quarter of all the energy-related greenhouse gas emissions, and it is foreseen that by 2050 two-third of the world population will be urban. This can double the motorized mobility and lead to a 60% increase in CO_2 emissions [39], [144]. Although there have been attempts to reduce the emissions by promoting the use of electric vehicles, public transit, and active transportation (e.g., walking and cycling), the traditional vehicles that exhaust emissions still make up a dominant part of the transportation. The European Environment Agency reported in 2017 that the amount of nitrogen dioxide produced annually across Europe had significantly violated its allowed values [45]. Nitrogen dioxide is a main component of air pollution that is very harmful to the environment and to human health. This pollution is mostly associated with vehicle emissions, and according to [45] 86% of the nitrogen dioxide exceedances have been detected at roadside monitoring locations.

Therefore, there is an urgent need for high-performing control systems that provide green mobility by reducing the traffic emissions, especially in urban networks, which are also the focus of this chapter. In order to coordinate with the climate policies, while finding a balanced trade-off between minimizing the traffic congestion and the level of harmful pollutants from the vehicle exhausts, such control systems should maintain the long-term emission levels to ensure that they do not exceed the annual emission limit.

Model predictive control (MPC) is an interesting approach for traffic control (see [11], [17], [126], [173], [200]). MPC has recently been proposed to provide green urban mobility [30], [86], [87]. The MPC optimization problems for green urban mobility are multi-objective and subject to several (nonlinear) control and state constraints. Thuse, these problems are mathematically and computationally complex, due to the large simulation horizon (i.e., weeks or months) and small control sampling time (i.e., seconds or minutes), accompanied by highly nonlinear and fluctuating dynamics of urban traffic. Next, we briefly introduce MPC and its open challenges for green urban mobility.

2.1.1. MODEL PREDICTIVE CONTROL (MPC)

Model predictive control or MPC is a feedback-based optimal control approach [13], [124]. An MPC-based controller (see Fig. 2.1 given for a discrete-time system with control sampling time c) consists of two main elements, a prediction model and an optimizer, which at every control time step run across a prediction horizon of size n^{p}. The prediction model mathematically formulates the evolution of the dynamics of the controlled system, and cooperates with the optimizer to determine a sequence of control inputs that satisfy the constraints and minimize the given cost function. The feedback-based nature of MPC, i.e., using the measured states per control time step, makes the controlled system to some extent robust to unexpected/unpredictable external disturbances [136]. Moreover, MPC has proven to be an efficient approach for problems that should handle both input and state constraints, while optimizing multiple cost functions [23], [158].

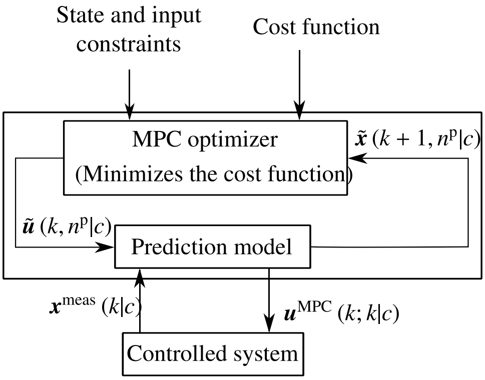
State and input constraints
Cost function
MPC optimizer
(Minimizes the cost function)
$\tilde{\boldsymbol{x}}\left(k+1, n^{\mathrm{p}}|c\right)$
$\tilde{\boldsymbol{u}}\left(k, n^{\mathrm{p}}|c\right)$
Prediction model
$\boldsymbol{x}^{\mathrm{meas}}\left(k|c\right)$
$\boldsymbol{u}^{\mathrm{MPC}}\left(k;k|c\right)$
Controlled system

Figure 2.1: Main structure of an MPC-based controller.

MPC has been widely used for urban traffic signal control, and a comprehensive survey can be found in [205]. [183] is one of the early studies that utilize MPC in urban traffic management. [66] developed a macroscopic traffic modeling approach for mixed networks of freeways and arterials and solved the corresponding the optimal traffic control problem using MPC. [197] proposed a hierarchical MPC structure considering the different dynamics in different levels of traffic networks, in which the higher layer provides reference outflow trajectories to the lower layer. [182] used robust MPC to develop a traffic-responsive optimal signal split algorithm taking uncertainty into account. [143] focused on the scalability of MPC for traffic signal control for large-scale traffic networks, and proposed a multi-agent MPC algorithm with graceful extension and localized reconfiguration, in which theoretical results have been investigated for the formulated linear traffic dynamic systems in terms of convergence and global optimum. However, very few studies consider the green urban mobility issue, which can introduce a long-term cumulative constraint that is difficult to be addressed by conventional MPC methods.

2.1.2. CURRENT CHALLENGES OF MPC FOR GREEN URBAN MOBILITY

The main challenges of implementing MPC for green urban mobility are explained below:

- The computational complexity of MPC can make MPC intractable in real time [162], particularly for green urban mobility where highly nonlinear dynamics, large spatial and temporal scales, and long-term control objectives and cumulative constraints are involved.
- Despite providing a longer-term vision of the future (which is in benefit of the long-term control objectives and cumulative constraints), using a large prediction horizon significantly increases the computational complexity. This issue may be tackled in two ways:

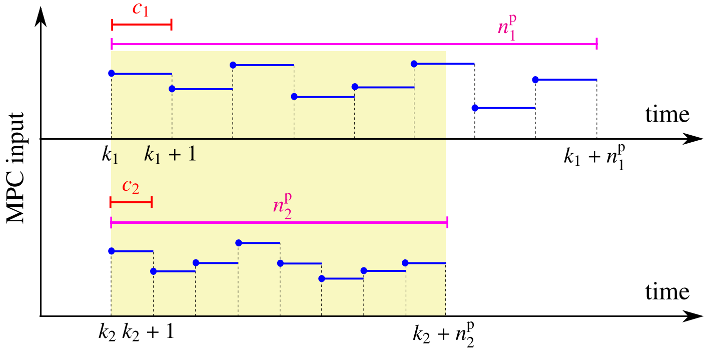

Figure 2.2: **Top plot:** A decreased time scale resolution (i.e., a larger control sampling time c_1) for a larger prediction horizon n_1^{p} may result in less dynamics and adaptability in the MPC input trajectory for a given prediction horizon n_2^{p}. **Bottom plot:** An increased time scale resolution (i.e., a smaller control sampling time c_2), which can be computationally tractable for a smaller prediction horizon n_2^{p}, can result in more dynamics and adaptability in the MPC input trajectory in the given prediction window.

    - Decreasing the time scaling resolution, i.e., using a larger control sampling time: This, however, may result in less dynamics and adaptability for the MPC input (see Fig. 2.2).
    - Simplifying the prediction model: This, however, may reduce the accuracy of the predicted states, and result in larger cumulative errors, particularly along a large prediction horizon.
- In general, the optimization horizon of MPC should be related to the time needed to travel through the traffic network [1], [93]. However, such a choice of prediction horizon cannot explicitly address long-term control objectives and cumulative constraints. This becomes particularly problematic for traffic systems that have a large time delay for the control inputs to take effects on the system.

The control frequency also matters in MPC for traffic control. A higher control frequency will improve the control performance, at the price of more intensive computational complexity, while a lower control frequency requires less computational efforts, resulting however in general in a less optimal control performance. On the other hand, a larger control sampling time results in a larger prediction window (as indicated in Fig. 2.2), but may also lead to loss of control performance. In this chapter, we reach a trade-off between accuracy and computational complexity by adopting a multi-frequency control framework, in which both a low-frequency MPC with large control sampling time and a high-frequency MPC with small control sampling time are integrated.

2.1.3. CONTRIBUTIONS & STRUCTURE OF THIS CHAPTER

Therefore, in this chapter, we will address the challenges of MPC for green urban mobility, to resolve the conflict between the long-term control objective and constraints on the one hand and the short-term optimization horizon on the other hand. In particular, we will propose a bi-level control architecture that embeds MPC controllers with different frequencies of operation and prediction horizons in the two control levels. The proposed

Table 2.1: Frequently-used mathematical notations in this chapter (using a discrete-time framework).

s	Simulation sampling time of the traffic model (e.g., 1 min). Without specification, s denotes the detailed model simulation sampling time, while denotes the rough model simulation sampling time
k	Without specification, k denotes the short-term MPC control time step, while k^{LT} denotes the long-term MPC control time step
c	Control sampling time: the number of time units during which a control input remains unchanged. Without specification, c denotes the short-term MPC control sampling time (e.g., 5 min), while c^{LT} denotes the long-term MPC control sampling time (e.g., 60 min)
c^{o}	Operation sampling time: every a certain number of time units the control sequence is optimized and updated. Without specification, c^{o} denotes the short-term MPC operation sampling time (e.g., 5 min), while denotes the long-term MPC operation sampling time (e.g., 60 min)
$n^{\mathrm{p}}(k)$	Prediction horizon at control time step k; in particular, $n^{\mathrm{ST}}(k)$ denotes the short-term MPC prediction horizon, while $n^{\mathrm{LT}}(k^{\mathrm{LT}})$ denotes the (shrinking) long-term MPC prediction horizon
$n^{\mathrm{s}}(k)$	Simulation horizon at control time step k, which initially equals N^{s} and shrinks gradually, where N^{s} is the considered total simulation interval length (e.g., for a simulation interval of 1 day with the simulation sampling time of 1 min, $N^{\mathrm{s}} = 1440$)
$\boldsymbol{u}(k+\ell\|c)$	Control input at control time step $k+\ell$ (where $\ell = 0, 1, \dots, n^{\mathrm{p}}(k) - 1$) that is computed at control time step k, with control sampling time c (e.g., the green time length for urban traffic control). This notation applies for both short-term and long-term MPC
$\boldsymbol{x}(k+\ell\|c)$	State variable at control time step $k+\ell$ (where $\ell = 1, 2, \dots, n^{\mathrm{p}}(k)$) that is estimated by the prediction model at control time step k, with control sampling time c (e.g., the number of vehicles, queue lengths, vehicle speeds on each lanes, etc.). In addition, $\check{\tilde{\boldsymbol{x}}}$ represents the states estimated by the rough model
$\boldsymbol{x}^{\mathrm{meas}}(k\|c)$	State variable measured at control time step k, with control sampling time c; note that $\boldsymbol{x}(k\|c) = \boldsymbol{x}^{\mathrm{meas}}(k\|c)$
$\tilde{\boldsymbol{u}}(k, n\|c)$	Sequence of the control inputs determined at control time step k for all control time steps across the horizon n, with control sampling time c, i.e., $\tilde{\boldsymbol{u}}(k, n\|c) = [\boldsymbol{u}(k\|c), \dots, \boldsymbol{u}(k+n-1\|c)]^{\top}$;
$\tilde{\boldsymbol{x}}(k, n\|c)$	Sequence of the state variables estimated by the prediction model at control time step k for all control time steps across the horizon n, with control sampling time c, i.e., $\tilde{\boldsymbol{x}}(k, n\|c) = [\boldsymbol{x}(k+1\|c), \boldsymbol{x}(k+2\|c), \dots, \boldsymbol{x}(k+n\|c)]^{\top}$; in addition, $\check{\tilde{\boldsymbol{x}}}$ represents the corresponding variables for the rough model
$f^{\mathrm{state}}(\cdot)$	Detailed integrated flow-emission traffic model (e.g., an integrated macroscopic traffic model and emission model [115]); while $\check{f}^{\mathrm{state}}(\cdot)$ denotes the extracted rough integrated model

Note: For $\boldsymbol{u}$, $\boldsymbol{x}$, $\boldsymbol{x}^{\mathrm{meas}}$, $\tilde{\boldsymbol{u}}$, and $\tilde{\boldsymbol{x}}$ to be complete in definition, in addition to the control sampling time, the initial control time step should generally also be given as an argument. However, we assume that the initial control time steps for all time frames, independent of the size of the control sampling time, are synchronized and coincide with a fixed, known initial time step.

control system will be implemented to an urban traffic network to achieve green mobility.

The main contributions of this chapter include: 1) For the first time, a multi-level MPC-based architecture with a larger prediction horizon at the high level and a smaller prediction horizon at the low level will be implemented for obtaining green urban mobility. 2) The link and inter-dynamics of the two control levels are defined differently from any existing work: The emissions allowed in the long term are determined via the high-level MPC controller, and are adjusted for the shorter terms via the low-level MPC controller. This idea can be generalized to other fields, e.g., for energy allocation in building energy management. 3) This is the first time that MPC is adopted for long-term control of the cumulative emissions for green urban mobility. In fact, the proposed framework can limit the emissions for a long enough time span (e.g., a year), with an affordable online computation time compared to existing control methods.

Next we present a background discussion on the related theory (i.e., multi-level MPC). The rest of this chapter has the following structure: Section 2.2 describes and formulates the MPC problem of green urban mobility. In Section 2.3, our proposed novel approaches for tackling complex MPC optimization problems, including that of the green urban mobility, are explained. Section 2.4 presents a case study, where our proposed MPC approach is compared with various state-of-the-art control methods for a simulated urban traffic network, and discusses the corresponding results. Section 2.5 concludes the chapter and gives topics for future research. Table 2.1 lists and defines the frequently-used mathematical notations in this chapter.

2.1.4. RELATED WORK

Hierarchical (multi-level) MPC schemes are often used to address complex control problem. This topic has been studied extensively, and a comprehensive review can be found in [169], where hierarchical MPC is classified into four categories.

1. Hierarchical MPC for coordinated control: In such architectures, a higher-level controller coordinates the control inputs generated by the lower-level local controllers, where the controllers of both levels can be MPC-based.

2. Hierarchical MPC for dealing with systems with multiple time scales: In general, the higher-level controller operates according to slow dynamics and a lower frequency, whereas the lower-level controller operates with faster dynamics and a higher frequency. Both control levels can be used for the same system that is then described via different time scales. The high-level controller optimizes the control variables that have a long-term effect on the system, and these values are then used as references for the low-level controller to track (see, e.g., [18], [188]). Moreover, the two control levels can be used for different sub-systems with different functionalities and control frequencies (see, e.g., [37], [67]).

3. Hierarchical MPC for control of systems with a hierarchical structure: This category corresponds to a classical cascade feedback control system. For examples of controllers that belong to this category, see [37] and [33].

4. Hierarchical control for plantwide optimization: The high level of control can use the detailed dynamics of the system to compute optimal operating conditions, whereas the low level of control employs simpler dynamics to follow the references generated by the high-level controller. This control architecture is usually used in the process industry. Such a control system can also be implemented in a dual way, i.e., the high level of control uses simplified or abstracted dynamics of the system to predict the long-term performance, and considers the objective function across a large prediction horizon. Meanwhile, the low level of control works with a more accurate model and calculates the current control inputs according to a shorter prediction horizon [149].

The following paper will be illustrated in more details, since it is more relevant to our work. Jin *et al.* consider the hierarchical MPC approach of category 2 to schedule the energy resources of smart buildings with a microgrid [91]. The high level of control follows a day-ahead dynamic optimal scheduling, where the schedules of the smart buildings, distributed generators, batteries, and day-ahead setpoints of electric tie-line power are optimized for an entire day. The corresponding prediction horizon covers the duration from the current time to the end of the day, and the optimization is performed hourly. The low level of control follows an intra-hour rolling adjustment, where the low-level MPC works with a detailed model and a faster control frequency, and performs with a smaller prediction horizon. The low-level MPC tries to follow the reference (i.e., day-ahead schedules) generated by the high-level optimization process. A similar strategy is used by Liu *et al.* for energy management of microgrids [117].

Most of the literature that use hierarchical MPC consider the application in energy management for buildings, process industry, or wastewater treatment. A few researchers have also implemented hierarchical MPC for traffic management. The early work [189] proposed a four-layer control architecture for freeway traffic, where the tasks of these four layers are route choice, path planning, maneuver, and regulation. Two more recent papers ([10], [165]) employed hierarchical MPC of category 1 for coordinated control of freeway traffic networks. Su *et al.* consider a multi-level control strategy for the maintenance of railway networks, in which a chance-constrained MPC is used at the high level to perform a long-term optimization for the overall maintenance plan, and to provide maintenance suggestions for the low-level controllers [175], [176]. Han *et al.* use a hierarchical control structure for the ramp metering control of a freeway network [68]. A high-level MPC-based controller determines the optimal total inflow from the on-ramps to the freeway stretch by using an aggregated model. Then the total inflow is distributed among the on-ramps via a low-level MPC-based controller. Nonetheless, no study has considered any temporally-distributed multi-level MPC for traffic management yet. In this chapter, we proposed a bi-level MPC control framework with a hierarchical structure of category 4.

2.2. GREEN URBAN MOBILITY BASED ON AN ANNUAL MPC SCHEDULE

In this section, the concept of cumulative constraints for MPC is first introduced. Then we formulate the MPC problem of green urban mobility, discuss the main characteristics of the resulting optimization problem, and explain our novel approach for tackling the complexities of this problem.

2.2.1. CUMULATIVE CONSTRAINTS

In general, ordinary MPC only considers instantaneous constraints on the states and inputs (see the second and the third top plots in Fig. 2.3), which indicate, respectively, that the realized value of an equality constraint should be equal to the given value, and that the realized value of an inequality constraint should not violate the upper bound. However, cumulative constraints (i.e., constraints defined on the summation of the realized values of a variable for multiple control time steps) should be considered for green urban mobility, since there are annual emission limits required by climate policies (i.e., the cumulative emissions over the entire year should not exceed an annual limit).

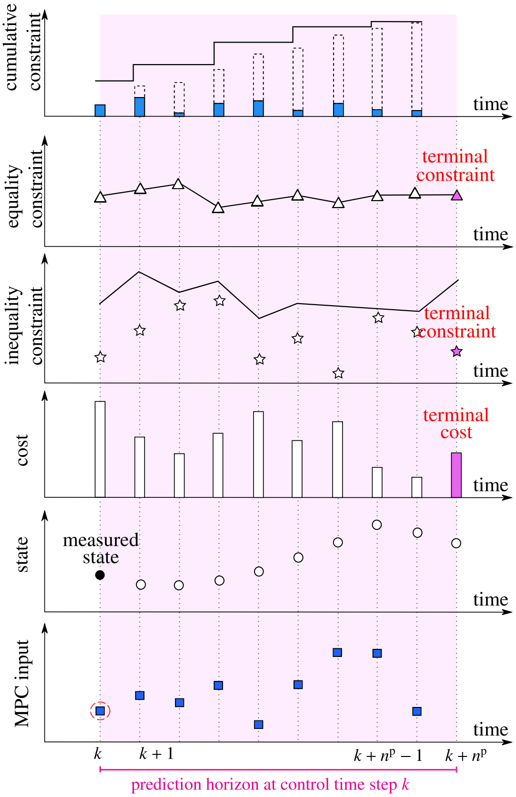

Figure 2.3: Illustration of the variables, cost functions, and constraints for an MPC optimization problem: In the second to fourth plots from the top, the white triangles, stars, and bars represent the stage constraints/costs and the colored triangles, stars, and bars show their terminal values. In the top plot, the dashed bars represent the realized cumulative (from control time step k until the current control time step) constraint for every control time step, whereas the colored (blue) part of the bars represent the realized value of the constraint for that particular control time step.

For every control time step $k+1,\dots,k+n^\mathrm{p}$ across the prediction horizon of MPC, the accumulated value of a specific variable (i.e., the length of the corresponding dashed bar in the top plot of Fig. 2.3) should not exceed a given upper bound (shown by the black continuous curve in the top plot of Fig. 2.3). Note that the length of each colored (blue) bar in Fig. 2.3 corresponds to the realized value of the variable at the current control time step. Moreover, the length of every dashed bar represents the accumulated value of this variable (i.e., the summation of the lengths of the current and all the previous colored blue bars).

A main feature of cumulative constraints is that the maximum value of the corresponding variable for a given control time step (i.e., the maximum allowed length of the corresponding colored blue bar) depends on the value of the cumulative constraint already realized, while for instantaneous constraints the upper bound is independent of

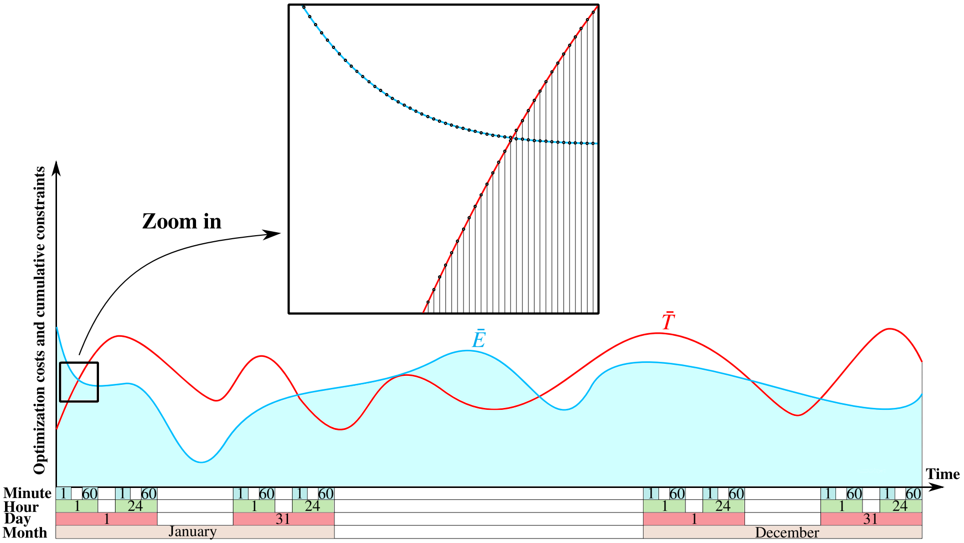

Figure 2.4: The significant difference between the temporal scales of the control costs (i.e., a year), cumulative constraints (i.e., up to a year), and the control input (i.e., a minute), including a zoomed-in sketch of the finest time resolution that corresponds to the control sampling time.

the previous values. This characteristic of cumulative constraints can provide flexibility in the predictive decision-making of MPC, i.e., by selecting an alternative optimal solution that further constrains the cumulative value at one control time step, MPC can loosen the upper bound constraint for the upcoming control time steps, and vice versa.

2.2.2. PROBLEM FORMULATION

The problem involves real-time scheduling and planning of traffic signals at the intersections of an urban traffic network, such that the congestion and total emissions of particular pollutants across a predefined simulation horizon are reduced. For the simulation horizon, we consider a fixed yearly time frame, where the control procedure begins at 0:00 of the first day of January and ends at 23:50 of the last day of December of the same year (considering a control sampling time of 10 min) . The size of the simulation horizon for the entire 1 year (i.e., 365 days × 24 h × 60 min divided by the control sampling time 10 min) is given by N^{s}. The main constraints are on the total emissions of particular pollutants at given monitoring time steps (e.g., at the end of the year).

At control time step k (when the measured state $x^{\text{meas}}(k|c)$ is received)[1], the corresponding green mobility control problem can be formulated across the simulation hori-

---

[1] We suppose that the simulation time steps coincide with the control time steps.

zon $n^{\mathrm{s}}(k)$ (which initially equals N^{s} and shrinks gradually) by:

$$\begin{aligned}\min_{\tilde{\boldsymbol{u}}(k,n^{\mathrm{s}}(k)|c)} & \Big(\bar{T}_k\Big(\tilde{\boldsymbol{x}}\left(k,n^{\mathrm{s}}\left(k\right)|c\right)\Big)+\sum_{\epsilon\in\mathbb{E}}\lambda_\epsilon\bar{E}_k\Big(\boldsymbol{P}_\epsilon,\tilde{\boldsymbol{x}}\left(k,n^{\mathrm{s}}\left(k\right)|c\right)\Big)+\\ & \lambda^{\mathrm{var}}\bar{V}\Big(\tilde{\boldsymbol{u}}\left(k,n^{\mathrm{s}}\left(k\right)|c\right)\Big)\end{aligned}\tag{2.1}$$

s.t. :

C1: state prediction model:

$$\tilde{\boldsymbol{x}}(k,n^{\mathrm{s}}(k)|c)=f^{\mathrm{state}}\big(\boldsymbol{x}^{\mathrm{meas}}\left(k|c\right),\tilde{\boldsymbol{u}}\left(k,n^{\mathrm{s}}\left(k\right)|c\right)\big),$$

C2: instantaneous stage and terminal constraints:

$$\tilde{\boldsymbol{x}}(k,n^{\mathrm{s}}(k)|c)\in\mathbb{X}^{n^{\mathrm{s}}(k)},\quad \tilde{\boldsymbol{u}}(k,n^{\mathrm{s}}(k)|c)\in\mathbb{U}^{n^{\mathrm{s}}(k)},$$

C3: cumulative constraint $\forall\epsilon\in\mathbb{E}$:

$$\bar{E}_k\big(\boldsymbol{P}_\epsilon,\tilde{\boldsymbol{x}}(k,n^{\mathrm{s}}(k)|c)\big)\le\bar{E}_\epsilon^{\mathrm{safe}}-\bar{E}_\epsilon^{\mathrm{real}}(k|c).$$

Every control time step, the simulation horizon is reduced by 1 unit compared to the previous control time step. Thus, $n^{\mathrm{s}}(k)=N^{\mathrm{s}}-k$. In (2.1), $\bar{T}_k(\cdot)$ and $\bar{E}_k(\cdot)$ are functions that give the cumulative travel time (a quantitative measure of the traffic congestion) and the cumulative emissions of a particular pollutant for all the vehicles within the time interval corresponding to the horizon $n^{\mathrm{s}}(k)$, starting at control time step k. Moreover, $\boldsymbol{P}_\epsilon$ is a matrix that includes parameter values that are identified experimentally for every pollutant ϵ (e.g., see [208]) with $\mathbb{E}$ the set of all pollutants, and λ_ϵ is a weight that indicates the relative importance of various pollutants. The function $\bar{V}(\cdot)$ computes the norm of the variation in between two consecutive control input vectors and λ^{var} is the corresponding weight. In **C1**, $f^{\mathrm{state}}(\cdot)$ is a generally nonlinear function that models the evolution of the dynamics of the traffic network. In **C2**, $\mathbb{X}$ and $\mathbb{U}$ are the admissible sets for the state variables and the control inputs, with the superscript $n^{\mathrm{s}}(k)$ denoting the dimension. In **C3**, $\bar{E}_\epsilon^{\mathrm{safe}}$ shows the maximum allowed value of the cumulative emissions for pollutant ϵ, which is illustrated by the continuous black curve in Fig. 2.3. Note that in the green urban mobility application this upper bound is fixed, i.e., it is equal to the allowed annual emissions of a pollutant. Finally, $\bar{E}_\epsilon^{\mathrm{real}}(\cdot|c)$ is the value of the total emissions of ϵ already realized by a given control time step (this value for every control time step is the length of the dashed bar at the previous time step in Fig. 2.3).

2.2.3. CHARACTERISTICS OF THE OPTIMIZATION PROBLEM

The constrained optimization problem (2.1) has the following characteristics:

- The problem involves minimization of a cost function subject to various control and state constraints, looking into the future across a finite simulation horizon with a fixed final control time step. This implies that (2.1) has the structure of a shrinking-horizon optimization problem [174].

- Due to the nonlinearities in the traffic behavior, $\bar{T}_k(\cdot)$, $\bar{E}_k(\cdot)$, and $f^{\mathrm{state}}(\cdot)$ are in general nonlinear, non-smooth, and possibly non-convex. Therefore, (2.1) is generally nonlinear and non-convex.

- The green urban mobility optimization problem, including the cost function and the cumulative constraints, is defined over a relatively long time span (e.g., 12 months), while the control inputs (i.e., the green times of the traffic signals) of the controlled system (with dynamics that may be prone to rapid nonlinear changes), need to be determined at relatively high frequencies (e.g., every few seconds or minutes). These result in small control sampling times and a large value N^{s} for the simulation horizon, which implies a large number of optimization variables that should be determined online and in real time via (2.1).

- For (2.1) to be computationally tractable, the details may be reduced via, e.g., simplifying the prediction models and increasing the control and operation sampling times, which respectively decrease the computational burden and the number of the optimization variables. Taking these measures may result in negative impacts on the accuracy and performance of the control system, as it was discussed in Section 2.1.2.

- The above-mentioned characteristics of (2.1), including nonlinearity, non-convexity, and long-term control objectives and constraints, next to the need for frequent online and real-time decision making, involving a large number of optimization variables, yield a complex optimization problem.

Fig. 2.4 illustrates the entire simulation horizon, over which the elements of the cost function, i.e., $\bar{T}_k(\cdot)$ and $\bar{E}_k(\cdot)$, and the cumulative constraints are defined. A cut of the plot (within the rectangular frame) has been zoomed in, which shows the significant difference between the temporal scales of the control input and the control costs and cumulative constraints. The characteristics of (2.1) mentioned earlier imply that this optimization problem may not be easy/tractable to tackle online and in real time by conventional methods. Next, we discuss how our proposed novel approaches can make (2.1) computationally tractable.

2.3. PROPOSED METHODOLOGY FOR TACKLING THE OPTIMIZATION COMPLEXITY RESULTING FROM VARIOUS TEMPORAL SCALES

In this section, we give our proposed approaches for tackling the complexities of (2.1), due to different temporal scales (i.e., small control sampling time and large simulation horizon). Our proposed methods consist of a temporal distribution and reformulation of the problem, using a shrinking-horizon approach, called *jumping-horizon*, and a bi-level multiple-frequency control architecture for implementation and solving the new formulations of the optimization problem (2.1).

2.3.1. BI-LEVEL TEMPORAL DISTRIBUTION OF THE PROBLEM

In the optimization problem (2.1), two very different temporal scales appear, due to $N^{\mathrm{s}} \gg c$. For a controlled system with a long-term cost function, an efficient control system should guarantee that the short-term control inputs will gradually lead the controlled system towards its desired long-term cost, while the short-term behavior of the controlled system also fulfills the requirements of the users of the controlled system, taking into account the rapid fluctuations of the system dynamics. Such a control system needs an overall vision of the controlled system through the entire control period, as well as more detailed information and vision about its short-term dynamics. Therefore, we propose to develop two linked MPC optimization formulations for the original optimization problem (2.1), where the long-term and short-term costs and constraints of (2.1) are distributed among these two formulations. The resulting MPC problems can be solved individually online, and their integrated solutions can result in a controlled behaviour for the system that is sufficiently close to the behaviour of a centralized controller that solves (2.1), while being significantly more computationally efficient. Next, we explain the two MPC formulations in detail.

ROUGH LONG-TERM MPC FORMULATION

A rough long-term MPC optimization problem is formulated within the same shrinking simulation window as (2.1), but with a (significantly) larger control sampling time c^{LT} (in this case, one-third of a month), resulting in different control time steps k^{LT}. Fig. 2.5 illustrates an example of the rough long-term MPC input at long-term control time step k^{LT}, assuming that k^{LT} coincides with April 1 at 0:00. Moreover, simplified versions of $\bar{T}_k(\cdot)$ and $\bar{E}_k(\cdot)$ (shown by $\check{\bar{T}}_k(\cdot)$ and $\check{\bar{E}}_k(\cdot)$), and a less detailed prediction model $\check{f}^{\mathrm{state}}(\cdot)$ for the state variables are considered. The prediction horizon of the rough long-term MPC at long-term control time step k^{LT} is given by $n^{\mathrm{LT}}(k^{\mathrm{LT}})$. The initial size of the long-term prediction horizon is $N^{\mathrm{s}}c/c^{\mathrm{LT}}$, and thus for the long-term prediction horizon we have $n^{\mathrm{LT}}(k^{\mathrm{LT}}) = N^{\mathrm{s}}c/c^{\mathrm{LT}} - k^{\mathrm{LT}}$. The rough long-term MPC optimization problem at long-term control time step k^{LT} is given by:

$$\min_{\tilde{\boldsymbol{u}}(k^{\mathrm{LT}},n^{\mathrm{LT}}(k^{\mathrm{LT}})|c^{\mathrm{LT}})} \Bigg(\check{\bar{T}}_{k^{\mathrm{LT}}}\Big(\check{\tilde{\boldsymbol{x}}}\big(k^{\mathrm{LT}},n^{\mathrm{LT}}(k^{\mathrm{LT}})|c^{\mathrm{LT}}\big)\Big) + \sum_{\epsilon\in\mathbb{E}} \lambda_{\epsilon}\check{\bar{E}}_{k^{\mathrm{LT}}}\Big(\boldsymbol{P}_{\epsilon},\check{\tilde{\boldsymbol{x}}}\big(k^{\mathrm{LT}},n^{\mathrm{LT}}(k^{\mathrm{LT}})|c^{\mathrm{LT}}\big)\Big)+$$
$$\lambda^{\mathrm{var}}\bar{V}\Big(\tilde{\boldsymbol{u}}\big(k^{\mathrm{LT}},n^{\mathrm{LT}}(k^{\mathrm{LT}})|c^{\mathrm{LT}}\big)\Big)\Bigg) \tag{2.2}$$

s.t. :

state prediction model:

$$\check{\tilde{\boldsymbol{x}}}\big(k^{\mathrm{LT}},n^{\mathrm{LT}}(k^{\mathrm{LT}})|c^{\mathrm{LT}}\big) = \check{f}^{\mathrm{state}}\big(\boldsymbol{x}^{\mathrm{meas}}(k^{\mathrm{LT}}|c^{\mathrm{LT}}),\tilde{\boldsymbol{u}}\big(k^{\mathrm{LT}},n^{\mathrm{LT}}(k^{\mathrm{LT}})|c^{\mathrm{LT}}\big)\big),$$

instantaneous stage and terminal constraints:

$$\check{\tilde{\boldsymbol{x}}}\big(k^{\mathrm{LT}},n^{\mathrm{LT}}(k^{\mathrm{LT}})|c^{\mathrm{LT}}\big)\in\mathbb{X}^{n^{\mathrm{LT}}(k^{\mathrm{LT}})},\ \tilde{\boldsymbol{u}}\big(k^{\mathrm{LT}},n^{\mathrm{LT}}(k^{\mathrm{LT}})|c^{\mathrm{LT}}\big)\in\mathbb{U}^{n^{\mathrm{LT}}(k^{\mathrm{LT}})},$$

cumulative constraint $\forall\epsilon\in\mathbb{E}$:

$$\check{\bar{E}}_{k^{\mathrm{LT}}}\Big(\boldsymbol{P}_{\epsilon},\check{\tilde{\boldsymbol{x}}}\big(k^{\mathrm{LT}},n^{\mathrm{LT}}(k^{\mathrm{LT}})|c^{\mathrm{LT}}\big)\Big)\leq\bar{E}^{\mathrm{safe}}_{\epsilon}-\bar{E}^{\mathrm{real}}_{\epsilon}(k^{\mathrm{LT}}|c^{\mathrm{LT}}).$$

Note that $\check{\tilde{\boldsymbol{x}}}$ is used to show that the corresponding states are determined by the prediction model $\check{f}^{\mathrm{state}}(\cdot)$, instead of by $f^{\mathrm{state}}(\cdot)$. To formulate the rough long-term MPC optimization problem (2.2), a good choice of c^{LT} (with $c^{\mathrm{LT}} > c$) that results in a balanced trade-off between the time and accuracy of computations is important. This variable has been represented by a different color (red) in (2.2) to specify that it is a design variable in the proposed temporally-distributed approach. The solutions of the rough long-term MPC optimization problem, which are determined based on a farther vision of the future and less details in the dynamics of the controlled system, may affect the solutions of the short-term MPC optimization problem (explained next in Section 2.3.1), while they do not directly steer the controlled system.

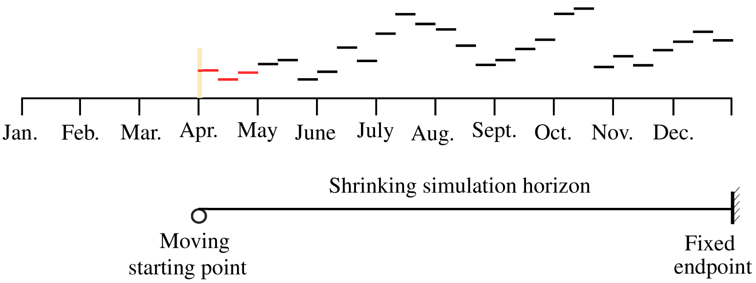

Figure 2.5: MPC input across the shrinking simulation horizon corresponding to the rough long-term MPC optimization formulation (c^{LT} =1/3 month).

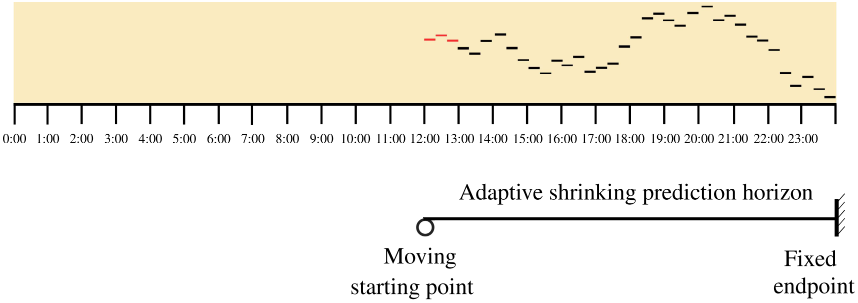

Figure 2.6: MPC input across the adaptive prediction horizon corresponding to the detailed short-term MPC optimization formulation for $c = 20$ min.

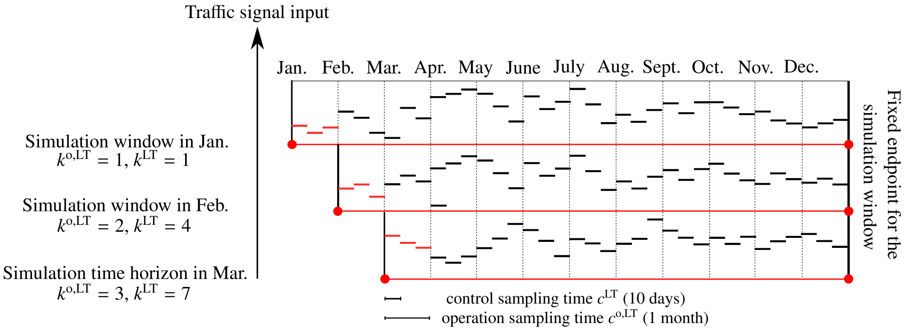
Traffic signal input
Jan.
Feb.
Mar.
Apr.
May
June
July
Aug.
Sept.
Oct.
Nov.
Dec.
Simulation window in Jan.
$k^{\mathrm{o,LT}} = 1$, $k^{\mathrm{LT}} = 1$
Simulation window in Feb.
$k^{\mathrm{o,LT}} = 2$, $k^{\mathrm{LT}} = 4$
Simulation time horizon in Mar.
$k^{\mathrm{o,LT}} = 3$, $k^{\mathrm{LT}} = 7$
Fixed endpoint for the simulation window
control sampling time c^{LT} (10 days)
operation sampling time $c^{\mathrm{o,LT}}$ (1 month)

Figure 2.7: Jumping-horizon MPC for green urban mobility for an entire operation period of 12 months, where the frequency of operation is 3 (i.e., the controller updates the control input sequence after every 3 control time steps), the operation sampling time is 1 month, with the operation time steps coinciding with Jan., …, Dec., and the control sampling time is one-third of a month.

DETAILED SHORT-TERM MPC FORMULATION

A second MPC optimization problem is formulated across an *adaptive* prediction horizon $n^{\mathrm{ST}}(k)$ starting at the current control time step k, with $n^{\mathrm{ST}}(k) \leq n^{\mathrm{s}}(k)$ and control sampling time c. Note that since the control sampling time of the short-term MPC formulation and (2.1) are the same, and also based on Remark 1, the short-term control time step is simply k. Additionally, detailed prediction models (e.g., the same as for (2.1)) are considered. The detailed short-term MPC optimization problem at control time step k is given by:

$$\begin{aligned}\min_{\tilde{\boldsymbol{u}}(k,n^{\mathrm{ST}}(k)|c)} \Big(& \bar{T}_k\Big(\tilde{\boldsymbol{x}}\big(k,n^{\mathrm{ST}}(k)|c\big)\Big) + \sum_{\epsilon\in\mathbb{E}} \lambda_\epsilon \bar{E}_k\Big(\boldsymbol{P}_\epsilon, \tilde{\boldsymbol{x}}\big(k,n^{\mathrm{ST}}(k)|c\big)\Big) + \\ & \lambda^{\mathrm{var}} \bar{V}\Big(\tilde{\boldsymbol{u}}\big(k,n^{\mathrm{ST}}(k)|c\big)\Big) + \lambda^{\mathrm{term}}_{(1)} \boldsymbol{x}_{(1)}\big(k+n^{\mathrm{ST}}(k)|c\big) - \\ & \lambda^{\mathrm{term}}_{(2)} \boldsymbol{x}_{(2)}\big(k+n^{\mathrm{ST}}(k)|c\big)\Big) \end{aligned} \tag{2.3}$$

$$\begin{aligned} \text{s.t.}: \; & \\ & \text{state prediction model:} \\ & \tilde{\boldsymbol{x}}\big(k,n^{\mathrm{ST}}(k)|c\big) = f^{\mathrm{state}}\Big(\boldsymbol{x}^{\mathrm{meas}}(k|c), \tilde{\boldsymbol{u}}\big(k,n^{\mathrm{ST}}(k)|c\big)\Big), \\ & \text{instantaneous stage and terminal constraints:} \\ & \tilde{\boldsymbol{x}}\big(k,n^{\mathrm{ST}}(k)|c\big) \in \mathbb{X}^{n^{\mathrm{ST}}(k)}, \quad \tilde{\boldsymbol{u}}\big(k,n^{\mathrm{ST}}(k)|c\big) \in \mathbb{U}^{n^{\mathrm{ST}}(k)}, \\ & \text{cumulative constraint } \forall \epsilon \in \mathbb{E}: \\ & \bar{E}_k\Big(\boldsymbol{P}_\epsilon, \tilde{\boldsymbol{x}}\big(k,n^{\mathrm{ST}}(k)|c\big)\Big) \leq \bar{E}^{\mathrm{safe,\,ST}}_\epsilon(k) - \bar{E}^{\mathrm{real}}_\epsilon(k|c). \end{aligned}$$

At every control time step k, the short-term prediction horizon $n^{\mathrm{ST}}(k)$ is determined and applied to the detailed short-term MPC optimization problem in a shrinking manner until the next time step $k+1$. In (2.3), $\boldsymbol{x}_{(1)}$ and $\boldsymbol{x}_{(2)}$ are, respectively, the sub-vector of the state variables of $\boldsymbol{x}$ (e.g., the number of vehicles moving on the lanes and the number of vehicles idling in the queues) that should be minimized at the terminal control time step, and the sub-vector of $\boldsymbol{x}$ including the kinetic state variables (e.g., the speeds and accelerations of the vehicles) that should be maximized at the terminal control time step. The last two terms in the argument of the min function in (2.1) correspond to the terminal cost that is added to the short-term MPC problem, in order to compensate for the effect of reducing the size of the prediction horizon with respect to the original optimization problem. The parameters $\lambda_{(1)}^{\text{term}}$ and $\lambda_{(2)}^{\text{term}}$ are weights for the components of the terminal cost.

Fig. 2.6 shows an example for the detailed short-term MPC input, where the adaptive shrinking prediction horizon starts at 0:00, has an initial size of 72, and gradually shrinks (e.g., the prediction horizon illustrated in Fig. 2.6 has already shrunk for 36 control time steps). Note that in this example, the detailed short-term MPC optimization problem temporally covers a part of the rough long-term MPC optimization problem that is shown within a highlighted yellow rectangle in Fig. 2.5 (i.e., one-tenth of the long-term control sampling time).

In formulation (2.3) for the detailed short-term MPC optimization problem, the choice of $n^{\text{ST}}(k)$ and $\bar{E}_{\epsilon}^{\text{safe, ST}}(k)$ plays an important role in the effectiveness of the determined control inputs. Therefore, we have shown these variables in In formulation (2.3) for the detailed short-term MPC optimization problem, the choice of $n^{\text{ST}}(k)$ and $\bar{E}_{\epsilon}^{\text{safe, ST}}(k)$ plays an important role in the effectiveness of the determined control inputs. Therefore, we have shown these variables in color (red) to specify that these are design parameters. In general, the value of $n^{\text{ST}}(k)$ can be selected according to the size of the traffic network, such that the horizon aligns with the time needed to travel through the traffic network [1], [93], [182]. In the proposed multi-frequency bi-level MPC framework, the high-level MPC controller can address the long-term plan with a large control sampling time and a low control frequency. Therefore, the low-level MPC can employ a normal size of prediction horizon as suggested by the references given above. The main aim of the proposed approach is to select $\bar{E}_{\epsilon}^{\text{safe, ST}}(k)$ in (2.3) based on the solution of (2.2), such that the resulting optimal MPC solution of (2.3) provides a high level of accuracy due to, both, the small control sampling time of (2.3) and the long-term temporal vision of (2.2), while a proper choice of $\bar{E}_{\epsilon}^{\text{safe, ST}}(\cdot)$, may result in more flexibility (i.e., less tight constraints) for the cumulative constraints in the remainder of the simulation time. Such novel integration of (2.2) and (2.3) will provide a balanced trade-off between the speed and accuracy of the optimization computations.

Remark 1. *We assume that the initial control time steps for* (2.1)*, long-term, and short-term MPC optimizations overlap, and that the control sampling times of the corresponding controllers are such that the terminal control time steps for all these frameworks fall on the terminal time instant of the simulation window.*

JUMPING-HORIZON MPC

We introduce the concept of *jumping-horizon MPC*, where the operation frequency of the MPC-based controller can be different from the control frequency. Operation frequency of MPC indicates how often the controller solves the optimization problem and updates the control input sequence, while control frequency implies how often the control input changes. Therefore, jumping-horizon MPC is a combination of shrinking-horizon MPC and multi-rate MPC.

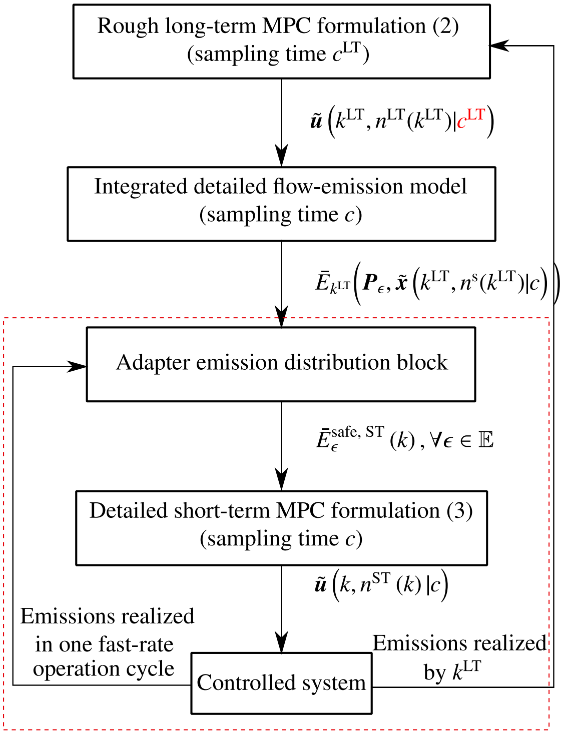
Rough long-term MPC formulation (2)
(sampling time c^{LT})
$\tilde{\boldsymbol{u}}\left(k^{\mathrm{LT}}, n^{\mathrm{LT}}(k^{\mathrm{LT}})|c^{\mathrm{LT}}\right)$
Integrated detailed flow-emission model
(sampling time c)
$\bar{E}_{k^{\mathrm{LT}}}\left(\boldsymbol{P}_{\epsilon}, \tilde{\boldsymbol{x}}\left(k^{\mathrm{LT}}, n^{\mathrm{s}}(k^{\mathrm{LT}})|c\right)\right)$
Adapter emission distribution block
$\bar{E}_{\epsilon}^{\text{safe, ST}}(k), \forall \epsilon \in \mathbb{E}$
Detailed short-term MPC formulation (3)
(sampling time c)
$\tilde{\boldsymbol{u}}\left(k, n^{\mathrm{ST}}(k)|c\right)$
Emissions realized
in one fast-rate
operation cycle
Controlled system
Emissions realized
by k^{LT}

Figure 2.8: Linking the long-term and short-term MPC formulations at long-term control time step k^{LT} that corresponds to short-term control time step k, using an adapter block for distribution of the estimated cumulative emissions among the short-term prediction window and the remainder of the simulation window.

In jumping-horizon MPC, the relationship between the operation sampling time c^{o} and the control sampling time c is given by:

$$c^{\mathrm{o}} = v \cdot c, \quad \text{with} \quad 1 \leq v \leq n^{\mathrm{p}}, \tag{2.4}$$

where $v = 1$ corresponds to regular MPC explained in Section 2.1.1. For every control time step that coincides with an operation time step and the next $v-1$ control time steps, the first v elements of $\tilde{\boldsymbol{u}}(k, n^{\mathrm{p}}|c)$ are implemented to the controlled system.

Fig. 2.7 illustrates jumping-horizon MPC for the green mobility control problem (see Section 2.2) applied to the rough long-term MPC formulation. In this figure, the control sampling time is one-third of the operation sampling time, i.e., $v = 3$.

LINKING THE LONG-TERM AND SHORT-TERM MPC FORMULATIONS

In order to link the long-term and short-term MPC formulations (2.2) and (2.3), we propose a bi-level control architecture with various frequencies of operation (see Fig. 2.8). The long-term MPC problem (2.2) is solved less frequently via a *slow-rate controller*, whereas the short-term MPC problem (2.3) is solved via a *fast-rate controller*. In order to simplify the formulations and thus let the operation time steps overlap with the control time steps, we suppose that the operation sampling time of the slow-rate MPC controller is a multiple of the long-term control sampling time, thus a multiple of the control sampling time c (see Remark 1).

The slow-rate computations are performed via the outer loop in Fig. 2.8. While the slow-rate MPC controller uses a rough model for prediction, its solution $\tilde{\boldsymbol{u}}\left(k^{\mathrm{LT}}, n^{\mathrm{LT}}(k^{\mathrm{LT}})|c^{\mathrm{LT}}\right)$ is used by a detailed integrated flow-emission model (e.g., $f^{\text{state}}(\cdot)$) with sampling time c to determine $\tilde{\boldsymbol{x}}\left(k^{\mathrm{LT}}, n^{\mathrm{s}}(k^{\mathrm{LT}})|c\right)$, and then $\bar{E}_{k^{\mathrm{LT}}}\left(\boldsymbol{P}_{\epsilon}, \tilde{\boldsymbol{x}}\left(k^{\mathrm{LT}}, n^{\mathrm{s}}(k^{\mathrm{LT}})|c\right)\right)$ for all $\epsilon \in \mathbb{E}$. Next, the values of the cumulative emissions estimated for the remainder of the simulation window are injected into an adapter block, which distributes these values between the current short-term prediction window (e.g., the section of the simulation window that is distinguished by a highlighted yellow rectangle in Fig. 2.5) and the remainder of the simulation window. The share of the cumulative emissions that is associated with the short-term prediction window by the adapter block will be used by the MPC formulation (2.3) as the upper bound value $\bar{E}_{\epsilon}^{\text{safe, ST}}(\cdot)$ for the cumulative constraints in order to determine the control input sequences $\tilde{\boldsymbol{u}}\left(k, n^{\mathrm{ST}}(k)|c\right)$. In practice, only those elements of this control input sequence that correspond to one fast-rate operation sampling time are used to steer the system (see for instance the control inputs illustrated in red in Fig. 2.6) are injected into the controlled system to control the actuators (in this case the traffic signals).

After one fast-rate operation sampling time, the values of the cumulative emissions realized within this interval are sent via the controlled system to the adapter block, which uses these values to update the upper bounds for the cumulative emissions, and redistribute these values between the current short-term window and the rest of the simulation window.

Note that the adapter block can be designed to produce in parallel various candidate distributions for the upper bound of the cumulative emissions. In that case, (2.3) will be solved for all these possible distributions in parallel, and from all the optimal solutions determined, the one that corresponds to the least realized cost and/or the least value for $\bar{E}_{\epsilon}^{\text{safe, ST}}(\cdot)$ (or to the least value for a weighted combination of these two quantities) will be selected.

2.4. CASE STUDY

In this section, we perform two case studies with different time scales in order to evaluate the performance and validate the temporal-scalability of the proposed bi-level temporally distributed MPC approach for green mobility in an urban traffic network. The cost function consists of the total time spent (TTS) and total emissions (TE) of the vehicles traveling in the urban traffic network within a given simulation window. For the emissions, we focus on CO_2, which is the main cause of greenhouse effect. For comparison, we consider state-of-the-art control methods, including fixed-time control, responsive control, optimized fixed-time control, and conventional MPC. The performance of these controllers is assessed according to the following criteria: realized values of the total time spent by the vehicles in the urban traffic network, total emissions of CO_2, the realized value of the cost function (i.e., a weighted summation of the total time spent and total

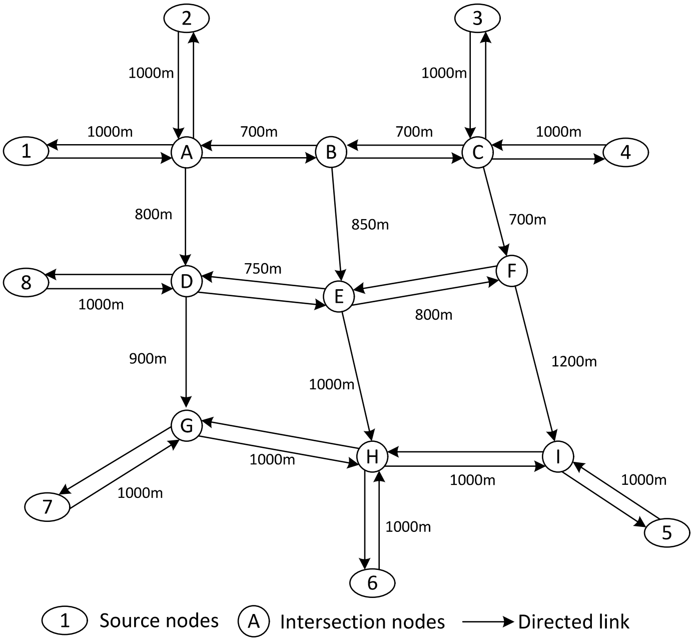

Figure 2.9: Urban traffic network used for the case study.

emissions, and for conventional MPC a penalty corresponding to constraint violation), as well as the CPU time for the computations of each controller.

2.4.1. SETUP FOR CASE STUDY 1

URBAN TRAFFIC NETWORK

In this case study, we consider an urban traffic network (shown in Fig. 2.9; a similar network has been considered by [99]) with 8 source/destination nodes labeled by numbers 1–8 where vehicles enter and leave the traffic network, and with 9 intersection nodes labeled by letters A–I. The arrows in the figure illustrate the links on which the vehicles can move from an upstream node to a downstream node. The numbers next to these arrows give the length of the corresponding link in m. Every two adjacent intersection nodes are connected by at least one link and at most two links with different directions. Each intersection node is controlled via a traffic signal, except for node B, which does not have a controller. A centralized controller is used for all the traffic signals, which have the same fixed cycle time equal to 60 s and are synchronous. Each directed link consists of 1-3 lanes, where the number of lanes corresponds to the number of the downstream links. As an example, the detailed illustration of a part of the urban traffic network that includes the links corresponding to nodes A and B is shown in Fig. 2.10. Since link (A,B) has two downstream links (B,C) and (B,E), it consists of two lanes. Vehicles that enter the traffic network via a source node are not allowed to turn immediately from the corresponding source link into a neighboring destination link and leave the traffic network

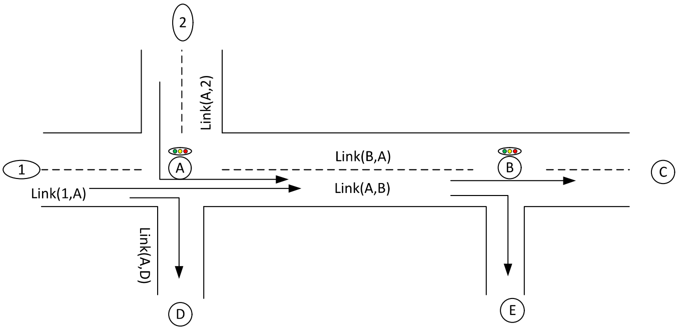

Figure 2.10: Detailed illustration of the part of the urban traffic network that includes intersection nodes A and B.

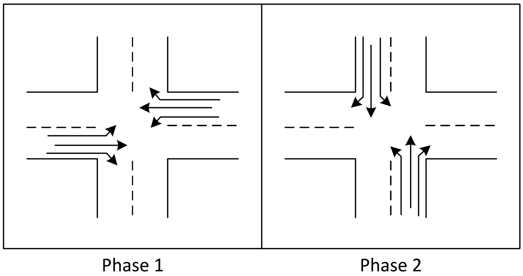

Figure 2.11: Illustration of the two phases corresponding to the cycle of every traffic signal.

(e.g., vehicles that enter via node 1 in Fig. 2.10 are not allowed to turn into link (A,2)). Finally, the cycle of every traffic signal includes two phases (see Fig. 2.11 for an intersection node with four links). Note that the same condition holds for T-shaped crosses, such as those at intersection nodes F, G, and I.

TRAFFIC FLOW AND EMISSION MODELS

In this case study, the dynamics of the urban traffic flow is modeled via the S-model [114], which is macroscopic and updates the state variables of every link of the urban traffic network per simulation time step (which is considered to be equal to the cycle time of the downstream traffic signal of the link). The state variables for every link include the total number of vehicles and the number of vehicles in the queue(s) on the link (see [114] and [88] for more details). In order to calculate the emissions of CO_2 by the vehicles in the network, we use VT-micro [208] integrated with the S-model (see [115] for details).

For the detailed short-term and rough long-term MPC controllers, two versions of the S-model are considered: the S-model with a detailed simulation sampling time (equal to the cycle time of the traffic signals, i.e., 60 s), and the S-model with a rough simulation sampling time (five times the cycle time of the traffic signals, i.e., 300 s), respectively. The rough version of the S-model approximates the state variables of the urban traffic network faster and with a reduced, but acceptable accuracy compared to the detailed version of the S-model. The parameters used for the integrated flow and emission model for the urban traffic network are presented in Table 2.2, where v^{free} is the free-flow speed,

Table 2.2: Parameters of the integrated flow and emission model used for the case study

v^{free}[m/s]	v^{idle}[m/s]	a^{acc}[m/s^2]	a^{dec}[m/s^2]	l^{veh}[m]
16.67	1.11	2	-2	5

v^{idle} is the idling speed, a^{acc} is the acceleration, a^{dec} is the deceleration, and l^{veh} is the average length of the vehicles in the traffic network.

DEMAND PROFILES

Six demand scenarios have been considered for a simulation window of 6 hours, beginning at 6:00 and ending at 12:00. This simulation window covers the morning rush hours. Although larger simulation windows can be considered to be controlled by the proposed approaches, we have considered this simulation window in order to compare the proposed control approach with more existing control methods with lower computational burden, where 6 h is large enough to represent various traffic flow and emission dynamics. The profiles for the traffic demands at the source nodes for the 6 scenarios are shown in Fig. 2.12. These scenarios are chosen since they can generate traffic congestion, such that the effectiveness of these controllers can be validated. Compared to Scenario 1, Scenario 2 has a delayed peak in the morning and no peak for the demand at noon, while for Scenario 3 the peaks correspond to larger values of demand both in the morning and at noon. Moreover, Scenarios 4 and 5 have lower peak values both in the morning and at noon, while Scenario 6 has a higher morning peak than Scenario 1, but no peak occurs for this scenario at noon. In order to make the case study more realistic, we have included some noise to the demand profiles for Scenarios 2–6, which will be used to evaluate and compare the performance of various controllers. More specifically, in each of the Scenarios 2–6, we have added the noise signals defined by $N_1(t) = 10\sin(10t)$, $N_2(t) = 40\sin(t)$, $N_3(t) = 40\cos(2t+1)$, $N_4(t) = 45\cos(t+1)$, $N_5(t) = 50\sin(0.5t)$, $N_6(t) = 50\sin(1.2t+1)$, $N_7(t) = 40\sin(1.5t+1)$, $N_8(t) = 40\cos(1.3t+1)$, to the demands at sources 1–8, respectively. The demand profiles that include the noise correspond to the predicted demands and imply that imperfect predictions of the real-life demand profiles may be available for the controllers.

CUMULATIVE EMISSION CONSTRAINTS

The maximum allowed cumulative emissions of CO_2 are set to 70000 kg for all the scenarios, except for Scenario 4, where the maximum is 65000 kg. The reason for considering a smaller cumulative emissions of CO_2 in Scenario 4 is that there the demands are significantly lower compared to the other scenarios.

2.4.2. CONTROLLERS

For all the controllers considered in this case study, the control input variable is the green time length for each traffic signal, with a lower bound of 10 s and an upper bound of 50 s. The controllers that have been considered in this case study are introduced next.

FIXED-TIME CONTROLLER

With the fixed-time controller, the green time lengths are not optimized, but are instead given as a fixed value of 30 s for all the controlled intersections within the entire simu-

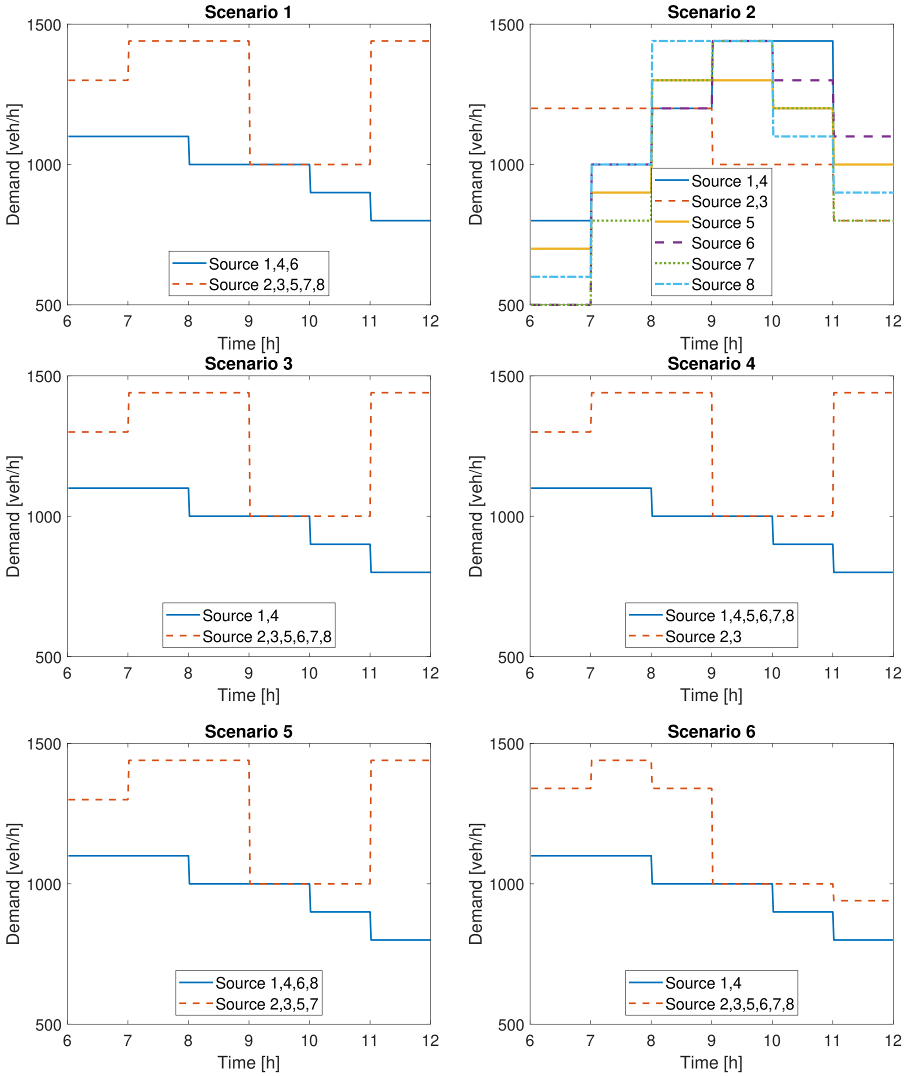

Figure 2.12: Demand profiles for the 6 scenarios in case study 1.

lation window of 6 h. This case is considered as a benchmark for all the other control approaches that are implemented in this case study.

RESPONSIVE CONTROLLER

The responsive controller is an online adaptive traffic controller that updates the green time length of a controlled intersection at every control time step according to the traffic volume of the connecting links of that intersection. The links that include more vehicles will receive a larger green time length (see [95] for details). The control sampling time of the responsive controller is 1 min.

OPTIMIZATION POLICIES FOR ADAPTIVE CONTROL (OPAC)

OPAC is a computational strategy for real-time demand-responsive traffic signal control [56]. For the next control sampling time, the controller estimates the upcoming traffic flows, and enumerates all the possible choices of green time length (which should be integer values within the allowed range of the control input variable) in order to find an optimal value for the corresponding green time length that results in the least total delay for the particular controlled intersection. Note that these estimations are performed for individual controlled intersections simultaneously (i.e., in a decentralized way). The dynamics of the links corresponding to the controlled intersection are updated via the S-model and are used to predict the future values of the state variables of these links. Note that the constraints on the emissions cannot be incorporated explicitly in an OPAC controller. The control sampling time of the OPAC controller is set to 1 min.

OPTIMIZED FIXED-TIME CONTROLLER

This controller is optimized off-line using a rough version of the S-model and demand Scenario 1 shown in Fig. 2.12. The optimization problem is solved considering a cost function that is defined as a weighted summation of the TTS and the TE within the simulation window, with a control sampling time of 1 h. A rough estimation of the total emissions within the simulation window is given as the upper bound for the cumulative constraint of the optimization problem. Whenever the optimizer fails to find a feasible solution with respect to the given constraint, the optimization problem is solved excluding the cumulative emission constraint, and a penalty corresponding to the emission constraint violation is added to the cost function.

CONVENTIONAL MPC CONTROLLER

In order to implement an MPC controller in real time for green mobility in the given urban traffic network, the prediction time interval is limited to 15 min with a control sampling time of 5 min (i.e., the prediction horizon is 3) and an operation sampling time of 5 min. The MPC optimization problem is solved considering the detailed S-model and a cost function defined as a weighted sum of the TTS and the TE within the prediction window. The upper bound for the cumulative emissions of CO_2 within the current prediction window is estimated based on the demand profiles, i.e., the ratio of the expected demand within the current prediction window and the future expected demand is used to distribute the remaining allowed cumulative emissions. By comparing the performance and CPU time of this MPC controller and the bi-level MPC controller, we can realize how and to what extent adding the long-term MPC controller impact the overall

Table 2.3: Parameters values for the case study

s	s^{LT}	$n^{\mathrm{ST}}(k^{\mathrm{LT}})$	$n^{\mathrm{LT}}(k^{\mathrm{LT}})$	c	c^{LT}	c^{o}	$c^{\mathrm{o,LT}}$	$\lambda_{\mathrm{CO_2}}$	λ^{var}	$\lambda^{\mathrm{term}}_{(1)}$	$\lambda^{\mathrm{term}}_{(2)}$
1 min	5 min	15 min	360 min	5 min	60 min	5 min	60 min	0.005	0	0	0

performance of the controlled system, as well as the computational burden of the MPC controller.

CONVENTIONAL MPC WITH A LARGE PREDICTION HORIZON

The same conventional MPC controller as the previous subsection, but with a larger prediction horizon is considered. The prediction time interval of this controller is doubled (i.e., it is 30 min), and therefore the prediction horizon is 6. The other settings are exactly the same as the previous conventional MPC controller. Comparing the performance of this MPC controller with that of the bi-level and conventional MPC controller will show us whether or not we can gain the desired performance via a single level of control, with still an affordable computation time.

SINGLE-LEVEL LONG-TERM MPC CONTROLLER

The rough long-term MPC-based controller in the high level of the proposed framework will be considered as the controller that directly controls the traffic network. The rough long-term MPC controller optimizes the multi-objective cost function (i.e., weighted summation of the TTS and the TE) considering a rough version of the S-model, with a simulation sampling time of 5 min, and a control and operation sampling time of 1 h. The controller originally has a prediction horizon of 6 (equal to the simulation horizon), which is implemented in a shrinking-horizon way.

BI-LEVEL TEMPORALLY-DISTRIBUTED MPC CONTROLLER

In the bi-level MPC controller the rough long-term MPC controller in the higher level of control has the same setting as the one introduced in Section 2.4.2. The parameters of the detailed short-term MPC controller in the lower level of control are similar to those of the conventional MPC controller explained in Section 2.4.2. Note that since the prediction interval of the detailed short-term MPC is 15 min, within one operation sampling time (i.e., 1 h) of the rough long-term MPC controller the short-term prediction horizon remains 3, except for the short-term control time step corresponding to the 50th minute, for which the short-term prediction horizon will be 2 and for the short-term control time step corresponding to the 55th minute, for which the short-term prediction horizon will be 1.

For the integrated flow and emission model in the adapter block (see Fig. 2.8), the detailed versions of the S-model and VT-micro are used. Thus, the rough control inputs determined via (2.2) and the predicted demands for the upcoming 1 h (see Fig. 2.12) are used to estimate the expected realized total emissions for the upcoming 1 h. This value is initially distributed via the adapter block evenly among the detailed short-term control time steps. After every control time step, the upper bounds for the cumulative emissions for the remaining control time steps are updated by evenly re-distributing the value of the previous upper bound minus the value of the cumulative emissions realized in the

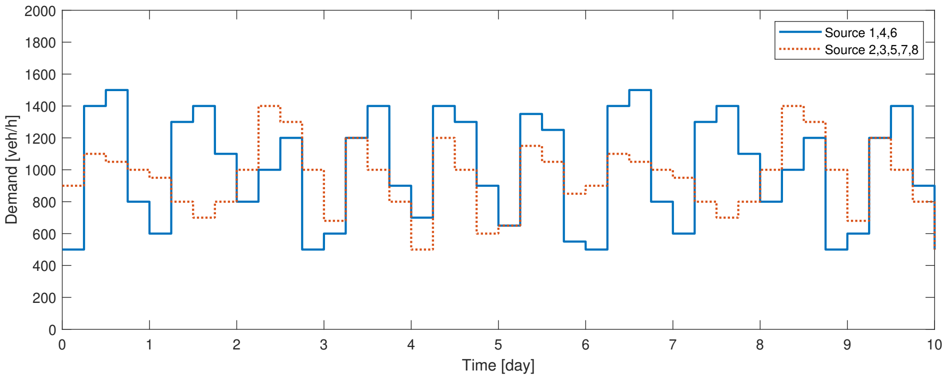

Figure 2.13: Demand profile for 10 days in case study 2

last control sampling time. The updated upper bounds for the cumulative emissions are used by the detailed short-term MPC optimization problem to determine an optimal control input sequence that will be injected into the controlled system for the upcoming control sampling time (5 min).

Remark 2. *The proposed bi-level MPC framework can be extended to larger simulation windows, e.g., one month or one year, where additional intermediate adapter blocks can be included. For instance, one rough adapter block allocates the estimated total emissions for the entire simulation window (e.g., a month) over the individual days in the month, and a second detailed adapter block distributes these daily upper bounds over the individual hourly intervals. This approach will make the procedure computationally more efficient.*

2.4.3. SETUP FOR CASE STUDY 2

In this case study, a larger simulation interval (i.e., 10 days) is considered to further assess the ability of the proposed framework to fulfill the long-term control task. This case study shares the same settings as case study 1, including the urban traffic network, traffic flow and emission model, and the controllers. The only difference is the traffic demand, which extends over a longer period (see Figure 2.13), and the same noise is added as in Section 2.4.1. Accordingly, the rough traffic model is modified with a simulation sampling time of 2 hours. The high-level MPC controller has a control and operation sampling time of 6 hours. All the controllers introduced in case study 1 are also implemented for this case study, and their control performance is compared. Moreover, the cumulative emission constraint on CO_2 over the 10-day simulation interval is 2 million kg.

2.4.4. RESULTS AND DISCUSSION

All controllers were implemented in MATLAB version R2019b running on a PC with an Intel Xeon Quad-Core E5-1620 V3 CPU with a clock speed of 3.5 GHz. Due to the nonlinear dynamics of the urban traffic network, for all the optimization-based controllers, the function `fmincon` from MATLAB has been used together with the SQP algorithm [15]. Moreover, due to the non-convex nature of the optimization problems, in order to avoid selecting local optima that may result in a performance for the controller that is (much) worse than that of the global optimum, a number of off-line experiments have been conducted to determine suitable numbers of optimization starting points for achieving near-global optima. Consequently, 10 and 15 starting points for, respectively, the rough long-term and the detailed short-term MPC optimization problems are considered. Moreover, 15 starting points are considered for the optimization problem of conventional MPC. Based on the off-line experiments, the parameters of `fmincon` were also determined such that a balance is achieved between accuracy and computational efficiency of the solver. So for the `fmincon` stopping criterion the values of the cost function tolerance, step tolerance, and constraint tolerance are selected to be 10^{-2} for the detailed short-term MPC and 10^{-1} for the rough long-term MPC. For the cost functions of (2.1), (2.2), and (2.3) $\lambda_{CO_2} = 0.005$ was considered, where the order of this weight corresponds to the relative orders of the total time spent of the vehicles and the total emissions of CO_2. The rest of the weights are set to 0.

Remark 3. *In case the optimization solver fails to find a feasible solution for an MPC optimization problem, it switches to another version of the problem, where the cumulative constraint on the emissions of* CO_2 *is excluded. A penalty is then added to the cost function with a weight equal to* 0.48. *This weight should be tuned carefully: with a very large value, the solver determines solutions that compromise reduction of the traffic congestion in order to decrease the total emissions of* CO_2*, especially for the short-term predictions, which impose short-sighted decision making. In such cases, the controller causes the vehicles to idle instead of traveling freely, since idling vehicles emit the least* CO_2 *per time step.*

Table 2.4 presents the results of the simulations for scenarios 2–6, including the CPU time and the realized values of TTS, TE, cost, and the change (in %) in the objective function (i.e., the weighted sum of the TTS and the TE) compared to the benchmark fixed-time controller for all the implemented controllers.

Overall, all controllers perform better than the fixed-time controller, while the MPC-based methods outperform the other controllers in terms of the realized values of TTS and TE, except for the single-level rough long-term MPC, which cannot guarantee the performance outside of the bi-level control architecture. For a few certain scenarios, the non-MPC methods can achieve a performance comparable to the MPC-based methods with negligible CPU time, but their performance cannot be guaranteed for all the scenarios. Furthermore, since some controllers (e.g., responsive controller and OPAC) cannot explicitly consider the constraints on the emissions, their realized TE values are much higher than those of the MPC-based methods. In addition, the bi-level MPC controller performs better than the conventional MPC, particularly in terms of the CPU time (i.e., in all cases the computational speed corresponding to the bi-level MPC controller is more than twice smaller than that of the conventional MPC controller). The bi-level MPC controller achieves a performance that is comparable to the large-horizon conventional MPC in terms of TTS and TE, but with significantly less CPU time. Moreover, during the simulations it was noticed that in all cases the conventional MPC controller failed to find a feasible solution under the given constraint, and hence it had to switch to the unconstrained version of the optimization problem and include a penalty term in the cost function. As a result, the computational complexity increased significantly and solutions that were obtained resulted in slightly poorer performance compared to the bi-level MPC controller. Due to the use of a higher-level rough MPC controller and the adapter block in the proposed bi-level architecture, however, the detailed short-term MPC controller most often received an upper bound for the cumulative constraint that prevented the corresponding constrained optimization problem to become infeasible.

As an extra remark, from Table 2.4 it is deduced that for scenario 4, the fixed-time controller performed better than all other controllers, except for the bi-level MPC-based controller. This is because the green time corresponding to the fixed-time control policy (i.e., 50% of the cycle time) is very close to the optimal solution for this scenario. Moreover, it has been verified that the MPC-based controllers result in a similar performance as the fixed-time controller. In addition, the performance of the opt. fixed-time controller cannot be guaranteed due to the quality of the historical data.

Table 2.5 presents the simulation results of the different controllers in case study 2. It is shown that the proposed bi-level control framework achieves the best control performance in terms of both TTS and TE, when considering a long-term green mobility control task (i.e., 10 days). In addition, the bi-level MPC framework is more computationally efficient than other MPC-based methods. This case study indicates that the proposed bi-level control framework is able to address long-term control objectives and long-term constraints that cannot be handled efficiently with conventional MPC control methods.

2.5. CONCLUSIONS

This chapter proposed a novel bi-level temporally-distributed MPC approach in order to tackle the challenge of high computational burden for complex constrained optimization problems with different time scales. Consequently, we have introduced two linked short-term and long-term MPC optimization problems. In the proposed framework, the rough long-term MPC problem is solved by a supervisory controller that may use a different prediction model, control sampling time, and operation time than the detailed short-term MPC problem. The controller corresponding to the detailed short-term MPC problem is implemented at the lowest control level and directly controls the system. The supervisory MPC controller determines new adaptive upper bounds for the constraints of the detailed short-term MPC problem, based on the rough long-term solutions. We have implemented the proposed control approaches to an urban traffic network in order to achieve green mobility. The results of the case study show that the proposed bi-level MPC controller outperforms other conventional control methods used for urban traffic control in terms of the total time spent, total emissions of CO_2, and CPU time. More specifically, the bi-level MPC controller has shown to require a computation time less than half of the computation time of a conventional MPC controller.

It is expected that for larger spatial and temporal scales of the network, the difference between the computation time of the bi-level MPC controller and the conventional MPC controller becomes more significant. Moreover, for future work we propose to use a more sophisticated adapter block, with several levels that distribute the upper bound of the constraints among various temporal scales. The proposed bi-level MPC architecture provides the opportunity of giving different weights to various costs in different temporal scales or for considering completely different cost functions in different temporal scales, while incorporating the inter-linked dynamics. Therefore, applying the proposed approach to various complex and non-linear dynamical systems and considering variations in the weights and costs in different temporal scales is an interesting topic for future work.

Table 2.4: CPU time in [s], total time spent (TTS) in [h], total emissions (TE) of CO_2 in [kg], cost, and TTS and TE compared to the benchmark fixed-time controller (in %) within the entire 6-hour simulation time window for Scenarios 2–6 of **case study 1**. For the abbreviations of the controllers: Opt. fixed-time is the optimized fixed-time controller; Conv. MPC is the conventional MPC controller; L-hori. MPC is the conventional MPC controller with a large prediction horizon; L-term MPC is the single-level long-term MPC controller. The notation '–' means that the item is not applicable to that controller. Note that the CPU time of the long-term MPC controller corresponds to a larger control sampling time than the other MPC controllers.

(a) **Scenario 2**

	Fixed-time	Responsive	OPAC	Opt. fixed-time	Conv. MPC	L-hori. MPC	L-term MPC	Bi-level MPC
CPU time [s]	–	–	–	–	2676	17223	269.6	1372
CPU time per control step [s]	–	–	–	–	37.17	239.2	44.93	15.06
TTS [hour]	5487.2	4815.0	4674.6	5287.2	4715.4	4675.2	5252.3	4674.9
TE(CO_2) [kg]	80232	76609	77571	79543	75484	75062	79198	75445
Objective function	5888.2	5198.0	5062.5	5684.9	5092.8	5050.5	5648.3	5052.1
Obj. value compared with Opt. fixed-time	+3.58%	-8.56%	-10.95%	–	-10.42%	-11.16%	-0.64%	-11.13%

(b) **Scenario 3**

	Fixed-time	Responsive	OPAC	Opt. fixed-time	Conv. MPC	L-hori. MPC	L-term MPC	Bi-level MPC
CPU time [s]	–	–	–	–	2917	22491	236.1	1608
CPU time per control step [s]	–	–	–	–	40.52	312.4	43.85	18.34
TTS [hour]	6790.3	4754.2	4603.9	4844.2	4685.1	4612.2	5185.2	4605.9
TE(CO_2) [kg]	90333	77773	78700	78245	77179	76380	80432	76745
Objective function	7242.0	5143.0	4997.4	5235.4	5071.0	4994.1	5587.4	4989.6
Obj. value compared with Opt. fixed-time	+38.33%	-1.76%	-4.55%	–	-3.14%	-4.61%	+6.72%	-4.69%

(c) **Scenario 4**

	Fixed-time	Responsive	OPAC	Opt. fixed-time	Conv. MPC	L-hori. MPC	L-term MPC	Bi-level MPC
CPU time [s]	–	–	–	–	3054	8956	153.4	1580
CPU time per control step [s]	–	–	–	–	42.42	124.4	25.57	17.50
TTS [hour]	4058.7	4081.2	4058.9	4399.4	4067.7	4058.7	4058.7	4058.7
TE(CO_2) [kg]	67612	67728	69990	68085	67395	67612	67612	67400
Objective function	4396.8	4419.8	4408.8	4739.8	4404.7	4396.7	4396.7	4395.7
Obj. value compared with Opt. fixed-time	-7.24%	-6.75%	-6.98%	–	-7.07%	-7.24%	-7.24%	-7.26%

(d) **Scenario 5**

	Fixed-time	Responsive	OPAC	Opt. fixed-time	Conv. MPC	L-hori. MPC	L-term MPC	Bi-level MPC
CPU time [s]	–	–	–	–	2990	12736	339.2	1598
CPU time per control step [s]	–	–	–	–	41.53	176.9	56.5	17.94
TTS [hour]	4426.2	4389.9	4329.4	4329.5	4352.1	4334.4	4330.4	4329.2
TE(CO_2) [kg]	72783	72445	74152	72482	71919	71638	71840	71858
Objective function	4790.1	4752.1	4700.2	4691.9	4711.7	4692.6	4689.6	4688.5
Obj. value compared with Opt. fixed-time	+2.09%	+1.28%	+0.18%	–	+0.42%	0.00%	0.00%	0.00%

(e) **Scenario 6**

	Fixed-time	Responsive	OPAC	Opt. fixed-time	Conv. MPC	L-hori. MPC	L-term MPC	Bi-level MPC
CPU time [s]	–	–	–	–	3114	18771	254	1577
CPU time per control step [s]	–	–	–	–	43.26	260.7	42.3	18.63
TTS [hour]	6686.7	4889.9	4722.6	4734.7	4760.6	4722.2	4975.2	4730.3
TE(CO_2) [kg]	88708	77811	78769	77120	76610	76231.4	78390	76602
Objective function	7130.2	5279.0	5116.4	5120.3	5143.6	5103.4	5367.1	5113.3
Obj. value compared with Opt. fixed-time	+39.25%	+3.10%	0.00%	–	+0.46%	-0.33%	+4.82%	-0.14%

Table 2.5: CPU time in [s], total time spent (TTS) in [h], total emissions (TE) of CO_2 in [kg], cost, and TTS and TE compared to the benchmark fixed-time controller (in %) within the entire 6-hour simulation time window of **case study 2**. For the abbreviations of the controllers: Opt. fixed-time is the optimized fixed-time controller; Conv. MPC is the conventional MPC controller; L-hori. MPC is the conventional MPC controller with a large prediction horizon; L-term MPC is the single-level long-term MPC controller. The notation '–' means that the item is not applicable to that controller. Note that the CPU time of the long-term MPC controller corresponds to a larger control sampling time than the other MPC controllers.

	Fixed-time	Responsive	OPAC	Opt. fixed-time	Conv. MPC	L-hori. MPC	L-term MPC	Bi-level MPC
CPU time [s]	–	–	–	–	105698	416035	20363	65432
CPU time per control step [s]	–	–	–	–	36.7	144.5	509.1	22.7
TTS [$\times 10^3$ hour]	188.830	160.993	159.611	167.606	179.748	159.527	169.357	159.578
TE(CO_2) [$\times 10^3$ kg]	2835.165	2610.805	2703.496	2655.158	2707.206	2586.102	2684.967	2598.886
Objective function value [$\times 10^3$]	203.006	174.047	173.129	180.882	193.284	172.458	182.782	172.572
Obj. value compared with Opt. fixed-time	+12.23%	-3.78%	-4.29%	–	+6.86%	-4.66%	+1.05%	-4.59%

3

GRAMMATICAL-EVOLUTION-BASED PARAMETERIZED MODEL PREDICTIVE CONTROL FOR URBAN TRAFFIC NETWORKS

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 chapter 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%.

Parts of this chapter have been published in [90], to which this PhD candidate contributes significantly.

3.1. 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.

3.1.1. TRAFFIC SIGNAL CONTROL

Traffic signal control has evolved over the years. Webster proposed one of the first traffic signal control methods for minimizing the delay per vehicle [195]. 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 were proposed to control multiple intersections at the same time [116], [163]. 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 [84], [172]. 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 [56], [77], [132] 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) [158] is a model-based control method that is widely used in different industrial areas [3], [156], and it has shown to be promising for urban traffic signal control [205].

3.1.2. 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 [88], [112], [113], [204], 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 [203], [204] for instance, the traffic network was divided into multiple subnetworks, each corresponding to one local optimization problem that takes the interactions with the neighbouring subnetworks into account. In [161], 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.* reformulated the nonlinear and nonconvex MPC optimization problem as a computationally efficient mixed-integer linear optimization problem (MILP) [114]. In this chapter, 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 [60], [119], [154], [207]. PMPC has shown promising results regarding computational efficiency in control of urban and freeway traffic networks [89], [100], [207], via substantially reducing the number of optimization decision variables with limited decrease in the performance. However, the parameterized control laws in [89], [100], and [207] are handcrafted based on expert knowledge and experiences and are therefore difficult to design.

3.1.3. MAIN CONTRIBUTIONS

The main contributions of this chapter 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 [89], 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 hand-crafted parameterized control laws from [89].

3.1.4. OUTLINE OF THE CHAPTER

The remainder of this chapter is organized as follows. First, in Section 3.2 we discuss the principles behind conventional MPC and PMPC and provide the necessary background on the urban traffic model that is used. In Section 3.3, 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 3.4. In Section 3.5 we present, compare, and discuss the effectiveness of the proposed training frameworks and the resulting parameterized control laws. Finally, in Section 3.6 we draw final conclusions and provide some suggestions for future work.

3.2. 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.

3.2.1. URBAN TRAFFIC PREDICTION MODEL

We use the S-model as the prediction model for the PMPC controllers since this model provides a suitable balance between accuracy and computational complexity [88], [112]. 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 chapter. For more details, we refer the reader to [112] and [88].

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. 3.1). 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 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 chapter, 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

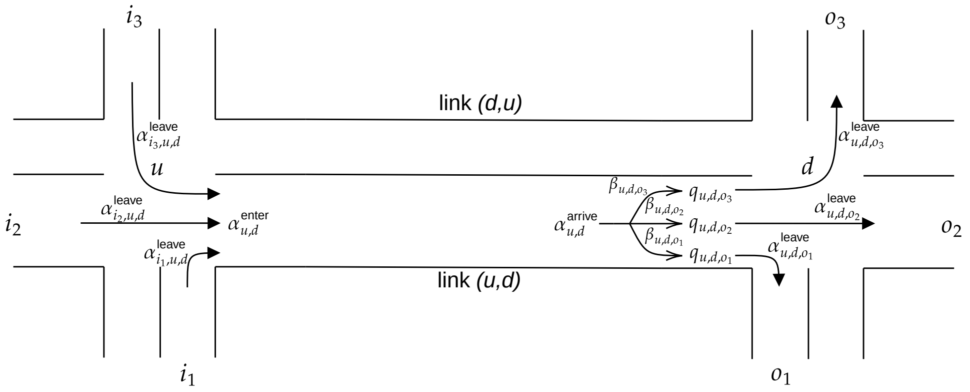
i_3
o_3
link (d,u)
$\alpha^{\text{leave}}_{i_3,u,d}$
u
d
$\alpha^{\text{leave}}_{u,d,o_3}$
β_{u,d,o_3}
β_{u,d,o_2}
β_{u,d,o_1}
q_{u,d,o_3}
q_{u,d,o_2}
q_{u,d,o_1}
i_2
$\alpha^{\text{leave}}_{i_2,u,d}$
$\alpha^{\text{enter}}_{u,d}$
$\alpha^{\text{arrive}}_{u,d}$
$\alpha^{\text{leave}}_{u,d,o_2}$
o_2
$\alpha^{\text{leave}}_{i_1,u,d}$
$\alpha^{\text{leave}}_{u,d,o_1}$
link (u,d)
i_1
o_1

Figure 3.1: A link in the S-model connecting two traffic-signal-controlled intersections, based on [112].

are updated every simulation time step k by

$$n_{u,d}(k+1) = n_{u,d}(k) + \left(\alpha^{\text{ent}}_{u,d}(k) - \alpha^{\text{leave}}_{u,d}(k)\right) \cdot c_d, \tag{3.1}$$

$$q_{u,d,o}(k+1) = q_{u,d,o}(k) + \left(\alpha^{\text{arr}}_{u,d,o}(k) - \alpha^{\text{leave}}_{u,d,o}(k)\right) \cdot c_d, \tag{3.2}$$

$$q_{u,d}(k) = \sum_{o \in O_{u,d}} q_{u,d,o}(k), \tag{3.3}$$

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.,

$$\alpha^{\text{leave}}_{u,d,o}(k) = h(\boldsymbol{x}(k), g_{u,d,o}(k)), \tag{3.4}$$

with $h(\cdot,\cdot)$ a nonlinear function taking different traffic conditions into account (see [112] 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:

$$c_d = y_d + \sum_{i=1} g_{d,i}(k), \tag{3.5}$$

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 is the number of phases for intersection d.

3.2.2. PARAMETERIZED MODEL PREDICTIVE CONTROL

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

$$\begin{aligned} \min_{\boldsymbol{g}(k)} & \big(w_{\mathrm{TTS}} J_{\mathrm{TTS}}(k) + w_{\mathscr{D}} \mathscr{D}(\boldsymbol{g}(k)) + w_{\mathscr{Q}} \mathscr{Q}(k) \big) \\ \text{s.t.} \quad & \boldsymbol{x}(k+j+1) = f(\boldsymbol{x}(k+j), \boldsymbol{g}(k+j)), \\ & \boldsymbol{g}_{d,\min} \le \boldsymbol{g}_d(k) \le \boldsymbol{g}_{d,\max} \quad \forall d \in J, \\ & (3.5), \end{aligned} \tag{3.6}$$

holds for $j = 0, \ldots, N_{\mathrm{p}} - 1$ and where $J_{\mathrm{TTS}}(k)$ 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, $\mathscr{D}(\boldsymbol{g}(k))$ and $\mathscr{Q}(k)$ 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 3.5.1. Furthermore, $f(\cdot,\cdot)$ is the prediction model (i.e., the S-model explained in Section 3.2.1), $\boldsymbol{x}(k)$ is the state vector of the model as defined in Section 3.2.1, $\boldsymbol{g}_d(k)$ contains the phase times at intersection d at time step k, $\boldsymbol{g}(k)$ is a column vector containing $\boldsymbol{g}_d(k), \boldsymbol{g}_d(k+1), \ldots, \boldsymbol{g}_d(k+N_{\mathrm{p}}-1)$ for all $d \in J$, $\boldsymbol{g}_{d,\min}$ 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}, $w_{\mathscr{D}}$, and $w_{\mathscr{Q}}$ the weights that describe the importance of the different control objectives, and (3.5) is the equality constraint on the phase times.

In order to reformulate (3.6) 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:

$$\min_{\boldsymbol{\theta}} \big(w_{\mathrm{TTS}} J_{\mathrm{TTS}}(k) + w_{\mathscr{D}} \mathscr{D}(\boldsymbol{g}(k,\boldsymbol{\theta})) + w_{\mathscr{Q}} \mathscr{Q}(k) \big). \tag{3.7}$$

which in addition to the constraints given by (3.6) is also subjected to the parameterized control law that calculates the phase times, i.e., for $j = 0, \ldots, N_{\mathrm{p}} - 1$:

$$\boldsymbol{g}_d(k+j, \boldsymbol{\theta}_d) = \mu_d(\boldsymbol{x}(k+j), \boldsymbol{\theta}_d) \tag{3.8}$$

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, $\boldsymbol{\theta}_d = \left[\theta_1, \ldots, \theta^{\theta}_{N_d}\right]^\top$, 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 $d \in J$.

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 $\boldsymbol{\theta}$ 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 in which the decision variables are held constant over several time steps to reduce the number of decision variables [21]. This yields a trade-off between performance and computational complexity.

RELATIVE QUEUE LENGTHS PARAMETERIZED CONTROL LAW

Later on, in the case study, we will use the best performing parameterized control law of [89], 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 chapter. 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 chapter. For more details, we refer the reader to [89].

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

$$g_{d,j} = \overline{g}_d + \frac{q^{\text{ph}}_{d,j} - \overline{q}^{\text{ph}}_d}{\sum_{i=1}^{N^{\text{ph}}_d} q^{\text{ph}}_{d,i} + \kappa_q} \cdot \theta_{d,1} + \frac{\alpha^{\text{ph,arr}}_{d,j} - \overline{\alpha}^{\text{ph,arr}}_d}{\sum_{i=1}^{N^{\text{ph}}_d} \alpha^{\text{ph,arr}}_{d,i} + \kappa_\alpha} \cdot \theta_{d,2} \tag{3.9}$$

where $\overline{g}_d$ is the mean green time during one cycle at intersection d, N^{ph}_d 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.

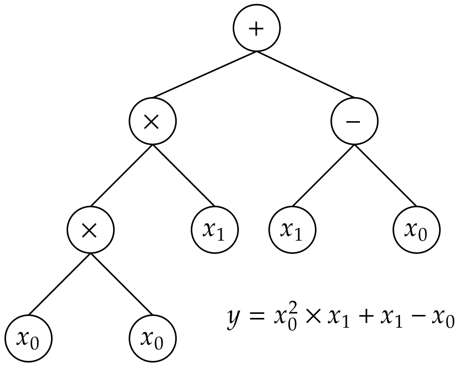

Figure 3.2: Tree representation of genetic programming for a symbolic regression problem. Here, the function set contains the mathematical operators and the terminal set contains x_0 and x_1.

3.3. GRAMMATICAL EVOLUTION

Grammatical Evolution is a form of genetic programming that produces functions based on a user-defined context-free grammar [141] and an evolutionary algorithm for evolving the functions [139].

3.3.1. GENETIC PROGRAMMING

Techniques that are used for the evolvement of functions with evolutionary algorithms are called genetic programming [104]. Genetic programming uses a (derivation) tree-based structure to represent the functions, where these trees can be evaluated in a recursive manner (see Figure 3.2). The resulting functions can consist of complex programming languages or can be more simple curve-fitting models or symbolic regressions [155]. For our application, the produced functions are parameterized control laws.

The basic genetic programming algorithm works with a function set and a terminal set [155]. 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 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 3.3.3). 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 as no constraints are set on the output of the algorithm [155]. 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 (3.5)). 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.

3.3.2. 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 [80]. Context-free grammars are often defined in Backus-Naur form [168], 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 Figure 3.3. 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., the arithmetic operators, the variables x, and the numbers) are in the production rules of some of the the non-terminals.

The grammar in Figure 3.3 can be used for symbolic regression purposes, but more advanced grammars can be used for, e.g., solving the Sante Fe ant trail problem [140], the control of femtocell network coverage [76], and finance and economics [16]. 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 Figure 3.3.

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 [89] the production rules are used to ensure that the parameters of the parameterized control laws appear in

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

Figure 3.3: The production rules of a context-free grammar in Backus-Naur form, based on [185], 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 conditions of if-statements, which leads to an efficient optimization step during the training of the parameterized control laws.

3.3.3. 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 [89], 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 [4], [141]. 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 [47].

3.3.4. CONTINUOUS GRAMMARS

In the grammar of [89], 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.

3.4. 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).

3.4.1. 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 (3.5) can be implicitly satisfied. We have:

$$g_{d,j}(k) = \frac{y_{d,j}}{\sum_{j=1}^{N_d^{\mathrm{ph}}} y_{d,j}} \cdot g_{d,\mathrm{tot}}(k), \tag{3.10}$$

where $y_{d,j}$ is a score for the j-th phase of intersection d that describes the importance of a certain phase with respect to the other phases, and $g_{d,\mathrm{tot}}$ 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:

$$y_{d,j} = f_{\mathrm{s}}\left(n_{d,j}^{\mathrm{ph}}, q_{d,j}^{\mathrm{ph}}, \alpha_{d,j}^{\mathrm{ph,ent}}, \alpha_{d,j}^{\mathrm{ph,arr}}, \theta_{d,1}, \theta_{d,2}\right) + \kappa_y, \tag{3.11}$$

where κ_y is a positive constant used to avoid a zero score, $n_{d,j}^{\mathrm{ph}}, q_{d,j}^{\mathrm{ph}}, \alpha_{d,j}^{\mathrm{ph,ent}}, \alpha_{d,j}^{\mathrm{ph,arr}}$ are respectively the total number of vehicles, queue lengths, and number of entering vehicles and arriving vehicles on the lanes that correspond to the j-th 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 (3.11) are the sums of the states corresponding to all the links for the same phase, instead of the average values used in [89]. 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 Figure 3.4 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 $f_{\mathrm{s}}(\cdot)$ in (3.11), 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

$$\mathrm{div}(x) = \frac{1}{x+1}, \tag{3.12}$$

where x is assumed to be non-negative. The entries of the non-terminal ⟨func⟩ are used to introduce nonlinearity to the score function (3.11).

In contrast to the grammar in [89] that also exploits the score-based structure to decide green time length, the grammar used in this chapter 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 $n_{d,j}^{\mathrm{ph}}$, $q_{d,j}^{\mathrm{ph}}$, $\alpha_{d,j}^{\mathrm{ph,ent}}$, $\alpha_{d,j}^{\mathrm{ph,arr}}$ can be realized by the grammar due to the second recursive rule in Figure 3.4. Moreover, the parameters $\theta_{d,1}$ and $\theta_{d,2}$ are also present in the grammar, which

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

Figure 3.4: The GE-based grammar used to obtain the score of each phase based on (3.11).

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 chapter more flexible than the one in [89] 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.

3.4.2. FRAMEWORK-1: MPC-MIMICKING

The scheme of Framework-1 is shown in Figure 3.5. 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 [89]. 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:

$$
\boldsymbol{x}_d^{\text{train}}(k) = \left[\boldsymbol{n}_d^{\text{ph}}(k)^\top, \boldsymbol{q}_d^{\text{ph}}(k)^\top, \boldsymbol{\alpha}_d^{\text{ph,ent}}(k)^\top, \boldsymbol{\alpha}_d^{\text{ph,arr}}(k)^\top \right]^\top, \tag{3.13}
$$

$$
\boldsymbol{g}_d^{\text{train}}(k) = \left[g_{d,1}(k), \ldots, g_{d,N_d^{\text{ph}}}(k) \right]^\top, \tag{3.14}
$$

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

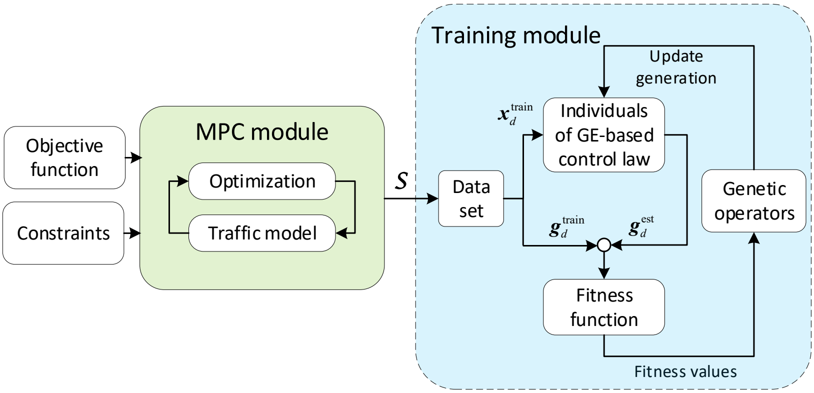

Figure 3.5: The diagram of Framework-1

intersections in the traffic network during a simulation interval $[0, k_{\mathrm{s}} t_{\mathrm{s}})$ form the training data set that is expressed as:

$$\begin{aligned} S = \Big\{ &\big(\boldsymbol{x}_1^{\text{train}}(1), \boldsymbol{g}_1^{\text{train}}(1)\big), \ldots, \big(\boldsymbol{x}_1^{\text{train}}(k_{\mathrm{s}}), \boldsymbol{g}_1^{\text{train}}(k_{\mathrm{s}})\big), \ldots, \\ &\Big(\boldsymbol{x}_{|J|}^{\text{train}}(1), \boldsymbol{g}_{|J|}^{\text{train}}(1)\Big), \ldots, \Big(\boldsymbol{x}_{|J|}^{\text{train}}(k_{\mathrm{s}}), \boldsymbol{g}_{|J|}^{\text{train}}(k_{\mathrm{s}})\Big)\Big\}, \end{aligned} \tag{3.15}$$

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 4. *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 neighbours 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 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 Figure 3.5). 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 (3.10). The output of this block is the estimated green time vector $\boldsymbol{g}_d^{\text{est}}(k) = [g_{d,1}^{\text{est}}(k), \ldots, g_{d,N_d^{\text{ph}}}^{\text{est}}(k)]^\top$. 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 4, 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:

$$\frac{1}{N_\text{t}}\sum_{d\in J}\sum_{k=0}^{k_\text{s}-1}\left[\min_{\boldsymbol{\theta}_d}\left\|\boldsymbol{g}_d^{\text{train}}(k)-\boldsymbol{g}_d^{\text{est}}(k)\right\|_2^2\right], \tag{3.16}$$

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 3.3. 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 (3.7), 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 (3.5) on the green time length is implicitly satisfied by the score-based allocation function (3.10), 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 (3.7).

3.4.3. 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 Figure 3.6. 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.:

$$\begin{aligned}\tilde{\boldsymbol{x}}^{\text{train}}(k) = \Big[&\boldsymbol{n}_1^{\text{ph}}(k)^\top, \boldsymbol{q}_1^{\text{ph}}(k)^\top, \boldsymbol{\alpha}_1^{\text{ph,ent}}(k)^\top, \boldsymbol{\alpha}_1^{\text{ph,arr}}(k)^\top, \dots,\\ &\boldsymbol{n}_{|J|}^{\text{ph}}(k)^\top, \boldsymbol{q}_{|J|}^{\text{ph}}(k)^\top, \boldsymbol{\alpha}_{|J|}^{\text{ph,ent}}(k)^\top, \boldsymbol{\alpha}_{|J|}^{\text{ph,arr}}(k)^\top, k\Big]^\top.\end{aligned} \tag{3.17}$$

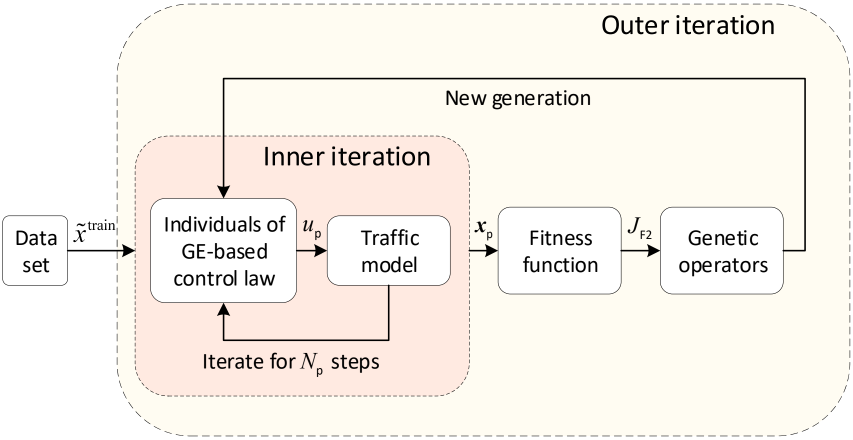

Figure 3.6: The diagram of Framework-2, where u_p is the control input generated by the parameterized control law, $\boldsymbol{x}_\text{p}$ is the vector containing all the state information and control inputs that are used in the objective function (3.18), and J_F2 is the optimized objective value of (3.18).

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 Figure 3.6 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 (3.7). 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:

$$\frac{1}{k_\text{s}}\sum_{k=0}^{k_\text{s}-1}\left[\min_{\boldsymbol{\theta}}\left(w_{\text{TTS}}J_{\text{TTS}}(k)+w_{\mathscr{D}}\mathscr{D}(\mathbf{g}(k,\boldsymbol{\theta}))+w_{\mathscr{Q}}\mathscr{Q}(k)\right)\right]. \tag{3.18}$$

In addition to the fact that the data points in Framework-2 contain more information, the built-in state evolution process within the inner 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 5. *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 it-*

erations 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 6. *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 5. 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.

3.4.4. 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 (3.10). After the phase times are calculated through (3.10), 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.

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:

$$g_1 + g_2 + g_3 = g_{\text{tot}}. \tag{3.19}$$

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 Figure 3.7 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 (3.19). 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 Figure 3.7. Therefore, these points are projected to the blue region (see the star points shown in Figure 3.7), 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

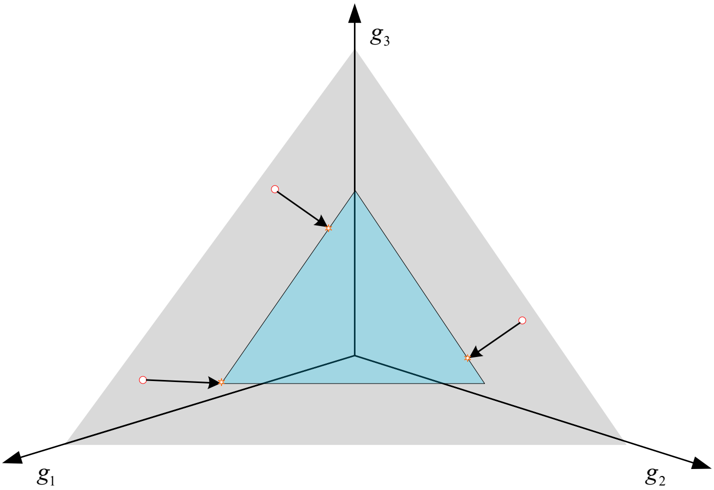

Figure 3.7: Illustration of the projection-based method for a three-phase intersection to guarantee the phase time constraint: the gray plane is the area where the sum constraint is satisfied, while the blue region is where both the bound constraints and the sum constraint are satisfied

constraint violation in the fitness function (3.18). 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 $\boldsymbol{\theta}$ in the PMPC optimization problem (3.7) 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 (3.7). This is illustrated next.

Proposition 3.4.1. *For the PMPC optimization problem* (3.7)*, replacing the explicit constraints on parameters* $\boldsymbol{\theta}$ *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 $\boldsymbol{\theta}$ that yield a feasible solution of the PMPC problem (3.7) with explicit constraints. Considering the projection-based method for a given form of the parameterized control law with parameters $\boldsymbol{\theta}$ is in fact equivalent to considering a new modified control law with the same parameters $\boldsymbol{\theta}$ for which the explicit constraints are always satisfied by construction (due to the projection). As such, the set R_2 of parameters $\boldsymbol{\theta}$ for which the modified projection-based control law will yield a feasible solution of (3.7) 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 gener-

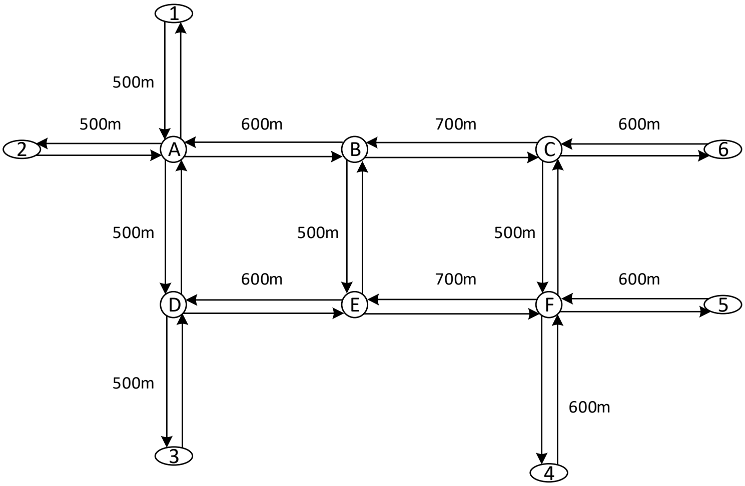

Figure 3.8: The layout and the link lengths of the urban traffic network

ated 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.

3.5. 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 [121], and the interface TraCI [198] 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.

3.5.1. SETUP

An urban traffic network is considered as shown in Figure 3.8 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 Figures 3.9 and 3.10. 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 Table 3.1.

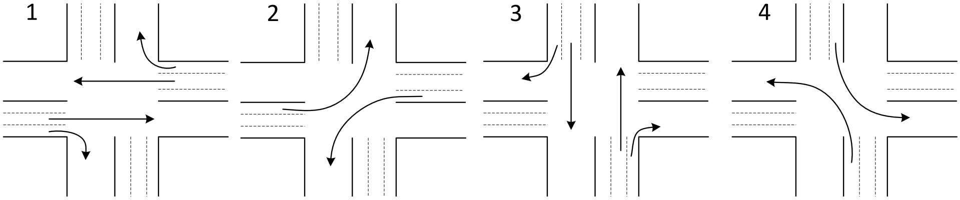

Figure 3.9: The phases of 4-phase intersections

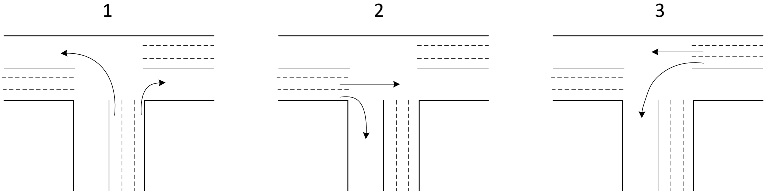

Figure 3.10: The phases of 3-phase intersections

The traffic flow demand profiles are generated with SUMO's built-in route generator, which generates routes based on flow profiles and turning rates. Three different demand profiles (see Figure 3.11) 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 3.2. 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 (3.6) and for PMPC (3.7) are identical, and the weights are $w_{\text{TTS}} = 1$, $w_{\mathcal{D}} = 1$, $w_{\mathcal{Q}} = 2$. The cost terms are defined as:

$$J_{\text{TTS}}(k) = \sum_{(u,d)\in L} \sum_{j=1}^{N_{\text{p}}} c_d \cdot n_{u,d}(k+j), \tag{3.20}$$

$$\begin{aligned}\mathcal{D}(\mathbf{g}(k)) = \Big\| \Big[&(\mathbf{g}(k)-\mathbf{g}(k-1))^\top, (\mathbf{g}(k+1)-\mathbf{g}(k))^\top, \dots, \\ &(\mathbf{g}(k+n^{\text{p}})-\mathbf{g}(k+n^{\text{p}}-1))^\top \Big]^\top \Big\|_2^2,\end{aligned} \tag{3.21}$$

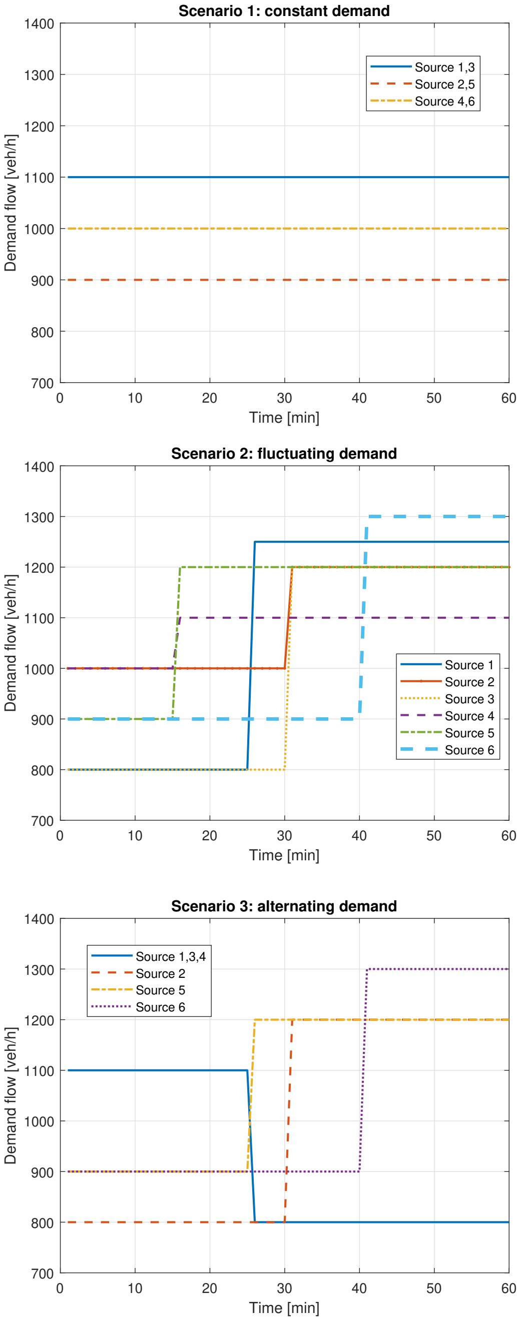

Figure 3.11: Demand profiles of the three scenarios used in the case study

$$\mathscr{Q}(k) = \sum_{j=1}^{n^{\mathrm{p}}} \sum_{d\in J} \max_{(u,d,o)\in L_d} q_{u,d,o}(k+j), \tag{3.22}$$

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:

$$\mathscr{V}(\mathbf{g}(k)) = d(\mathbf{g}(k)) + d(\mathbf{g}(k+1)) + \ldots + d(\mathbf{g}(k+n^{\mathrm{p}}-1)), \tag{3.23}$$

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

3.5.2. 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[1]. 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.

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.

---

[1] 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.

CONVENTIONAL MPC CONTROLLER

Since the sum of the phase times is fixed, the linear constraint (3.5) 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.

RQL-PMPC CONTROLLER

Using the parameterized control law (3.9), 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.

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 [47], 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 [47].

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 (3.11), the final obtained score function for 3-phase intersections is:

$$
\begin{aligned}
y^{\text{3-ph}}_{d,j} = \theta_{d,1}\left(\left(n^{\text{ph}}_{d,j} - \alpha^{\text{ph,arr}}_{d,j}\right)\cdot\frac{n^{\text{ph}}_{d,j}}{2\alpha^{\text{ph,ent}}_{d,j}+1}\cdot\left(\alpha^{\text{ph,arr}}_{d,j}+\theta_{d,2}\alpha^{\text{ph,arr}}_{d,j}\right)\right)+ \\
\theta_{d,2}\left(\left(\alpha^{\text{ph,arr}}_{d,j}\right)^2+\alpha^{\text{ph,arr}}_{d,j}-\theta_{d,2}\frac{\alpha^{\text{ph,arr}}_{d,j}}{\alpha^{\text{ph,arr}}_{d,j}+1}\right)+1.
\end{aligned}
\tag{3.24}
$$

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 6, 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:

$$
y^{\text{4-ph}}_{d,j} = \theta_{d,3}n^{\text{ph}}_{d,j}+\theta_{d,2}\alpha^{\text{ph,ent}}_{d,j}-\theta^2_{d,3}\frac{\sqrt[3]{n^{\text{ph}}_{d,j}}}{\alpha^{\text{ph,arr}}_{d,j}+n^{\text{ph}}_{d,j}-\sqrt[3]{\alpha^{\text{ph,ent}}_{d,j}}+1}-\theta_{d,1}\left(n^{\text{ph}}_{d,j}-q^{\text{ph}}_{d,j}\right)+1.
\tag{3.25}
$$

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 (3.24) not all the state information is included in the function, e.g., queue state $q^{\text{ph}}_{d,j}$ is not selected by the grammar. In score function (3.25), 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.

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:

$$
y^{\text{3-ph}}_{d,j} = \sqrt[3]{\left(n^{\text{ph}}_{d,j}q^{\text{ph}}_{d,j}\right)^2+\alpha^{\text{ph,arr}}_{d,j}-\theta_{d,1}\alpha^{\text{ph,arr}}_{d,j}\alpha^{\text{ph,ent}}_{d,j}+\theta_{d,2}n^{\text{ph}}_{d,j}}+1,
\tag{3.26}
$$

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

$$y_{d,j}^{\text{4-ph}} = \sqrt[3]{q_{d,j}^{\text{ph}} + \frac{\theta_{d,2}\alpha_{d,j}^{\text{ph,ent}}}{\alpha_{d,j}^{\text{ph,ent}}+1}+1}. \tag{3.27}$$

It is worth mentioning that the score function (3.26) is simpler than (3.24), and (3.26) 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.

3.5.3. RESULTS AND DISCUSSION

In Table 3.3 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.

3.6. CONCLUSIONS

In this chapter 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 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 [89]. 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.

Table 3.1: 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)

Table 3.2: 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)

Table 3.3: 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]	$\mathrm{TTS}^{\mathrm{rel}}$ [%]	Mean computation time [s]	Max computation time [s]	$\mathrm{CT}^{\mathrm{rel}}$ [%]
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

4

A NOVEL FRAMEWORK COMBINING MPC AND DEEP REINFORCEMENT LEARNING WITH APPLICATION TO FREEWAY TRAFFIC CONTROL

Model predictive control (MPC) and deep reinforcement learning (DRL) have been developed extensively as two independent techniques for traffic management. Although the features of MPC and DRL complement each other very well, few of the current studies consider combining these two methods for application in the field of freeway traffic control. This chapter proposes a novel framework for integrating MPC and DRL methods for freeway traffic control that is different from existing MPC-(D)RL methods. Specifically, the proposed framework adopts a hierarchical structure, where a high-level efficient MPC component works at a low frequency to provide a baseline control input, while the DRL component works at a high frequency to modify online the control input generated by MPC. The control framework, therefore, needs only limited online computational resources and is able to handle uncertainties and external disturbances after proper learning with enough training data. The proposed framework is implemented on a benchmark freeway network in order to coordinate ramp metering and variable speed limits, and the performance is compared with standard MPC and DRL approaches. The simulation results show that the proposed framework outperforms standalone MPC and DRL methods in terms of total time spent (TTS) and constraint satisfaction, despite model uncertainties and external disturbances.

Parts of this chapter have been published in [161] and [178]

4.1. INTRODUCTION

The ever-growing number of vehicles worldwide is challenging current traffic systems. Especially during morning or evening rush hours, congestion easily occurs due to insufficient road capacity. Traffic jams do not only increase commute time for individuals, but also create negative impacts for society, including environmental, economic and health issues due to the large amount of emissions and the loss of productive time. Constructing new lanes and expanding the freeway network can alleviate these issues. However, this is not always feasible due to space, financial, or environmental restrictions. Efficient management of traffic on the existing infrastructure is a promising alternative to improve traffic efficiency and safety. Among freeway control measures, ramp metering (RM) [148] and variable speed limit (VSL) [107] are the most widely used strategies, which have been shown to substantially decrease the travel delay in various real-world implementations [79], [81]. These two control measures can be either used independently or coordinated together within a control method, such as model predictive control (MPC) [24] or deep reinforcement learning (DRL) [8]. MPC and DRL are two powerful control techniques and have been studied extensively in the literature. These two methods have been applied to freeway traffic control successfully, but they also come along with their shortcomings.

MPC is a model-based and optimization-based control approach, and has become mature in terms of stability and feasibility theory since 1990s [128]. It is widely applied in industry and many other fields, due to its robustness and ability to explicitly deal with input and state constraints, thus satisfying safety requirements, which is a crucial concern in many real-world applications. However, an accurate mathematical model is usually required for MPC to guarantee the closed-loop performance, while acquiring such a model is commonly not possible in practice. Particularly, large-scale and complex systems, such as freeway networks, lead to highly nonlinear and non-convex optimization problems with multiple variables, which are difficult to solve in real time [73]. Even though some efficient MPC approaches have been developed to improve the computational efficiency of MPC [90], [207], the optimality and satisfaction of state constraints cannot always be guaranteed in case of model mismatches and external disturbances. Robust and stochastic MPC methods [129], including tube MPC [106] and scenario-based MPC [22], can address uncertainties to some extent. However, these methods require assumptions or descriptions of the uncertainties that are often difficult to validate.

DRL is a recent technique that has shown its success and potential in the field of control, including intelligent traffic control. Unlike conventional reinforcement learning algorithms, artificial neural networks are deployed in DRL to deal with large-dimension state and action spaces. This addresses the so-called issue of the curse of dimensionality [134]. Nevertheless, DRL still suffers from several challenges in real-world applications [36]. For example, safety constraints are of significant importance in operation of real-world systems, while satisfaction of constraints cannot be guaranteed during the learning phase and in implementation of DRL. In addition, the sample efficiency issue and delayed reward for large-scale systems (e.g., for traffic networks) remain considerable challenges for DRL that are still active research topics [36].

Both MPC and DRL have their advantages and disadvantages, and they complement each other well (see Table 4.1). On the one hand, MPC suffers from degraded performance due to model uncertainties and external disturbances. Moreover, large-scale systems introduce multiple variables and long prediction horizons, which make MPC computationally intractable in real time. On the other hand, DRL can naturally cope with uncertainties, and tackle infinite prediction horizons with negligible online computational resources. However, it is usually time-consuming to train a well-performing DRL agent from scratch, especially for complex systems. Although there are clear potential benefits of combining MPC and DRL, very limited work has been done to explore the synergy between these two methods. In addition, very little work has been done to apply combined MPC-(D)RL algorithms in the field of traffic management. One of the representative studies is [161], which applied a model-reference framework that utilizes MPC and deep Q-network algorithm to urban traffic signal control.

This chapter contributes to the state-of-the-art by proposing a novel framework for combining MPC and DRL, and by applying it to traffic management of freeway networks. To be more specific:

1. Different from the previous work [161], the newly proposed MPC-DRL framework adopts a hierarchical structure in order to incorporate the advantages of both MPC and DRL approaches. The combined framework can learn from the environment, while providing basic control performance. By taking advantage of the dynamic model knowledge and environment information, the framework can deal with uncertainties and improve sample efficiency of the learning process. In particular, an efficient MPC controller operates at the upper control level with a low control frequency to provide initial optimality while explicitly incorporating the constraints. Meanwhile, a DRL agent works at the lower control level with a high control frequency in order to modify the MPC inputs, and to compensate for model mismatches that may affect MPC. Therefore, the combined MPC-DRL framework is computationally feasible even for large freeway networks, due to the very limited online computation burden required by the high-level MPC.

2. The resulting MPC-DRL framework is implemented on a benchmark freeway network, and the results validate the effectiveness of the proposed method. In particular, the objective function of MPC and the reward function of DRL are designed properly, such that the two components are complementary with each other. In addition to MPC, DRL addresses the state and input constraints by introducing penalties on the constraint violation in the reward function. Simulation results show that the combined MPC-DRL outperforms other controllers in terms of both control performance and constraint satisfaction.

The rest of this chapter is organized as follows: Section 4.2 summarizes related work about MPC and DRL and their application in freeway traffic management, as well as the latest MPC-DRL algorithms and their applications. Section 4.3 presents and provides details on the novel MPC-DRL framework that is proposed in this chapter. Section 4.4 gives a case study that implements MPC, DRL, and the proposed MPC-DRL framework on the same benchmark network. Finally, Section 2.5 concludes this chapter and proposes topics for future work.

Table 4.1: Comparison of MPC and DRL characteristics

	MPC	**DRL**
Need a model	Yes	No
Developed stability theory	Yes	No
Developed feasibility theory	Yes	No
Handling constraints explicitly	Yes	No
Adaptive to uncertainties	No	Yes
Online computational time	High	Low
Offline computation time	Low	High

4.2. RELATED WORK

A large number of studies about traffic management of freeway networks exist in the literature, and a recent comprehensive survey is given in [173]. Among all the traffic control approaches, MPC and DRL have drawn significant attention due to their appealing features. MPC and DRL have been developed and applied for both freeway and urban traffic networks. As the case study in Section 4.4 involves a freeway traffic network, we will mainly focus on MPC and DRL for freeway traffic control in this section[1]. After that, recent results about combined MPC-DRL methods are presented.

4.2.1. MPC FOR FREEWAY TRAFFIC CONTROL

The idea of utilizing rolling horizon optimization in traffic signal control was first introduced by Gartner [56], after which the suggestion of adopting MPC in traffic signal control was formally made by De Schutter and De Moor in [31]. Since then extensive studies about MPC have been carried out in the field of traffic control, including both urban and freeway traffic networks. Particularly, an RM strategy and a VSL strategy was adopted in MPC for freeway traffic control in, respectively, [11] and [74]. These two control methods were first coordinated within MPC in the work of Hegyi *et al.* [73].

As an online optimization-based control method, MPC struggles with computational complexity, especially when the scale of the freeway network is large. Therefore, a large amount of efforts have been devoted to alleviate this issue. One major direction is to reduce the complexity of the dynamic model of the freeway network, and many efficient mathematical models have been developed to describe traffic flow dynamics. METANET is a second-order macroscopic traffic flow model that has been widely used for freeway traffic control [101], [103], since it can reproduce necessary freeway traffic phenomena such as capacity drop and blocking, with relatively simple mathematical formulations. Thus, using METANET, a trade-off is achieved between accuracy and complexity for the implementation of MPC in freeway traffic control. Another popular choice is the cell transmission model (CTM), which is a first-order model and has a more simple formulation than METANET [28], [215]. CTM is preferred in some cases since it can result in simpler optimization problems for MPC. The simplified variation of CTM, called asymmetric cell transmission model (ACTM), can even transform the optimization problem into a linear one, which can be solved efficiently for larger freeway networks and for

---

[1] It is, however, important to note that the novel MPC-DRL framework proposed in this chapter can be applied to both freeway and urban traffic networks.

longer prediction horizons [58]. However, this efficient model cannot reproduce some necessary freeway phenomena, such as weaving and capacity drop, which limits its application in cases where these phenomena are observed [147].

The other direction to improve the computational efficiency of MPC is to simplify the problem by linearizing it, or by adopting efficient optimization techniques. For example, Zegeye *et al.* [207] employed the parameterized MPC technique to reduce the number of the decision variables of the optimization problem. By introducing state-feedback control laws, the control inputs can be described as a function of the states and several function parameters. Thus, only the parameters need to be optimized to obtain the control inputs. A similar strategy was adopted by Van de Weg *et al.* [196], in which the parameterized control laws were extended to the coordinated VSL and ALINEA ramp metering approaches [146], resulting in a substantially reduced number of optimization variables that is independent of the number of actuators and the size of the prediction horizon. Jeschke *et al.* further extended this approach by using a grammatical evolution method to generate the state-feedback laws automatically, and applied it to urban traffic control [90]. In [142], Oleari *et al.* used classification and regression trees to train a state-feedback control law, in order to reproduce the behavior of the centralized MPC. Muralidharan and Horowitz [137] analyzed the nonlinearities of the nonlinear and nonconvex optimization problem, and made corresponding heuristic restrictions and assumptions to transform the original problem into a set of equivalent linear optimization problems. Similarly, Han *et al.* [69] extended the original Lighthill-Whitham-Richards (LWR) model [110] for MPC-based VSL control, in which the average speeds of the traffic flows were taken as the decision variables. This allows to formulate the original state constraints as control input constraints, and to formulate the optimization problem as a linear programming problem. Ferrara *et al.* [50] incorporated an event-trigger mechanism in the MPC framework to reduce the frequency of solving the optimization problems. Besides, the finite-horizon optimization problem within the MPC scheme was formulated as a mixed-integer linear programming problem that can be solved efficiently, thanks to the revised linear model obtained from CTM.

In order to deal with the control problem of large-scale freeway networks, sophisticated control schemes such as distributed MPC (DMPC) are considered. In a DMPC framework, a centralized MPC problem is decomposed into several small sub-problems that can be solved efficiently by corresponding local MPC controllers. Meanwhile, the local MPC controllers communicate and coordinate with their neighbors to achieve a global optimal performance. There are a lot of ways to implement DMPC in freeway traffic control that are demonstrated to be effective (see, e.g., [25], [27], [49], [53], [57], [96], [160]).

Despite the success that efficient MPC algorithms have achieved, MPC still suffers from issues that are caused by uncertainties, since it relies heavily on the prediction model. Mismatches between the (macroscopic) models and the real-world traffic systems, as well as presence of external disturbances are inevitable, which deteriorate the closed-loop performance of MPC. To address these issues, a few studies have considered robust MPC for freeway traffic control. Liu *et al.* [118] utilized a scenario-based approach [22] to describe the uncertainties as a set of scenarios with their corresponding probabilities, including global uncertainties (e.g., global weather conditions) and local uncertainties (e.g., local weather conditions, local traffic compositions, and local demands at the origins). Coordinated with DMPC, the resulting scenario-based DMPC improves the control performance for a large-scale freeway network considering some uncertainties. In [48], Ferrara *et al.* proposed a hierarchical control scheme for freeway networks with demand disturbances. Nevertheless, current robust MPC algorithms for freeway traffic control require assumptions and simplifications about the uncertainties and disturbances that are usually hard to satisfy in practice. Moreover, the extra computational burden introduced by robust control methods is another issue. Therefore, developing efficient and robust MPC algorithms for traffic management remains a challenging and urgent task.

4.2.2. DRL FOR FREEWAY TRAFFIC CONTROL

Reinforcement learning (RL) [180] is a machine learning technique that usually follows a two-stage procedure. In the first stage, the RL agent learns how to take actions by interacting with the environment/system (or a model of it), in order to maximize the notion of cumulative reward. After that, the trained RL agent is then implemented for control. RL is attracting more and more interest from the system and control community, since it can naturally deal with uncertainties and automatically learn a long-term optimal policy through interacting with the environment. In particular, the emergence of DRL algorithms significantly broadens the applications of RL and unlocks great potentials for various fields. DRL introduces neural networks to RL and thus can handle more complex state and action spaces [109]. DRL has also been studied for intelligent traffic signal control, and a recent survey is offered in [72]. However, current work mainly focuses on urban traffic signal control, while relevant research about DRL for freeway traffic control is still inadequate. Before the introduction of DRL, standard RL algorithms have already been studied and implemented for freeway traffic control. We believe that standard RL methods for freeway traffic control are also important, and associated studies are therefore briefly presented.

Table 4.2 summarizes several current representative studies of RL-based freeway traffic control. First of all, the common ground of these studies is that the freeway network can be described as a Markov Decision Process (MDP), on the basis of which RL algorithms can be implemented immediately with a properly defined environment. Therefore in Table 4.2, the definitions of the main RL components are given, including the state and action space, and the design of the reward function. By analyzing these studies, we can get some conclusions that are helpful to understand how RL algorithms have been implemented for freeway traffic control and how they can be further improved.

From Table 4.2, the common choices of state space are basically the typical states of freeway network models (i.e., density, speed, queue length, and demands), while the choice for the action space corresponds to the adopted control strategy. Meanwhile, the definition of the reward function depends on the control purpose, which is mainly about minimizing the total time spent (TTS) by the vehicles or regulating the densities to maximize the outflow. As for the simulation environment, both macroscopic models and microscopic simulators have been considered to test the proposed algorithms. Note that the best improvement of TTS with regard to the no-control case is presented as simulation result in the table for those papers that considered the reduction of TTS as their control objective.

Table 4.2 shows that Q-learning [194] as a notable standard RL algorithm has been chosen by most researchers due to its success in many applications. However, Q-learning relies on a lookup table to store Q-values of the state-action pairs, where these Q-values refer to the rewards expected for an action taken in a given state. Therefore, Q-learning can only deal with finite-dimensional state and action spaces (i.e., discrete state and action spaces), and the computational complexity explodes as the dimensions of the state and action spaces increase. That is also the reason why most studies consider a small freeway network, because one single RL agent can only operate a very limited number of VSL or RM controllers, which also makes the coordination of VSL and RM challenging. In order to deal with continuous state spaces, function approximation methods are used to estimate the Q-values, such as kNN-TD learning algorithm [120] that is based on the k-nearest neighbor clustering technique [152]. In addition, multi-agent RL (MARL) techniques [20] are needed when dealing with large-scale freeway networks, which makes the algorithms more complicated and difficult to implement. This drawback is particularly highlighted when encountering large action spaces or the coordination of VSL and RM.

The emergence of DRL techniques fully releases the potentials of RL algorithms, since the embedded deep neural networks can deal with both large state space and large action space, and they allow to use VSL and RM simultaneously in a centralized way for large-scale networks. Furthermore, the associated target network and experience replay techniques substantially improve the learning process in terms of stability and convergence [135]. However, the limitations of DRL mentioned in the previous section still apply. Most DRL algorithms can be divided into two categories: value-based methods and policy-based methods. Value-based methods inherit the idea of Q-learning to approximate the value functions for state-action pairs. One of the most well-known value-based methods is Deep Q-Network (DQN) [134], where the main limitation of DQN is that it can only deal with discrete action spaces. Policy-based methods directly search for an optimal policy that maximizes the expected accumulative long-term reward. Actor-critic algorithms [133] exploit the strengths of both value-based and policy-based methods, where an actor determines how the agent behaves and a critic evaluates the chosen action.

Despite the great progress in DRL techniques, only a very limited number of studies have applied DRL for freeway traffic control. In [62], DQN was implemented on a small freeway network with one single VSL controller that generates discrete actions. Deng *et al.* considered an actor-critic method, called Proximal Policy Optimization [171], and used it for controlling a simple freeway network with multiple RM controllers, where the actions are continuous [32]. A multi-agent DRL framework was adopted in [12] for the ramp metering control of a larger freeway network with 29 on-ramps and a continuous action space. Their simulation results showed that the control strategy can reduce the travel time with 20%. Wu *et al.* [201] considered distinct speed limits for different lanes, and they used an actor-critic algorithm called Deep Deterministic Policy Gradient (DDPG) [111]. The robustness of the DRL-based control agent was tested on environments with different driving behavior attributes. Furthermore, Greguric *et al.* proposed a novel VSL strategy based on a spatially adjustable speed limit zone, in which the position and length of the speed limit zone were adjustable [61]. This strategy was also based on DDPG, and realized by considering the involvement of Connected-Autonomous Vehicles. The same DRL method was also implemented with standard VSL and the performance of the two control methods were compared. The simulation results showed that the proposed strategy outperforms other controllers in terms of overall freeway throughput. Han *et al.* [70] adopted an off-line and on-line hybrid training framework for an actor-critic algorithm called Batch-constrained deep Q-learning [55], which is similar to DDPG. The off-line training process enables more freedom for action exploration, without damaging the real-world system. Meanwhile, Han *et al.* [70] considered a two-level DRL structure to coordinate multiple RM controllers.

To the best of our knowledge, [191] is the only paper that coordinates VSL and RM with a DRL algorithm, where both DDPG and TD3 [54] algorithms were implemented and their performance was compared. It was also shown that a centralized DRL agent can handle a large freeway network with multiple VSL-RM hybrid controllers.

However, none of the above studies considers the state constraints. For example, the queue length of the on-ramps should be constrained, otherwise it interferes with the connected urban road network and safety issues may occur. Moreover, although a lot of research has studied how to improve the practicability of learning-based methods, such as by training with real-world data or by pre-training (i.e., before implementation), there is still a huge gap between real-world deployment and simulator-based applications. DRL methods have the potential to deal with uncertain environments, but they also suffer from the requirement of a prolonged training process (i.e., low sample efficiency), as well as the lack of performance and safety guarantees. How to maintain the positive feature of DRL, while circumventing the negative sides remains an interesting and meaningful research topic.

4.2.3. COMBINED MPC AND DRL METHODS

Considering the features of both MPC and DRL, the idea of merging these two methods to exploit their complementary advantages sounds promising. Although there are a few studies that have investigated this topic, no one has implemented relevant methods in the field of traffic management, especially for freeway traffic control. Therefore, the latest work relevant to combined MPC-(D)RL methods and their applications in other fields are presented and analyzed in this subsection.

Learning-based MPC is a relevant and broad research topic. Hewing *et al.* [78] gave a comprehensive review of learning-based MPC methods, in which the authors divide the methods into three categories: learning the system dynamics, learning the controller design, and MPC for safe learning. Moreover, a survey that places more emphasis on RL-based optimal control can be found in [159]. Several studies that analyze the similarities and differences of MPC and RL include [7], [42], [59].

Table 4.2: Summary of representative reinforcement learning studies for freeway traffic control

Reference	RL algorithm	Strategy	State	Action	Reward	Simulation
Li *et al.* [109]	Q-learning	VSL	Discrete; densities of mainline and ramp	Discrete; speed limits	Deviation from critical density	CTM (macro) Reduced TTS for 49.34%
Wang *et al.* [192]	Distributed Q-learning (Value function approximation)	VSL	Continuous; densities and speeds	Discrete; speed limits	Deviation from critical density, time to collision	MOTUS (micro) Reduced TTS for 51.11%
Walraven *et al.* [190]	Q-learning (Value function approximation)	VSL	Continuous; current and predicted densities and speeds	Discrete; speed limits	Total time spent	METANET (macro) Reduced TTS for 30%
Zhu *et al.* [214]	R-MART	VSL	Discrete; densities of all the links	Discrete; speed limits	Total travle time	LDNL (macro) Reduced TTS for 18%
Lu *et al.* [123]	Q-learning	RM	Discrete; densities and demands of mainline and ramps	Discrete; metering rates	Total time spent and total waiting time	ACTM (macro) Reduced TTS for 18.5%
Zhou *et al.* [213]	Q-learning (Value function approximation)	RM	Continuous; densities of the mainline and demand of on-ramp	Discrete; metering rates	Deviation from the user-specified density	CTM (macro) Followed the desired density well
Davarynejad *et al.* [29]	Q-learning	RM	Discrete; density, queue length, current and next-step demands, current metering rate	Discrete; metering rates	Outflow and queue length	METANET (macro) Queue length maintained on the desired value
Schmidt-Dumont *et al.* [170]	Multi-agent Q-learning (Value function approximation)	VSL-RM	Continuous; density and queue length	Discrete; speed limits and red phase duration	Deviation from the desired density and outflow	METANET (macro) Reduced TTS 25.33%

The paper [138] is the earliest one that utilizes a value function to approximate the infinite-horizon objective function of MPC, where an MDP is used as the prediction model. Moreover, the prediction horizon is reduced to look only one step ahead, while accounting for the long-term value of the performance criteria. The value function can be learned gradually on-line using RL techniques, and meanwhile MPC operates with a simplified optimization problem to provide data samples. This work opened up a research direction to combine MPC and RL algorithms, and inspired consequent research. The method was extended to more general dynamics in [212], where two different value function approximations are used and implemented for various control examples, including the inverted pendulum, the double pendulum, and the acrobot. A similar idea was adopted in [209], where the Q-value function is used to replace the whole objective function of a Lyapunov-based MPC. Thus the original optimization problem is transformed into a policy iteration problem, and is solved using an actor-critic scheme. The main contribution of [209] is embedding a set of constraints based on an existing Lyapunov-based controller into the learning process of the DRL agent, thus ensuring stability. However, the learning process still struggles with a low sample efficiency and unsafe exploration issues. Arroyo *et al.* [7] further extended the method given in [138] to a realistic scenario for building energy management, by encoding domain knowledge. Then the initial complex MPC optimization problem is reformulated as an optimization problem with a prediction horizon of one step. In [7] a simulation model is extracted from the simulator via system identification, and is used as the prediction model for MPC, as well as for pre-training of the DRL agent. The simulation results show that the proposed RL-MPC approach can meet the state constraints and provide satisfying performance. However, it is not demonstrated in [7] whether or not the RL-MPC approach outperforms MPC in uncertain environments.

The above MPC-DRL combined algorithms can be categorized as objective function truncating methods. This can reduce the on-line computational complexity of MPC, while RL is used to handle the uncertain environment. Nevertheless, these algorithms still suffer from several issues. First, one-step ahead MPC optimizes the control input only for the next time step, which therefore can only guarantee the short-term safety constraints. Second, although the value function can address the constraints by introducing a penalty on constraint violations in the reward function, in this way the constraints become soft constraints that do not necessarily provide guarantees. Third, an inaccurate system model is still used for the one-step ahead optimization of MPC, which influences the optimality of the performance. Forth, optimizing the joint objective function that contains a value function can be quite challenging.

Another direction to connect MPC with RL is developed by Gros and Zanon. In [64] they proposed to use a parameterized MPC scheme instead of deep neural networks to approximate the value function and policy for the RL agent. It is shown that the MPC scheme can guarantee the optimality of the learned policy by adjusting the objective function of MPC, even with an inaccurate system model. Furthermore, they extended the algorithm by utilizing robust MPC techniques to address the safety issue of RL [206]. The method is implemented with a Q-learning algorithm and the results show that the constraints are well handled. In fact, Gros and Zanon [64], [206] are basically using RL tools to solve the MPC problem by using the connection between the parameterized

MPC and RL. However, how to parameterize the cost function of MPC is not considered in a structured way.

A different trend is to directly combine the control inputs of MPC and RL. The paper [202] proposed a framework that contains independent MPC and DDPG agents, in which the overall output is a weighted sum of the control inputs generated by MPC and DRL. The idea is to use MPC to play a guiding role by applying its control action directly to the system to obtain more effective data samples for the training of the DDPG agent, thus improving the sample efficiency. However, the weight parameter needs to be tuned by trails for various tasks, and it is not clear how the MPC and DDPG agents collaborate with each other. Zhang *et al.* [211] integrated DRL within a model reference control framework, in which the DRL module is used to compensate for the error between the real state and the desired state that is generated by the nominal system with a baseline controller.

Many variations of combined MPC-RL methods have been developed for specific applications, especially for robot control. Johannink *et al.* [92] proposed a residual RL approach that is tailored for a robot motion control task. This approach combines a hand-engineered controller and a residual RL controller. In order to deal with real-world contacts and friction that cannot be captured by modeling, the control task is decomposed into a main part and a residual part, which are solved by a conventional feedback controller and an RL controller, respectively. A temporal difference MPC approach was developed in [71], where a terminal value function is used to estimate the long-term return and a task-oriented model is learned. Both components are jointly learned by temporal difference learning. The proposed method is evaluated on a total of 92 control tasks, and shows to outperform both model-free and model-based approaches for these tasks.

There is not yet an extensive comparison study about the MPC-RL algorithms discussed above, so it is still an open question that which approach surpasses the other, and in which cases. However, it is clear that each algorithm is designed to address a specific issue or a particular task. This chapter intends to develop a novel framework that combines MPC and DRL in a flexible way, i.e., it allows the designers to freely choose the detailed MPC and DRL schemes. The framework is also designed in a hierarchical structure with multiple operation rates, such that MPC and DRL can coordinate well with each other, making the framework applicable for various complex applications. The proposed framework is tested for a freeway traffic control problem from [73], and the performance is compared with standard MPC and DRL methods.

4.3. COMBINED MPC-DRL FRAMEWORK

This section presents the proposed MPC-DRL control framework. Section 4.3.1 gives an intuitive description of the framework from a high-level point-of-view. Section 4.3.2 defines the MPC and the DRL modules. Section 4.3.3 details the learning algorithm of the framework. The mathematical notations used in this chapter are presented and defined in Table 4.3.

Table 4.3: Mathematical notations used for the combined MPC-DRL framework

Notation	Definition
k_{s}	Simulation step counter of the system
k_{d}	Control step counter of the DRL module, which also corresponds to the control step of the controlled system
k_{c}	Control step counter of the MPC module
T_{s}	Simulation sampling time of the system
T_{d}	Control sampling time of the DRL module
T_{c}	Control sampling time of the MPC module
T_{ini}	Time to initialize the freeway network before control for the simulation
T	The whole simulation time interval
$\boldsymbol{u}_{\mathrm{b}}(k_{\mathrm{c}})$	Control input of MPC at control step k_{c}
$\boldsymbol{u}_{\mathrm{s}}(k_{\mathrm{s}})$	Control input of MPC at simulation time step k_{s}
$\boldsymbol{u}_{\mathrm{rl}}(k_{\mathrm{d}})$	Action of DRL generated by the actor network at control step k_{d}
$\boldsymbol{u}'_{\mathrm{rl}}(k_{\mathrm{d}})$	Action of DRL generated by the target actor network at control step k_{d}
$\boldsymbol{u}_{\mathrm{c}}(k_{\mathrm{d}})$	Final control input given to the system at control step k_{d}
$\boldsymbol{x}(k_{\mathrm{s}})$	Real traffic state at simulation time step k_{s}
$\hat{\boldsymbol{x}}(k_{\mathrm{s}})$	Predicted traffic state at simulation time step k_{s}
$\hat{\boldsymbol{d}}(k_{\mathrm{s}})$	Predicted traffic demands at simulation time step k_{s}
$\boldsymbol{d}(k_{\mathrm{s}})$	Real traffic demands at simulation time step k_{s}
$N_{\mathrm{p,c}}$	Prediction horizon length counted in terms of the MPC control time step
$N_{\mathrm{p,s}}$	Prediction horizon length counted in terms of the simulation time step
$\boldsymbol{s}(k_{\mathrm{d}})$	State vector for the DRL agent at control step k_{d}
$r(\boldsymbol{s}(k_{\mathrm{d}}), \boldsymbol{u}_{\mathrm{rl}}(k_{\mathrm{d}}))$	Reward of the DRL agent for taking action $\boldsymbol{u}_{\mathrm{rl}}(k_{\mathrm{d}})$ when at state $\boldsymbol{s}(k_{\mathrm{d}})$
y_i	Learning target of the DRL agent for data sample i
R	Experience replay buffer of the DRL agent
N	Size of mini-batch sampled from the replay buffer
n	Number of steps to look ahead for the reward in the DRL algorithm
$w_n(k_{\mathrm{d}})$	Random noise added to the DRL actions at control step k_{d} for exploration
w_u	Scaling parameter of the DRL actions
w_p	Penalty weight in the DRL reward function

4.3.1. MPC-DRL FRAMEWORK

As illustrated in Figure 4.1, the proposed MPC-DRL framework has a hierarchical structure. The MPC module operates at the high level to provide a basic control input that is optimized over the prediction window based on the objective function of MPC with the associated nominal model and the predicted traffic demands. The objective function is given according to the control purpose (e.g., minimizing TTS), and the state and input constraints are considered explicitly during the optimization. In practice, the MPC control input $\boldsymbol{u}_{\mathrm{b}}$ is not optimal, mainly due to the mismatch between the prediction model and the real system, as well as due to the error in the predicted demands. Accordingly, the state constraints cannot be guaranteed. Note that MPC performs with a larger control sampling time T_{c} than the simulation sampling time T_{s}, such that the number of the optimization variables is substantially reduced, even with a long prediction horizon. Therefore the online optimization problem of MPC is computationally tractable.

In order to improve the optimality of the MPC input and to avoid severe constraint violations, the DRL module works at the lower level to modify the MPC input $\boldsymbol{u}_{\mathrm{b}}$ during the learning process by interacting with the real system. The state space of the DRL agent includes the freeway states $\boldsymbol{x}$ and the MPC input $\boldsymbol{u}_{\mathrm{b}}$ (see (4.10)), while the reward function

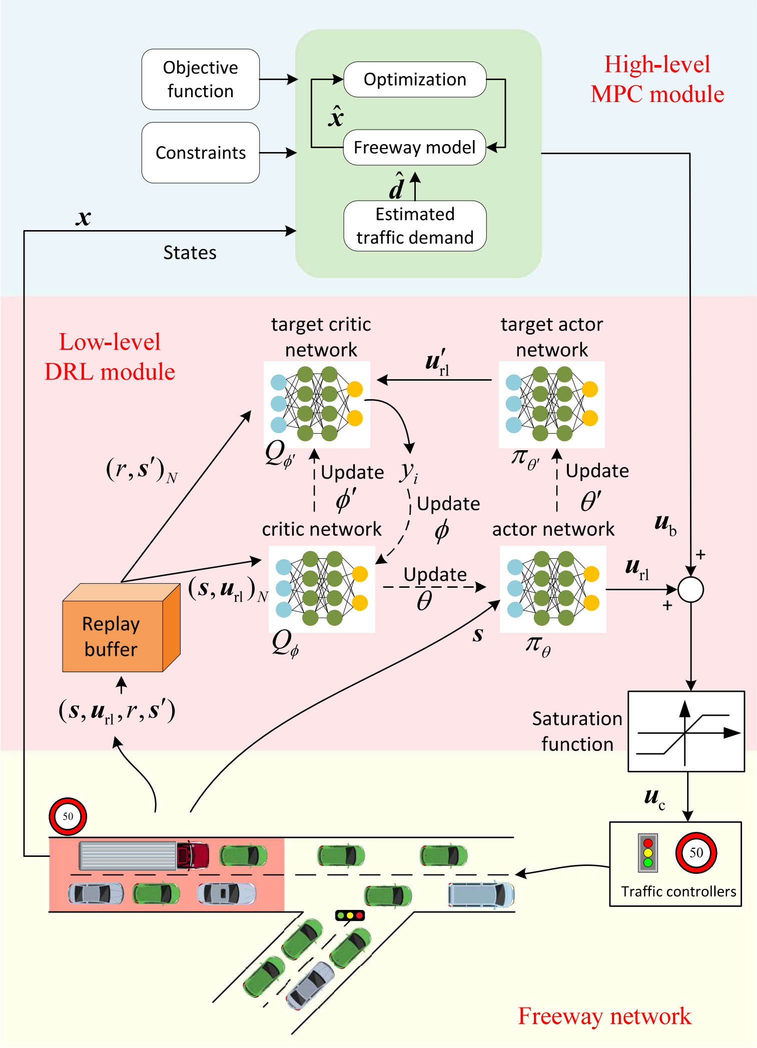

Figure 4.1: Block diagram of the hierarchical MPC-DRL control framework.

is designed to complement the objective function of MPC, such that these two modules can collaborate to optimize the overall objective. In addition, the traffic demands are also fed into the DRL agent, and a penalty on the constraint violation is added to the reward function. The action u_{rl} of DRL has the same dimension as u_{b}, but its elements have smaller magnitudes. The components of the DRL module are defined in detail in Section 4.3.2. The update algorithms of the network parameters in the figure are presented in Section 4.3.3.

Assume that the model of the freeway dynamic is discrete-time with a simulation sampling time T_{s}, and the DRL module works with a control sampling time T_{d}. Then the

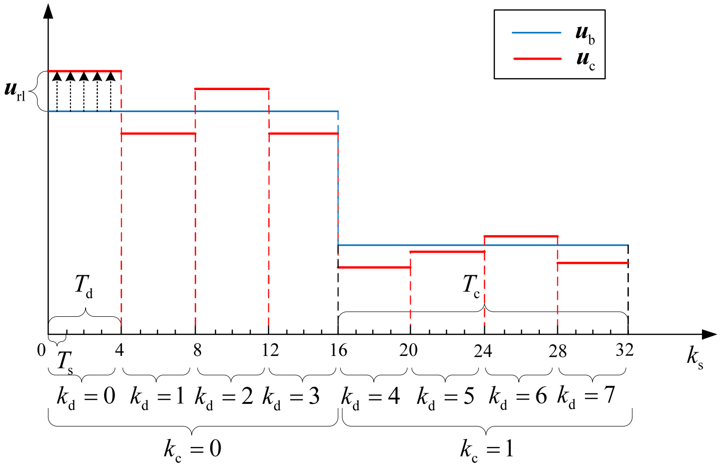

Figure 4.2: Illustration of different time scales and how DRL modifies the MPC control input.

relationship between T_{s}, T_{d}, and T_{c} are described as:

$$T_{\mathrm{c}} = m_1 \cdot T_{\mathrm{d}} = m_1 \cdot m_2 \cdot T_{\mathrm{s}}, \quad m_1, m_2 \in \mathbb{N}^+, m_1 > 1. \tag{4.1}$$

Note that for the sake of simplicity and brevity in the notations, we assume that the simulation step, DRL control step, and MPC control step coincide (see Figure 4.2). Therefore, the overall control input of the combined framework is a combination of $\boldsymbol{u}_{\mathrm{b}}$ and $\boldsymbol{u}_{\mathrm{rl}}$, which is updated every T_{d} time units. For control step k_{d} that corresponds to the MPC control step k_{c} (i.e., $k_{\mathrm{d}} T_{\mathrm{d}} \in [k_{\mathrm{c}} T_{\mathrm{c}}, (k_{\mathrm{c}}+1) T_{\mathrm{c}})$), the overall control input is given by:

$$\boldsymbol{u}_{\mathrm{c}}(k_{\mathrm{d}}) = \mathrm{sat}(\boldsymbol{u}_{\mathrm{rl}}(k_{\mathrm{d}}) + \boldsymbol{u}_{\mathrm{b}}(k_{\mathrm{c}})), \tag{4.2}$$

where a saturation function sat(·) is used to guarantee that the additive control input $\boldsymbol{u}_{\mathrm{c}}(k_{\mathrm{d}})$ satisfies the bound constraints, and is defined in element-wise by:

$$\mathrm{sat}(u) = \begin{cases} u_{\max}, & \text{if } u > u_{\max} \\ u_{\min}, & \text{if } u < u_{\min} \\ u, & \text{otherwise,} \end{cases} \tag{4.3}$$

with $u_{\min}$ and $u_{\max}$ the minimal and maximal allowed values for the corresponding elements in the control inputs for the freeway network, and $\boldsymbol{u}_{\mathrm{b}}(k_{\mathrm{c}})$ the corresponding MPC input. Figure 4.2 illustrates the different time scales of MPC and DRL control sampling time, and how $\boldsymbol{u}_{\mathrm{rl}}$ modifies $\boldsymbol{u}_{\mathrm{b}}$.

4.3.2. DETAILED DESCRIPTION OF THE FRAMEWORK

The details of the MPC and DRL modules are provided in this section.

MPC MODULE

A standard MPC procedure is performed within the MPC module, where a nominal model F is used to describe the freeway dynamic. The system states $\boldsymbol{x}$ are updated every simulation step k_s. The simulation time steps that correspond to the MPC control step k_c are given by:

$$k_{\mathrm{s,c}}(k_\mathrm{c}) \in \{k_\mathrm{c}m, k_\mathrm{c}m+1, \ldots, k_\mathrm{c}m+m-1\}, \tag{4.4}$$

where $m = m_1 m_2$. Thus at simulation time steps $k_\mathrm{s} = k_\mathrm{c}m$, the real states of the freeway network are measured and are fed into the MPC module. The following optimization problem is solved at every control time step k_c:

$$\min_{\tilde{\boldsymbol{u}}_\mathrm{b}(k_\mathrm{c}), \tilde{\boldsymbol{x}}(k_\mathrm{c})} \sum_{\ell=1}^{N_\mathrm{p,s}} J(k_\mathrm{c}m+\ell) \tag{4.5}$$

$$\begin{aligned}
\text{s.t.}\quad & \hat{\boldsymbol{x}}(k_\mathrm{c}m+\ell+1) = F(\hat{\boldsymbol{x}}(k_\mathrm{c}m+\ell), \boldsymbol{u}_\mathrm{s}(k_\mathrm{c}m+\ell), \hat{\boldsymbol{d}}(k_\mathrm{c}m+\ell)), \quad \text{for } \ell = 0, \ldots, N_\mathrm{p,s}-1,\\
& \hat{\boldsymbol{x}}(k_\mathrm{c}m+\ell) \in \mathscr{X}, \quad \text{for } \ell = 1, \ldots, N_\mathrm{p,s}, && (4.6)\\
& \boldsymbol{u}_\mathrm{b}(k_\mathrm{c}+k) \in \mathscr{U}, \quad \text{for } k = 0, 1, \ldots, N_\mathrm{p,c}-1, && (4.7)\\
& \boldsymbol{u}_\mathrm{s}((k_\mathrm{c}+k)m+\ell)) = \boldsymbol{u}_\mathrm{b}(k_\mathrm{c}+k), \quad \text{for } \ell = 0, 1, \ldots, m-1, k = 0, 1, \ldots, N_\mathrm{p,c}-1, && (4.8)
\end{aligned}$$

where $J(k_\mathrm{s})$ represents the predicted objective function value (e.g., TTS) during the simulation interval $[k_\mathrm{s}T_\mathrm{s}, (k_\mathrm{s}+1)T_\mathrm{s})$, and $\tilde{\boldsymbol{u}}_\mathrm{b}(k_\mathrm{c}) = [\boldsymbol{u}_\mathrm{b}^\top(k_\mathrm{c}), \ldots, \boldsymbol{u}_\mathrm{b}^\top(k_\mathrm{c}+N_\mathrm{p,c}-1)]^\top$ denotes the variables to be optimized over the prediction window of length $N_\mathrm{p,c}$, with $\boldsymbol{u}_\mathrm{b}(k_\mathrm{c})$ the control inputs (e.g. ramp metering rates or variable speed limits) at control time step k_c. Moreover, $\boldsymbol{u}_\mathrm{s}(k_\mathrm{s})$ is the MPC control input at simulation time step k_s, and $\tilde{\boldsymbol{x}}(k_\mathrm{c}) = [\hat{\boldsymbol{x}}^\top(k_\mathrm{c}m+1), \ldots, \hat{\boldsymbol{x}}^\top(k_\mathrm{c}m+N_\mathrm{p,s})]^\top$ with $\hat{\boldsymbol{x}}(k_\mathrm{s})$ denoting the predicted future state at simulation time step k_s. Besides, $\hat{\boldsymbol{d}}(k_\mathrm{s})$ contains the estimated traffic demands at simulation time step k_s. In addition, $N_\mathrm{p,s}$ and $N_\mathrm{p,c}$ are the prediction horizon length counted in terms of the simulation time steps and MPC time steps, respectively, in which $N_\mathrm{p,s} = N_\mathrm{p,c}m$. Equations (4.6) and (4.7) represent the constraints on the states and the inputs, respectively, where $\mathscr{X}$ and $\mathscr{U}$ represent the feasible sets for the states and the inputs. Due to the nonlinearity and non-smoothness of the traffic model, the resulting optimization problem is, in general, nonlinear and non-convex. Therefore, a nonlinear optimization solver, such as multi-start sequential quadratic programming (SQP), simulated annealing, or genetic algorithms [51] is required. After the above optimization problem is solved, the first element of the optimized control input $\tilde{\boldsymbol{u}}_\mathrm{b}(k_\mathrm{c})$ is given to the DRL module.

DRL MODULE

Consider the freeway network as a Markov Decision Process (MDP), which is a discrete-time stochastic control process. It can be represented by a five-tuple $\langle S, A, P, \mathscr{R}, \gamma \rangle$. The state space S, action space A, and reward distribution $\mathscr{R}$ are defined in this section. Furthermore, P denotes the transition probability among the states, and is implicitly defined by the freeway network model. Moreover, $\gamma \in [0,1)$ is the user-defined discount factor on the future rewards. The DRL module operates at the low level with a higher frequency than the MPC module. The control sampling time T_d of the DRL module is larger than the simulation sampling time T_s, in order to avoid a too frequent change in the control inputs to the freeway network. Therefore, the simulation time steps that correspond to the DRL control time step k_d includes:

$$k_{\mathrm{s,rl}}(k_\mathrm{d}) \in \{k_\mathrm{d} m_2, k_\mathrm{d} m_2 + 1, \ldots, k_\mathrm{d} m_2 + m_2 - 1\}, \tag{4.9}$$

where m_2 is defined in (4.1). The state, action, and reward of DRL are updated every control step k_d, and are defined as it follows.

State $\boldsymbol{s}(k_\mathrm{d}) \in S$: The state space of DRL should consist of all necessary information of the framework. Since deep neural networks are employed in DRL, the states fed into the input layer are normalized to the same order of magnitude in order to facilitate the learning. Thus, we have:

$$\boldsymbol{s}(k_\mathrm{d}) = [\bar{\boldsymbol{x}}^\top(k_\mathrm{d} m_2), \bar{\boldsymbol{u}}_\mathrm{s}^\top(k_\mathrm{d} m_2), \bar{\boldsymbol{d}}^\top(k_\mathrm{d} m_2), \bar{\boldsymbol{u}}_\mathrm{c}^\top(k_\mathrm{d} - 1)]^\top, \tag{4.10}$$

where $\bar{\boldsymbol{x}}(k_\mathrm{d} m_2)$, $\bar{\boldsymbol{u}}_\mathrm{s}(k_\mathrm{d} m_2)$, $\bar{\boldsymbol{d}}(k_\mathrm{d} m_2)$, and $\bar{\boldsymbol{u}}_\mathrm{c}(k_\mathrm{d} - 1)$ are, respectively, the normalized states of the freeway network, the MPC inputs, the real demands at simulation time step $k_\mathrm{d} m_2$, and the overall control input of the framework at previous control time step $k_\mathrm{d} - 1$. The state space S of DRL is the Cartesian product of the MPC state set $\mathscr{X}$, MPC input set $\mathscr{U}$, the traffic demand set that can be defined by $\mathscr{D}$, and $\mathscr{X}$, i.e., $S = \mathscr{X} \times \mathscr{U} \times \mathscr{D} \times \mathscr{X}$.

Action $\boldsymbol{u}_\mathrm{rl}(k_\mathrm{d}) \in A$: The action $\boldsymbol{u}_\mathrm{rl}$ is used to modify the MPC input $\boldsymbol{u}_\mathrm{b}$. Therefore, they have the same dimension, i.e., $\dim \boldsymbol{u}_\mathrm{rl} = \dim \boldsymbol{u}_\mathrm{b}$. For simplicity, it is assumed that the action space is also continuous.

Note that action $\boldsymbol{u}_\mathrm{rl}$ is generated from the output layer of the DNNs, and the original values of its elements are between $[-1, 1]$. Thus these values are scaled back to the real control inputs before they are added. Moreover, the elements of $\boldsymbol{u}_\mathrm{rl}$ have a smaller magnitude than those of $\boldsymbol{u}_\mathrm{b}$, such that $\boldsymbol{u}_\mathrm{b}$ dominates the control input in this framework and provides basic performance, while $\boldsymbol{u}_\mathrm{rl}$ is an ancillary control input that acts at a higher frequency and aims at improving the performance. Furthermore, $\boldsymbol{u}_\mathrm{rl}$ meets the following inequalities that defines the action space A:

$$-w_u \Delta \boldsymbol{U} \leq \boldsymbol{u}_\mathrm{rl} \leq w_u \Delta \boldsymbol{U}, \tag{4.11}$$

where $\Delta \boldsymbol{U} = \boldsymbol{u}_\mathrm{max} - \boldsymbol{u}_\mathrm{min}$, with $\boldsymbol{u}_\mathrm{max}$ the upper bound and $\boldsymbol{u}_\mathrm{min}$ the lower bound of $\mathscr{U}$, and $w_u \in [0, 1)$ is the scaling parameter that determines to what extent $\boldsymbol{u}_\mathrm{rl}$ influences $\boldsymbol{u}_\mathrm{b}$.

Reward $r(\boldsymbol{s}(k_\mathrm{d}), \boldsymbol{u}_\mathrm{rl}(k_\mathrm{d})) \in \mathscr{R}$: In order to coordinate MPC and DRL to achieve the optimal performance, the reward function should include the objective function J of the MPC module:

$$r(\boldsymbol{s}(k_\mathrm{d}), \boldsymbol{u}_\mathrm{rl}(k_\mathrm{d})) = \sum_{k=1}^{m_2} \big(-J(k_\mathrm{d} m_2 + k) - w_p P_s(k_\mathrm{d} m_2 + k) \big), \tag{4.12}$$

where P_s denotes the state constraint violation, and $w_p > 0$ is the penalty weight parameter. Let $r_\mathrm{t}(k_\mathrm{d})$ be an equivalent representation of $r(\boldsymbol{s}(k_\mathrm{d}), \boldsymbol{u}_\mathrm{rl}(k_\mathrm{d}))$, which denotes the observed reward based on the traffic condition during DRL control step k_d. In order to evaluate J, the relevant state $\boldsymbol{x}(k_\mathrm{s})$ can be measured at every simulation time step, and the MPC input $\boldsymbol{u}_\mathrm{s}(k_\mathrm{s})$ can be obtained from (4.8). Moreover, $P_s(k_\mathrm{s})$ can be calculated directly based on state $\boldsymbol{x}(k_\mathrm{s})$, for $k_\mathrm{s} = k_\mathrm{d}m_2 + 1, \ldots, k_\mathrm{d}m_2 + m_2$. The reward is a negative value, and thus $\mathscr{R}$ is the set of negative numbers.

The deep actor-critic algorithms are considered to train the framework, among which Deep Deterministic Policy Gradient (DDPG) [111] is chosen for the DRL agent, which is an off-policy and model-free algorithm that can deal with continuous state and action spaces, and has been implemented successfully in many freeway traffic studies (see, e.g., [61], [191], [201]).

Remark 7. *The standard MPC procedure within the high-level MPC module can be replaced with any efficient MPC variations, such as parameterized MPC or DMPC for large-scale freeway networks. The DDPG agent can easily be extended to arbitrary off-policy DRL algorithms that can deal with continuous state and action spaces.*

4.3.3. ALGORITHM FOR TRAINING THE FRAMEWORK

The goal of learning is to train a policy π, such that the expected return at state $\boldsymbol{s}(k_\mathrm{d})$ after taking action $\pi(\boldsymbol{s}(k_\mathrm{d}))$ is maximized. The expected return for action $\pi(\boldsymbol{s}(k_\mathrm{d}))$ taken at state $\boldsymbol{s}(k_\mathrm{d})$ is given by:

$$\begin{aligned} Q^\pi\left(\boldsymbol{s}(k_\mathrm{d}), \pi(\boldsymbol{s}(k_\mathrm{d}))\right) =& \mathbb{E}_{r,\boldsymbol{s}\sim E}\left[\sum_{k=0}^{\infty} \gamma^k r(\boldsymbol{s}(k_\mathrm{d}+k), \boldsymbol{u}_\mathrm{rl}(k_\mathrm{d}+k))\right] \\ =& \mathbb{E}_{r,\boldsymbol{s}\sim E}\left[r(\boldsymbol{s}(k_\mathrm{d}), \boldsymbol{u}_\mathrm{rl}(k_\mathrm{d})) + \gamma Q^\pi\left(\boldsymbol{s}(k_\mathrm{d}+1), \pi(\boldsymbol{s}(k_\mathrm{d}+1))\right)\right], \end{aligned} \tag{4.13}$$

where the subscript $r, \boldsymbol{s} \sim E$ implies that the transitions among the states in the environment are stochastic. Note that the return depends on the chosen actions, and thereafter on the policy π. In DDPG, both the return and the policy π are approximated by deep neural networks, which are notated as $Q^\pi_\phi(\boldsymbol{s}(k_\mathrm{d}), \boldsymbol{u}_\mathrm{rl}(k_\mathrm{d}))$ and π_θ, respectively. They are also known as the critic and the actor, and ϕ and θ represent the parameters of the corresponding neural networks, as shown in Figure 4.1. In addition, the target network technique is used, which introduces two target critic and actor deep neural networks, corresponding to ϕ' and θ', that are updated in a slower pace in order to improve the stability of the training process. Experience replay is also utilized to remove correlations in the observation sequence, and to provide better learning convergence, and a replay buffer R is used to store the agent's experience. For more details, the readers are referred to [111].

Instead of the traditional one-step temporal-difference (TD) target in DDPG, we use the n-step TD method [180], i.e.:

$$y_{k_\mathrm{d}} = r_n(k_\mathrm{d}) + \gamma^n Q^\pi_{\phi'}(\boldsymbol{s}(k_\mathrm{d}+n), \boldsymbol{u}'_\mathrm{rl}(k_\mathrm{d}+n)), \tag{4.14}$$

in which $\boldsymbol{u}'_\mathrm{rl}(k_\mathrm{d}+n) = \pi_{\theta'}(\boldsymbol{s}(k_\mathrm{d}+n))$, and the n-step reward is given by:

$$r_n(k_\mathrm{d}) = \sum_{k=0}^{n-1} \gamma^k r(\boldsymbol{s}(k_\mathrm{d}+k), \boldsymbol{u}_\mathrm{rl}(k_\mathrm{d}+k)), \tag{4.15}$$

where $\boldsymbol{u}_{\mathrm{rl}}(k_{\mathrm{d}}) = \pi_\theta(\boldsymbol{s}(k_{\mathrm{d}}))$. Then the loss function for the critic is given by:

$$L(\phi) = \frac{1}{N}\sum_i \Big(y_i - Q_\phi^\pi(\boldsymbol{s}(i), \boldsymbol{u}_{\mathrm{rl}}(i))\Big), \tag{4.16}$$

where i is the index of the data points of a mini-batch of size N that is sampled randomly from the experience replay buffer. The critic parameters ϕ are therefore updated by minimizing the loss function with the Adaptive Moment Estimation (Adam) optimizer [97].

The benefits of using n-step TD here are fourfold:

1. A freeway network is a large-scale system with time delays, which means the control measures only take effect after a period of time. Thus, looking n steps to the future can better evaluate the quality of the actions taken.
2. The optimization of the DDPG agent considers the reward for n future steps, which coincides with the predicted objective function in MPC. In practice, taking $n = N_{\mathrm{p,s}}/m_2$ makes the look-ahead time of DDPG and MPC the same, and thus these two modules cooperate better.
3. Looking n steps ahead makes the learning process more efficient than the one-step TD method of the conventional DDPG algorithm, where the update is only based on bootstrapping from the value of the state one step later [180].
4. By introducing future rewards in (4.14), there is no need to predict future demand information as the MPC module does. Therefore, the state space definition (4.10) is simpler and has a smaller dimension.

After the optimization of the critic network, the actor is subsequently updated by maximizing the return Q_ϕ^π based on the policy gradient. More details can be found in [111].

One advantage of DDPG as an off-policy algorithm is that its exploration policy is independent from the learning process, which means that a stochastic exploration is allowed. In this context, the Ornstein-Uhlenbeck model [186] is used to produce the noise $w_n(k_{\mathrm{d}})$ for exploration (i.e., added on the DRL actions), where the magnitude decays with time step k_d. The overall learning algorithm of the MPC-DRL framework is summarized in Algorithm 1. Note that the entire simulation time is supposed to be T.

Algorithm 1 Hierarchical MPC-DRL Framework Algorithm for Freeway Traffic Control

```
Initialize critic and actor networks $Q_\phi^\pi$ and $\pi_\theta$ with parameters $\phi$ and $\theta$
Initialize target network $Q_{\phi'}$ and $\pi_{\theta'}$ with parameters $\phi'$ and $\theta'$
Initialize experience replay buffer $R$
for episode from 1 to $M$ do
    Initialize the empty traffic network with initial traffic demands for $T_{\text{ini}}$ time units
    for $k_{\text{c}} = 0$ to $\frac{T}{T_{\text{c}}} - 1$ (MPC outer loop) do
        Observe current traffic state $\boldsymbol{x}(k_{\text{c}}m)$ from the environment, and estimate the traffic demands $\hat{\boldsymbol{d}}(k_{\text{c}}m + k), k = 0, 1, \dots, N_{\text{p,s}}$
        Perform high-level MPC with freeway model $F$ and prediction horizon $N_{\text{p,s}}$, by solving optimization problem (4.5)-(4.7)
        Pass the optimized control input $\boldsymbol{u}_{\text{b}}(k_{\text{c}})$ to the low-level DRL module
        for $k_{\text{d}} = k_{\text{c}}m_1$ to $(k_{\text{c}} + 1)m_1 - 1$ (DRL inner loop) do
            Receive state $\boldsymbol{s}(k_{\text{d}})$ according to (4.10)
            Select action $\boldsymbol{u}_{\text{rl}}(k_{\text{d}}) = \pi_\theta(\boldsymbol{s}(k_{\text{d}})) + w_n(k_{\text{d}}))$
            Combine the control input of MPC and RL with a saturation function using (4.2)
            for $k_{\text{s}} = k_{\text{d}}m_2, \dots, k_{\text{d}}m_2 + m_2 - 1$ do
                Execute action $\boldsymbol{u}_{\text{c}}(k_{\text{d}})$ in the freeway network, with the real traffic demand $\boldsymbol{d}(k_{\text{s}})$
            end for
            Observe reward $r_{\text{t}}(k_{\text{d}})$ and new state $\boldsymbol{s}(k_{\text{d}} + 1)$
            Store transition $(\boldsymbol{s}(k_{\text{d}}), \boldsymbol{u}_{\text{rl}}(k_{\text{d}}), r_{\text{t}}(k_{\text{d}}), \boldsymbol{s}(k_{\text{d}} + 1))$ in $R$ (the transitions are stored in order)
            Sample a mini-batch of $N$ transitions from $R$ randomly, each of which contains $n$ steps: $\boldsymbol{s}(i), \boldsymbol{u}_{\text{rl}}(i), r_{\text{t}}(i), \boldsymbol{s}(i+1), \dots, \boldsymbol{s}(i+n-1), \boldsymbol{u}_{\text{rl}}(i+n-1), r_{\text{t}}(i+n-1), \boldsymbol{s}(i+n)$
            Update the critic network $Q_\phi^\pi$ by minimizing the loss function $L(\phi)$ with (4.14)-(4.16)
            Update the actor network $\pi_\theta$ by the sampled policy gradient [111]
            Update the target networks:
                $\theta' \leftarrow \tau\theta + (1-\tau)\theta', \phi' \leftarrow \tau\phi + (1-\tau)\phi'$
        end for
    end for
end for
```

4.4. CASE STUDY

The proposed MPC-DRL framework is now implemented and evaluated via a benchmark freeway network from [73]. METANET is adopted to model this network, for which the readers are referred to [73], [103]. Model uncertainties and external disturbances are introduced into the model to represent the real-world system, as illustrated in Section 4.4.1. Furthermore, the proposed MPC-DRL framework is compared with standalone MPC and DRL methods. In this case study, the performance criteria consist of TTS of all

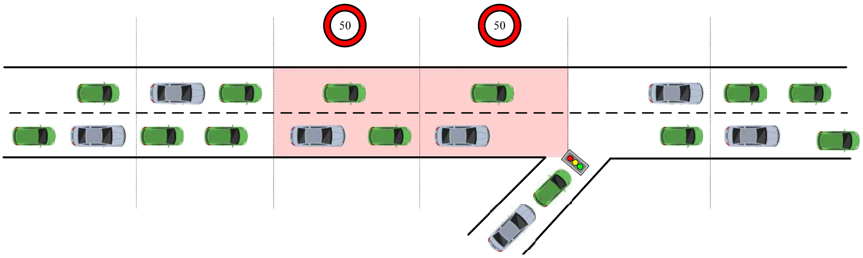

Figure 4.3: The benchmark freeway network with one metered on-ramp and two segments with speed limits (marked in red) used for the case study.

Table 4.4: Real and estimated values for freeway network parameters in the case study

	T [s]	$\rho_{\max}$ [veh/km/lane]	κ [veh/km/lane]	η [km^2/h]	a_m	σ
Real	10	180	40	60	1.867	0.0122
Estimates	10	150	48	50	2.160	0.01
	v_{free} [km/h]	ρ_{crit} [veh/km/lane]	α	τ [s]	L_m [m]	
Real	102	33.5	0.1	18	1000	
Estimates	102	37.5	0.08	14.5	800	

the vehicles for the entire traffic network and constraint violations of the queues on the lanes. All the simulations were conducted in Matlab version 2022a running on a PC with an Intel Xeon Quad-Core E5-1620 V3 CPU with a clock speed of 3.5 GHz.

4.4.1. SETUP

FREEWAY TRAFFIC NETWORK

A benchmark network is taken from [73]. Note that this benchmark network has also been used in other freeway traffic studies [25], [46], [52], [170]. As shown in Figure 4.3, the network consists of two origins (i.e., one main-stream and one on-ramp) and one destination. The length of the main stretch is 6 km, which is divided into 2 links. The first link includes the 4 segments before the on-ramp, and the second link consists of the 2 segments after the on-ramp. The mainstream has two lanes with a capacity of 2000 veh/h each, and its maximal allowed queue length is 200 veh. The on-ramp has one lane with the capacity of 2000 veh/h, and the maximum on-ramp queue length is 100 veh. The network parameters are taken from [73] and the same mathematical notations are used here. The real parameters are assumed unknown in this case study, and the estimated values for these parameters are used in the prediction model. Both the real and the estimated parameter values are given in Table 4.4.

DEMAND SCENARIO

A typical demand scenario as shown in Figure 4.4 similar to [73] is considered in order to evaluate the controllers,. This demand scenario can cause severe traffic congestion without control, and is suitable to examine the control effectiveness of both ramp metering and variable speed limits in this freeway network. To reproduce the stochastic phenomena of the traffic network, random noises with Gaussian distribution are added

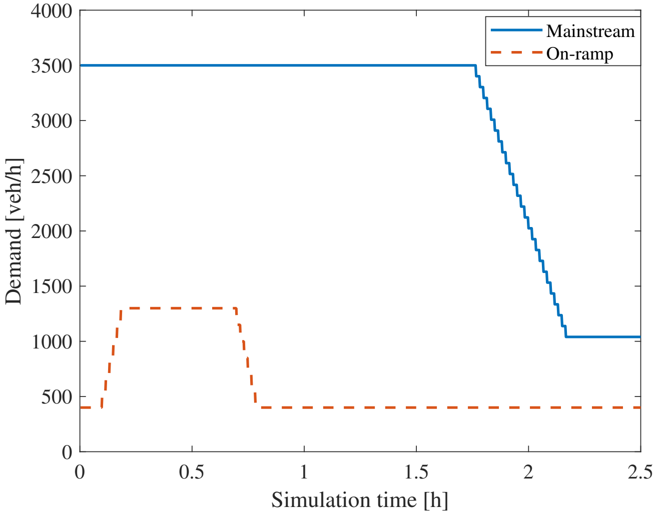

Figure 4.4: Predicted traffic demands used in the case study.

to the demands of both main-stream and on-ramp, for which the mean values are 0 and the standard deviations are 5% of the corresponding maximal demand values. The freeway network is initially empty, and is next simulated with a constant demand at 3000 veh/h for the main-stream and 500 veh/h for the on-ramp for a period of 10 min, before the control simulations start.

4.4.2. CONTROLLERS

In this case study, the following controllers are implemented and compared: standalone MPC, standalone DRL (with n-step TD), and combined MPC-DRL framework (with n-step TD). The simulation parameters are $T_\mathrm{s} = 10$ s, $T_\mathrm{d} = 60$ s, $T_\mathrm{c} = 300$ s, $T_\mathrm{ini} = 600$ s, $T = 9000$ s. These parameters apply for all the controllers. The ramp metering rate ranges from 0 to 1, and the variable speed limits range from 20 km/h to v_free, which is 102 km/h. Both control actions are continuous.

STANDALONE MPC CONTROLLER

The objective function (4.5) used in the MPC controller is written as

$$J(k_\mathrm{s}) = w_\mathrm{TTS} J_\mathrm{TTS}(k_\mathrm{s}) + w_\mathscr{D} \|\boldsymbol{u}_\mathrm{s}(k_\mathrm{s}) - \boldsymbol{u}_\mathrm{s}(k_\mathrm{s}-1)\|_2^2,$$

where $J_\mathrm{TTS}(k_\mathrm{s})$ denotes the TTS value predicted for simulation time interval $[k_\mathrm{s} T_\mathrm{s}, (k_\mathrm{s}+1) T_\mathrm{s})$, the second term imposes a penalty on the fluctuations between consecutive control inputs, and the weights are selected as $w_\mathrm{TTS} = 1$, $w_\mathscr{D} = 0.4$. The prediction window is 10 min, which means $N_\mathrm{p,c} = 2$, $N_\mathrm{p,s} = 60$. The prediction model used by the MPC controller is the METANET model with the estimated parameter values in Table 4.4, and the predicted demands used by MPC are shown in Figure 4.4. The control inputs include one

Table 4.5: Parameters used for DRL agent training

Parameter	Value
Maximal episodes M	4000
Mini-batch size N	512
Experience replay buffer R size	$2 \cdot 10^5$
Discounter factor γ	0.99
Learning rate (both actor and critic networks)	0.001
Target network update rate τ	0.01
Noise w_n standard deviation	0.3
Noise w_n decay rate	$5 \cdot 10^{-6}$

ramp metering rate and two variable speed limits, which are constrained by the lower and upper bounds given before.

For simplicity, the control problem is transcribed into an optimization problem by single-shooting [34]. The resulting optimization problem is nonlinear and non-convex, so the Matlab `fmincon` function with the SQP algorithm are used to solve the optimization problem . In order to avoid getting stuck in local optima, multiple starting points are used to solve the optimization problem (i.e., 30 for this case study[2]). In order to achieve a trade-off between the computational accuracy and efficiency, the `fmincon` stopping criteria for the cost function tolerance, step tolerance, and constraint tolerance are selected to be 10^{-2}.

STANDALONE DRL (WITH n-STEP TD)

The standalone DRL agent in this case study shares the same definition as the DRL module in Section 4.3.2. Therefore, the dimensions of the state space and action space are 30 and 3, respectively. The actor network contains one input layer of size 30, one output layer of size 3, and two inner layers with 256 neurons for both layers. Accordingly, the critic network has two input layers, in which one layer that corresponds to the states is of size of 30 and is followed by a layer with 256 neurons, and the other input layer that corresponds to the actions is of size 3 and is followed by a layer with 128 neurons. Both input layers are connected to two consecutive inner layers with 256 and 128 neurons, respectively. The size of the output layer, which generates the Q-values of the state-action pairs, is 1. ReLU activation functions are used in all the neural networks. Moreover, the reward function consists of the objective function defined for the MPC controller and a penalty for constraint violation with weight $w_p = 10$. The actions of the standalone DRL are the same as the MPC controller. The other DRL parameters that are tuned during the learning process are given in Table 4.5. These parameters apply for both conventional standalone DRL and n-step TD DRL.

COMBINED MPC-DRL FRAMEWORK (WITH n-STEP TD)

The combined MPC-DRL framework (with n-step TD) consists of an MPC module that is the same as the standalone MPC controller, and a DRL module that is similar to the standalone DRL (with n-step TD). The action space of the MPC-DRL framework is different from the standalone DRL, due to the scaling parameter $w_u = 0.4$, which means that

---

[2] This number is obtained via several experiments, such that it achieves a good trade-off between the computational efficiency and optimality.

according to (4.11) the action bounds of the DRL module within the MPC-DRL framework are ± 0.32 for variable speed limits and ± 0.4 for ramp metering rate. The training parameters in Table 4.5 also apply for the combined MPC-DRL framework, except for the noise w_n, which has a smaller standard deviation of 0.2 and a decay rate of 2×10^{-5}. Furthermore, $n = 10$ is chosen for the DRL module, which makes the look-ahead time of the DRL module the same as the prediction horizon of the MPC controller.

4.4.3. RESULTS FOR THE LEARNING PROCESS

The standalone DRL (with 10-step TD) and the combined MPC-DRL framework (with 10-step TD) are both trained 10 times independently over the stochastic environment (i.e., the benchmark freeway network with stochastic demands), with 4000 episodes for each run. Each episode contains a simulation interval of 9000 s with the mentioned stochastic demands. In the plots, the episode rewards have first been smoothed by a moving average filter of size 21 to better present the learning progress. The learning performance is presented in Figure 4.5, in which the solid lines represent the mean episode reward values of the 10 runs, and the shaded areas show the 95% confidence intervals for each method.

Figure 4.5 shows that the learning curves of the combined MPC-DRL framework methods start with higher rewards than the standalone DRL methods. In addition, the combined MPC-DRL framework methods converge faster, which indicates that the proposed framework has a better sample efficiency than the conventional DRL methods. The reason is that the MPC module within the MPC-DRL framework generates basic control inputs that provide baseline control performance, and the DRL module within the framework only requires a limited exploration space with smaller action bounds and thus requires less sample data. Furthermore, the methods with 10-step TD have better learning performance than the ones without 10-step TD, which validates the advantages of the n-step TD method for large-scale freeway networks. In particular, the framework with 10-step TD has a more stable learning curve than the one without 10-step TD. This implies that a DRL module with a similar prediction window to the MPC module can cooperate better within the combined MPC-DRL framework.

4.4.4. RESULTS FOR THE IMPLEMENTATIONS

According to the learning performance, only the trained standalone DRL with 10-step TD controller and combined MPC-DRL framework with 10-step TD controller are implemented on the benchmark freeway network, and their control performance are evaluated in terms of TTS and constraint violations. The standalone MPC controller and the no-control case (i.e., no ramp metering or speed limit) are also included for comparison. Due to the stochastic feature of the network, the experiments for each controller are repeated 20 times independently with random demands in order to evaluate the control performance. Figure 4.6 shows the total number of vehicles in the traffic network during the simulation interval, which indicates the congestion degree of the traffic. In the figure, the solid lines denote the mean values of the 20 runs, and the shade areas indicate the 95% confidence intervals. It is shown that the combined MPC-DRL framework achieves the best control performance in terms of reducing the traffic congestion among all the controllers. In contrast to standalone MPC, the framework has learned from interacting

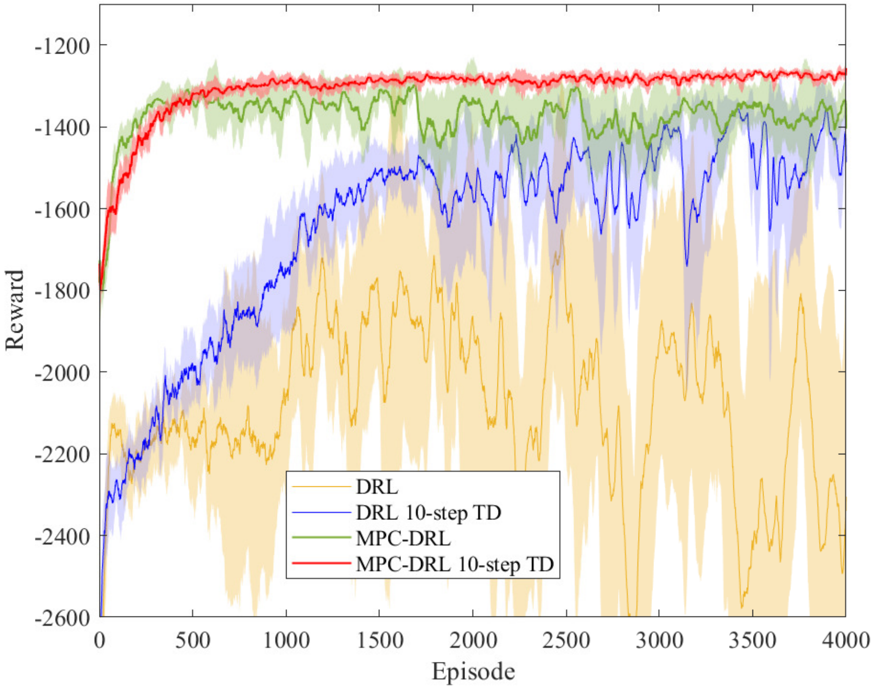

Figure 4.5: Learning performance of standalone DRL and combined MPC-DRL with and without 10-step TD.

with the environment, and the DRL module within the framework can compensate for the model uncertainties and external disturbances and therefore, provides extra optimality. Accordingly, the distribution of the TTS values for different controllers over those 20 runs are depicted in Figure 4.7 using a box plot. This plot confirms that the combined framework outperforms the other controllers in terms of TTS. The variance of the TTS values for standalone DRL is the largest, and its TTS value is not necessarily better than the no control case, which implies that the performance of standalone DRL is not always reliable.

Furthermore, the queue constraint violations during the simulation interval for different controllers are shown in Figure 4.8. This figure shows that all the controllers can reduce the constraint violation compared to the no control case. However, standalone MPC still violates the constraint due to the model uncertainties. Standalone DRL also violates the constraints. This is because standalone DRL has a larger action space, and thus it takes more training data for exploration (i.e., low sample efficiency). The DRL agent is not sufficiently trained at 4000 episode in this case study, and the fluctuation of its learning curve is still significant (see Figure 4.5). Therefore, the DRL agent is still exploring and would take actions that are worse than the no-control case, which explains the TTS performance of DRL. In contrast, the combined MPC-DRL framework with 10-

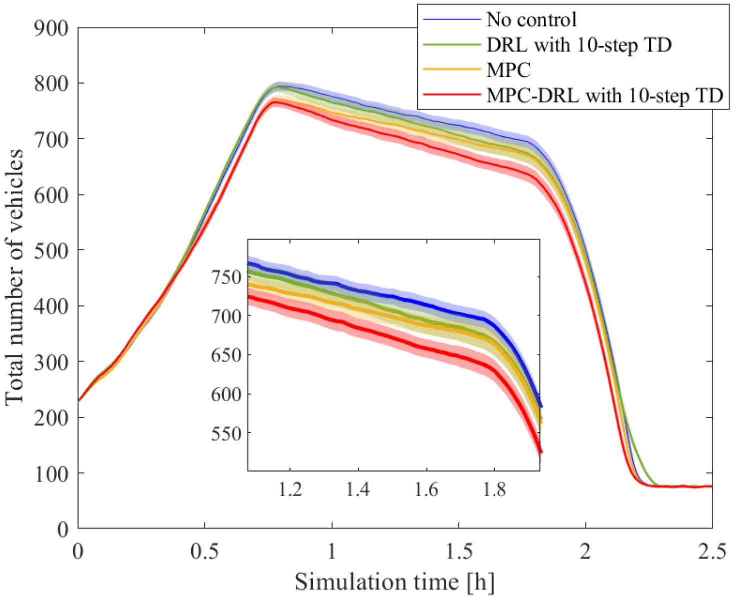

Figure 4.6: The comparison of the total number of vehicles in the traffic network during the simulation interval with different controllers.

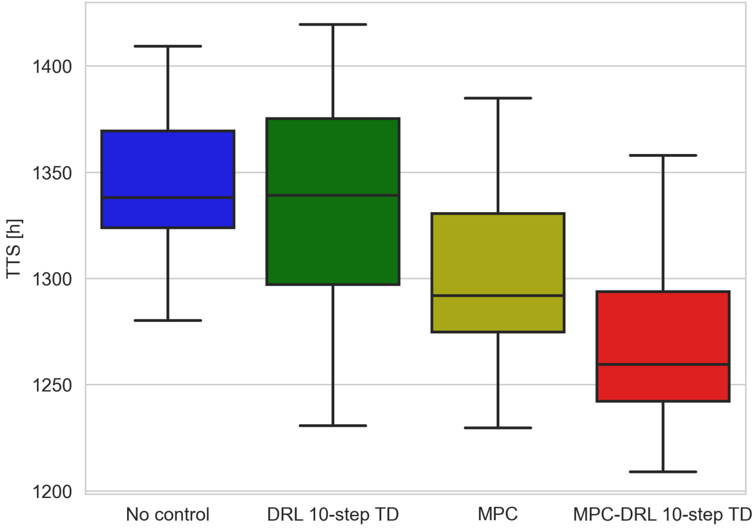

Figure 4.7: Box plot of TTS for different controllers.

step TD can avoid the constraint violation for every independent run. In addition to the smaller action space and high sample efficiency of the combine MPC-DRL framework,

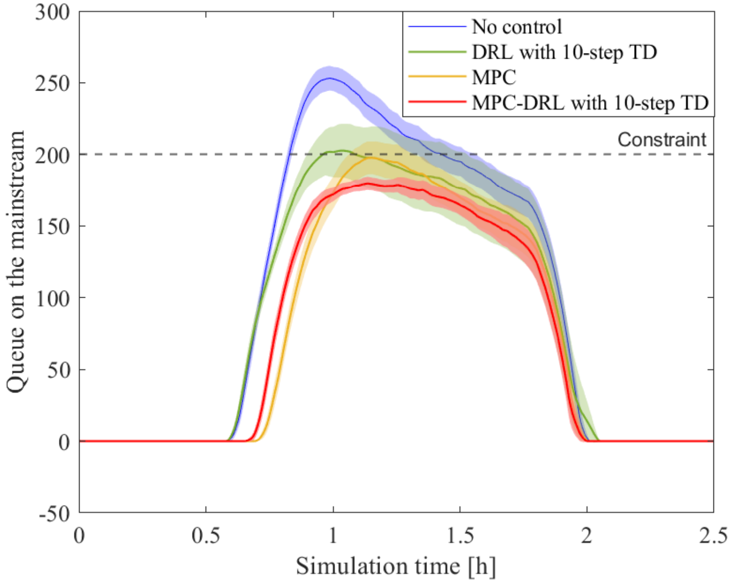

Figure 4.8: Mainstream queue constraint violations for different controllers.

the penalty on the constraint violations within the reward function of the DRL module, which coincides with the state constraints within the optimization problem of the MPC module, also contributes to avoiding the constraint violations. So in the proposed MPC-DRL framework, the MPC and the DRL modules complement each other during both the learning process and the implementation stage, which results in a better sample efficiency and control performance in terms of both TTS and constraint violation.

In order to present the traffic states during the simulation, we fix the demand noise and run the simulation for each compared controller. The traffic situation includes traffic flow speed, density, velocity on all the segments, and the control inputs for the on-ramp metering and variable speed limits over the entire simulation length. Simulation results are presented in Figure 4.9. For the no control case, the increased traffic demand from the on-ramp causes traffic congestion on segment 5 first, and the congestion area propagates to segment 4, 3, and 2 gradually. Eventually, the queue length at the mainstream grows significantly, and approximates 300. The TTS for this case is 1419.6 veh·h. For the standalone MPC case, the traffic congestion still exists but is postponed, in order to reduce the queue length at the mainstream. This is achieved by decreasing the ramp metering rate, and the queue length at the on-ramp increases accordingly. In addition, the overall outflow of the segments is larger than the no control case. However, constraint violation still exists. The TTS for this case is 1366.9 veh·h. Simulation results of the standalone DRL with 10-step TD controller show that the queue length at the mainstream is further reduced compared with the standalone MPC controller. However, the TTS for this case is 1403.7 veh·h, which is not improved compared with the no control case. For the combined MPC-DRL with 10-step TD controller, it can be observed that the queue length at the mainstream is limited under 200, and the traffic efficiency is also improved as the TTS for this case is 1315.3 veh·h. The main issue of the controller is that the fluctuation of the control inputs (i.e., ramp metering rate and variable speed limits) still exists, which might be undesired in practice. However, the fluctuation is already significantly reduced compared with the standalone DRL controller.

4.5. CONCLUSIONS

This chapter has developed a novel framework combining MPC and DRL for freeway traffic control. Since MPC and DRL each suffer from their own shortcomings and their characteristics complement each other well, it is beneficial to merge these two methods. The proposed MPC-DRL framework inherits the ability of DRL in learning from the environment to deal with uncertainties, and the ability of MPC in using the model information to provide basic performance. Specifically, the novel framework has a hierarchical structure, in which an MPC controller works at a high level with a lower frequency, while the DRL agent operates at a low level with a high frequency. An additional advantage of the proposed framework is that it requires less computational efforts compared to conventional MPC, due to the lower control frequency of the MPC module.

A simulation study has been conducted on a benchmark freeway network with model uncertainties and stochastic traffic demands. The proposed MPC-DRL framework (with n-step TD), standalone MPC, and DRL (with n-step TD) were trained and implemented to this traffic network. The results of the case study showed that the proposed MPC-DRL framework outperforms MPC and DRL in terms of both the learning process and the control performance, and the n-step TD method can improve the learning-based controllers for large-scale traffic networks. Moreover, the combined framework is easy to implement and can also potentially be applied to other systems that struggle with model uncertainties and high computational complexity, such as for the control of urban traffic networks or for the energy management of smart buildings. Future research will be conducted on extending the current framework to control very large traffic networks by combining distributed MPC and multi-agent DRL.

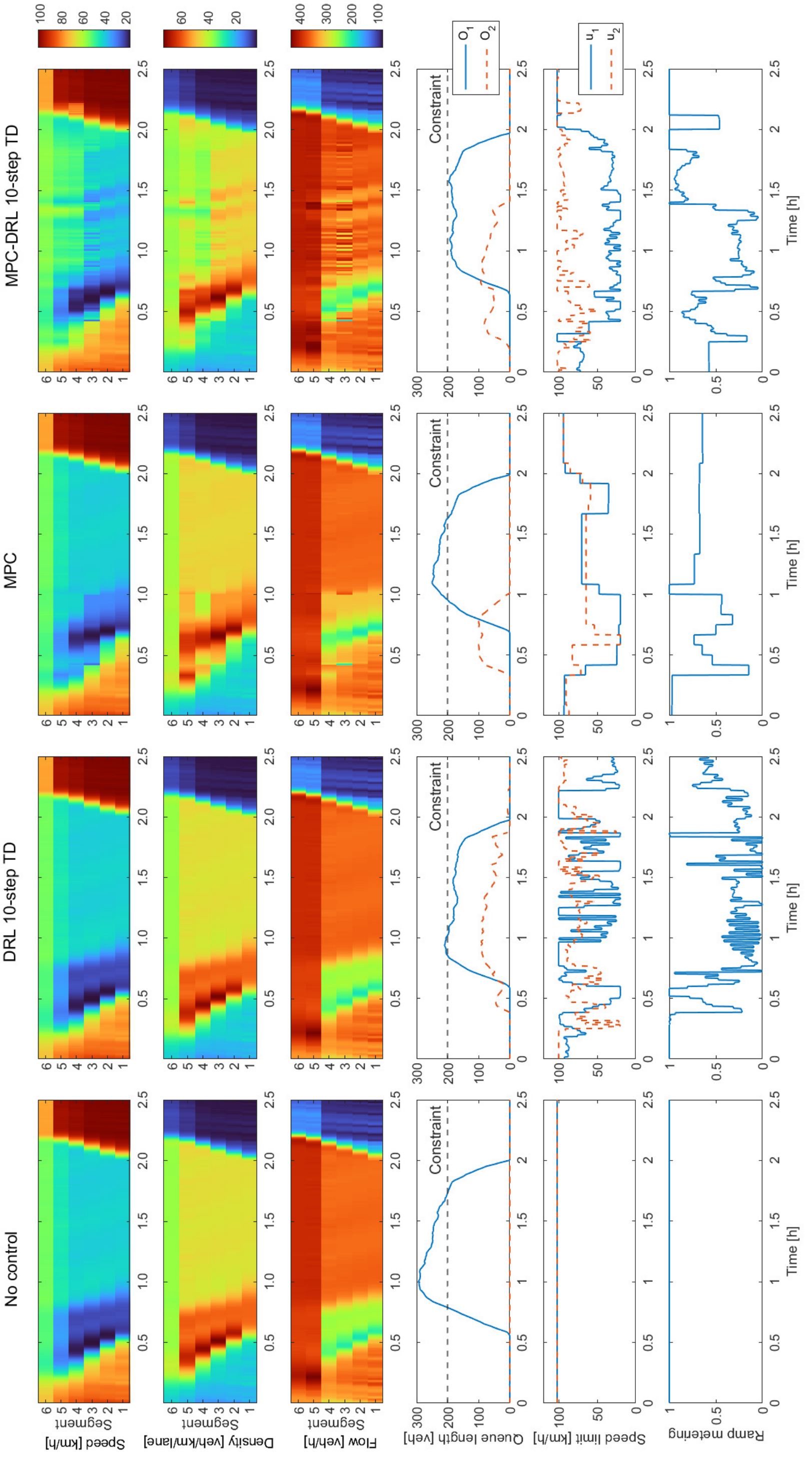

Figure 4.9: Traffic states during the simulation of all the compared controllers; O_1 represents the mainstream, O_2 represents the on-ramp, u_1 represents the variable speed limits on segment 1-3, and u_2 represent the variable speed limits on segment 1-4.

5

ADAPTIVE PARAMETERIZED MODEL PREDICTIVE CONTROL BASED ON REINFORCEMENT LEARNING: A SYNTHESIS FRAMEWORK

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

Parts of this chapter have been published in [177] and [179]

5.1. INTRODUCTION & MOTIVATION

Model predictive control (MPC) has been studied extensively within the last century, and mature theoretical results have been established for it [24], [136]. MPC operates in a receding-horizon style, where an optimization problem is solved at every control step to determine a sequence of control inputs based on a prediction of the future states using a prediction model. Only the first element of this control input sequence is implemented in practice, and after shifting the prediction horizon to the next control step, the entire procedure is repeated. In addition, since MPC can explicitly deal with state and input constraints and provide robust control performance [145], it has been widely used in engineering practice, such as in industrial processes, power systems, robotics, and management of transportation networks [73], [156]. However, for a large number of real-life systems where the dynamics are in general nonlinear and nonconvex, the optimization problem of MPC may become too complex that is not feasible for real-time implementations. Besides, a sufficiently accurate model is needed to ensure the control performance, which is not always available in practice. Therefore, the applications of MPC are often impacted by two main issues: high online computational complexity and model mismatches.

To address the first issue, a large number of studies have investigated computationally efficient MPC approaches, and have achieved satisfying results (see e.g., [6], [94]). One of the successful methods is parameterized MPC (PMPC) [60], [119], which simplifies the optimization problem of conventional MPC by reducing the number of the optimization variables. More specifically, PMPC introduces a parametric state-feedback function as the control law. This means that the PMPC input function does not vary across the prediction horizon, and only the parameters in the control law need to be optimized per control step. More details of PMPC and its applications will be given in Section 5.2.1. Nonetheless, although PMPC can reduce the online computational complexity, it still suffers from model mismatches that are caused by the changing environment and by unknown disturbances.

To address this issue, extensive research efforts have been made. Two representative directions are robust MPC and stochastic MPC [14], [129]. Robust MPC solves a robust optimal control problem at each control step within the MPC scheme by considering the worst-case scenarios for the external disturbances, which results in conservatism in the control performance. Stochastic MPC considers the probability distribution of the uncertainties to guarantee the chance-constraint satisfaction on the basis of conventional MPC. However, these approaches design the controllers by assuming certain knowledge of the uncertainties. In addition, these methods require even more computational power than conventional MPC. Another representative direction is learning-based or data-driven MPC methods [78], [108] where most of these methods focus on the identification of a system model, and are, thus, also known as adaptive MPC. More details about learning-based MPC will be given in Section 5.2.2. Another research direction that has recently drawn great interest is to combine MPC and reinforcement learning (RL) [180]. RL is a technique from the field of machine learning that learns how to take action in an uncertain environment so that to maximize an accumulative return. Deep reinforcement learning (DRL) incorporates deep learning techniques within the conventional RL schemes, allowing agents to deal with the problems large state space [134]. For simplicity, we use the term RL to refer to all the RL-related algorithms, including DRL. RL has recently shown great potential in many fields, including optimal control [8], [135]. Since RL is able to improve its control policy via interacting with the system and with its environment, it can naturally deal with unknown environments and disturbances. Despite these appealing features, RL still struggles with its own shortcomings, including constraint violations (i.e., safety issues) and low sample efficiency (i.e., a prolonged training process) [36]. Due to the very complementary characteristics of RL and MPC [59], many studies have developed various methods to exploit the advantages of both techniques. More details about relevant research that combine MPC with RL will also be presented in Section 5.2.

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

The remainder of this chapter is organized as follows. Section 5.2 introduces more details on related work. Section 5.3 presents the novel PMPC-RL synthesis framework and all the available strategies to adjust PMPC via RL techniques. Section 5.4 performs a case study to illustrate the proposed approach by the application to traffic management of a freeway network. Section 5.5 concludes the paper and proposes topics for future work.

5.2. RELATED WORK

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

5.2.1. PMPC

Lofberg [119] introduced a feedback loop in the control sequences for a robust MPC problem, by parameterizing the future control inputs in terms of the future states and several new parameters. This parameterization process reduces the number of the decision variables from the number of control inputs over the prediction horizon to the number of the introduced parameters. Note that conventional MPC can be regarded as a special case of PMPC, with an identity function as the parameterized control law. Goulart *et al.* [60] further extended this approach by parameterizing the control sequence as an affine function of the sequence of past disturbances. In this way, the disturbances can be accounted for by solving a convex optimization problem resulting from the parameterization. These studies all focus on the robust MPC problem for linear systems. However, most systems in practice are nonlinear.

Zegeye *et al.* [207] applied PMPC in freeway traffic management for the first time by parameterizing the ramp metering rates and variable speed limits, such that compared to solving the original nonlinear MPC problem the computation time was significantly reduced without much loss of performance. Van Kooten *et al.* [100] also employed the idea of parameterization to design a state-based adaptive controller for an urban traffic network. Pippia *et al.* [154] applied the PMPC method to the operation of microgrids. Jeschke and De Schutter [89] applied PMPC in signal control for urban traffic management, and achieved comparable control performance with substantially decreased computation time, compared to conventional MPC. However, the design of the parametric function that maps the states to control inputs is difficult and often requires expert knowledge. Jeschke *et al.* [90] addressed this issue by introducing grammatical evolution method to generate the parametric function automatically. They applied this method for traffic signal control of an urban network. The results show that the generated parametric state-feedback function even outperformed the handcrafted function.

5.2.2. LEARNING-BASED ADAPTIVE MPC

This section presents a brief overview of the literature on learning-based MPC, whose major purpose is to handle model uncertainties during the implementation of MPC. Hewing *et al.* [78] gave a very comprehensive review of learning-based MPC approaches, from which the reader can find more details. Conventional adaptive MPC refers to the studies that focus on system identification to compensate for the model uncertainties. In this paper, we broaden the scope of adaptive MPC to any MPC approach that can adapt to model uncertainties and disturbances.

ADAPTIVE MPC BY SYSTEM IDENTIFICATION

Lorenzen *et al.* [122] considered a constrained linear system with unknown but constant system parameters. A set-membership system identification method is used to estimate the set that contains the real parameter values, which results in a robust MPC problem. Then tube MPC techniques [157] are used to solve the problem, and to construct the terminal constraint and terminal set to guarantee stability and recursive feasibility. Heirung *et al.* [75] proposed an adaptive dual MPC approach for a single-input single-output linear time-invariant system for which the dynamic matrices are known and determined by orthogonal basis functions. A recursive least squares method is used to estimate the unknown parameters using observed data. The resulting optimization problem is subject to probabilistic output constraints, and is then reformulated as a quadratic programming problem that can be solved efficiently. Tanaskovic *et al.* [181] also employed a two-stage method for adaptive MPC of a linear time-varying multiple-input multiple-output system subject to model uncertainties and measurement noise. First a set-membership algorithm is used to estimate the parameter matrix. Then the obtained set is exploited in the MPC optimization problem to enforce constraints, which results in a robust finite-horizon optimal control problem. A similar method was used by [210] for adaptive linear MPC, where the main contribution is in adding extra variables to the optimization problem in order to adjust the shape and size of cross section of the tube. The obtained results are less conservative w.r.t. conventional adaptive MPC, while guaranteeing closed-loop stability and recursive feasibility.

The above studies are all about linear systems. There are also a few studies that focus on nonlinear adaptive MPC. Adetola *et al.* [2] proposed adaptive MPC for constrained nonlinear systems, where the model uncertainties are assumed to be static and can be expressed by unknown constant parameters. An uncertainty set is updated recursively to estimate the bounds of these parameters, which results in a robust MPC problem that is solved via both a Min-Max approach and a Lipschitz-based approach. Köhler *et al.* [98] presented a tube-based robust adaptive MPC for an uncertain nonlinear system subject to unknown constant model parameters and additive disturbances. Compared to ordinary robust MPC problems, the work improves the computational efficiency by modifying the tube formulations, while providing robust recursive feasibility and robust constraint satisfaction. Akpan and Hassapis [5] proposed to utilize neural networks to approximate the system model for MPC, since neural networks can approximate any nonlinear function with an arbitrary high accuracy [105]. The neural network is trained online based on a recursive least squares algorithm, and the resulting optimization problem with neural network-based model is solved using gradient-based methods.

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

ADAPTIVE MPC BY ADJUSTING THE CONTROLLER

Another direction of learning-based adaptive MPC is to adjust the controller design, such as by learning the cost function, constraint set, or terminal components. Marco *et al.* [127] considered the design of a linear quadratic regulator (LQR) for a linearized model. Instead of identifying the model, they directly tuned the introduced parametric cost function by iteratively evaluating the controller on the real system. This approach can also be integrated into an MPC scheme as in [9]. Piga *et al.* [153] adjusted the model parameters oriented towards the overall performance of the MPC controller, instead of minimizing the error between the model and the real system. In addition, the size of the prediction horizon is also added to the parameters to be adapted. Brunner *et al.* [19] worked on enlarging the terminal set of an MPC controller for linear systems by using the collected historical data. Rosolia and Borrelli [166] focused on iterative tasks with nonlinear MPC, in which the terminal cost and the terminal set are adjusted at every iteration in order to guarantee constraint satisfaction and system stability. Experiences from previous iterations are employed, and it is ensured that the cost does not increase from iteration to iteration. They further extended the result to a robust control context in [167].

5.2.3. RL-BASED ADAPTIVE MPC

RL-based adaptive MPC is another direction that utilizes RL techniques to obtain adaptive MPC. Due to the complementary features of MPC and RL, as mentioned in the previous section, combining MPC and RL is a promising research direction that has been drawing more and more attention recently. The very early study that employed RL in an MPC scheme is [138], in which the value function of RL was used to represent the infinite-horizon objective function value of MPC for Markov decision processes. The value function can be learned on-line during the implementation, and the MPC prediction horizon reduced to one step due to the approximation of the objective function. This idea opened up a research direction to combine MPC and RL, and has been adopted in many studies, such as [7], [209], [212]. Although this approach can alleviate the computational issue of conventional MPC by reducing the prediction horizon, it still suffers from model mismatches since the control inputs are optimized based on the prediction model. In addition, solving the optimization problem with a value function in the objective function is even more difficult due to the introduced extra nonlinearity. By connecting the objective function of MPC and the value function of RL, Gros and Zanon [64] parameterized the objective function of MPC and adapted it using RL. It is shown that the optimal policy can be obtained even based on an inaccurate model by modifying the objective function. However, it is not explained how to parameterize the objective function in a structured way, which is the core procedure for implementing this method.

The other main direction to combine MPC and RL is to merge their control inputs directly. Zhang *et al.* [211] integrated an RL agent in a model-reference scheme together with a conventional nonlinear controller. The RL agent is trained by performing repetitive tasks to compensate for the mismatches between the nominal model and the real system, and to eliminate the errors between the real states and the desired states. This idea is adopted by [161], which combines RL and MPC in a model reference framework and applies the resulting MPC-RL framework to traffic signal control for urban networks. [178] further extended the work by constructing a hierarchical framework in which the MPC and RL controllers work with different control frequencies and their control inputs are summed up. The resulting framework is applied to traffic management of freeway networks, and the results show that the combined MPC-RL controller can excellently deal with model mismatches. Another related study is [82]. They proposed a hierarchical structure for power distribution system restoration, in which the RL agents work at the lower level to make fast decisions on the active power dispatches, and a quadratic programming agent operates at the higher level using the local RL decisions to check the major grid constraints and to ensure system resilience. Based on the commands from the high-level controller, the RL agents revised their actions accordingly.

Although many studies have explored the combination of MPC and RL in various fields, so far there is not a comprehensive survey on this topic. The authors believe that the potentials of combining MPC and RL have not yet been fully developed. In addition, despite the fruitful results of adaptive MPC, relevant research on PMPC is quite limited. Therefore, a synthesis framework that utilizes RL to adjust PMPC is presented in this paper. It will be shown that not only the RL-MPC methods in Section 5.2.3, but also the learning-based adaptive MPC techniques introduced in Section 5.2.2 can be extended to PMPC and be embedded in this framework.

Table 5.1: Definitions of the mathematical notations used in this chapter

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

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

5.3. THE SYNTHESIS FRAMEWORK OF RL-BASED ADAPTIVE PMPC

In this section we first extend the conventional definition of PMPC such that all the components of PMPC can be modified. Based on this definition, we present a novel synthesis framework for RL-PMPC, and further consider five cases, each corresponding to a specification of the novel framework by parameterizing a different component of PMPC. The frequently used mathematical notations are defined in Table 5.1.

5.3.1. EXTENDED PMPC SCHEME

In a general PMPC scheme, there are three time scales: the simulation sampling time T_{s} of the prediction model, the control sampling time T_{c}, and the PMPC operation sampling time T_{p}, and the corresponding counting steps are k_{s}, k_{c}, and k_{p}. The relationships between them are:

$$T_{\mathrm{p}} = m_2 \cdot T_{\mathrm{c}} = m_2 m_1 \cdot T_{\mathrm{s}}, \quad m_1, m_2 \in \mathbb{N}^+, \tag{5.1}$$

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

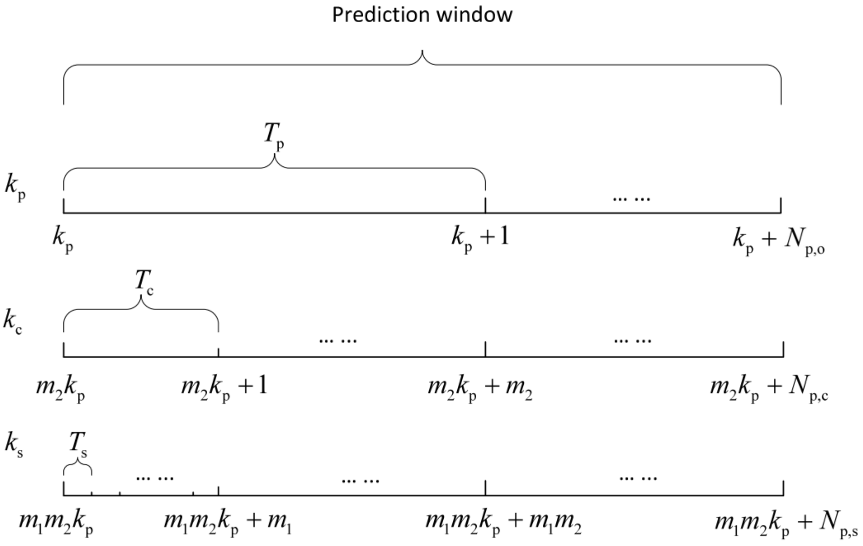

Figure 5.1: Illustration of different time scales of the PMPC scheme during a prediction window, in which k_p is the PMPC operation step, k_c is the control step, and k_s is the simulation step of the prediction model.

are measured every T_s time units[1]. Within the prediction window, each PMPC operation step k_p corresponds to the control steps $\{m_2k_\mathrm{p},\ldots,m_2k_\mathrm{p}+m_2-1\}$, and each control step k_c corresponds to the simulation steps $\{m_1k_\mathrm{c},\ldots,m_1k_\mathrm{c}+m_1-1\}$ of the prediction model. The relationships among different time scales over a prediction window are illustrated in Figure 5.1, in which

$$N_\mathrm{p,s}=m_1\cdot N_\mathrm{p,c}=m_1m_2\cdot N_\mathrm{p,o},\quad m_1,m_2\in\mathbb{N}^+. \tag{5.2}$$

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

$$\begin{aligned}
&\min_{\boldsymbol{u}_\theta(k_\mathrm{p})}\quad J(\tilde{\boldsymbol{x}}(k_\mathrm{p}),\tilde{\boldsymbol{u}}(k_\mathrm{p}),\boldsymbol{\theta}_J) && (5.3)\\
&\text{s.t. } \hat{\boldsymbol{x}}(m_1(m_2k_\mathrm{p}+k)+\ell+1)=F\big(\hat{\boldsymbol{x}}(m_1(m_2k_\mathrm{p}+k)+\ell),\boldsymbol{u}_\mathrm{c}(m_2k_\mathrm{p}+k),\boldsymbol{\theta}_F\big),\\
&\qquad\qquad\qquad\qquad\text{for } \ell=0,\ldots,m_1-1,\ k=0,\ldots,N_\mathrm{p,c}-1, && (5.4)\\
&\mathcal{G}\big(\tilde{\boldsymbol{x}}(k_\mathrm{p}),\tilde{\boldsymbol{u}}(k_\mathrm{p}),\boldsymbol{\theta}_\mathcal{G}\big)\le 0, && (5.5)\\
&\tilde{\boldsymbol{x}}(k_\mathrm{p})=[\hat{\boldsymbol{x}}^\top(m_1m_2k_\mathrm{p}+1),\ldots,\hat{\boldsymbol{x}}^\top(m_1m_2k_\mathrm{p}+m_1N_\mathrm{p,c})]^\top, && (5.6)\\
&\tilde{\boldsymbol{u}}(k_\mathrm{p})=[\boldsymbol{u}_\mathrm{c}^\top(m_2k_\mathrm{p}),\ldots,\boldsymbol{u}_\mathrm{c}^\top(m_2k_\mathrm{p}+N_\mathrm{p,c}-1)]^\top, && (5.7)\\
&\boldsymbol{u}_\mathrm{c}(m_2k_\mathrm{p}+k)=f\big(\hat{\boldsymbol{x}}(m_1(m_2k_\mathrm{p}+k)),\boldsymbol{u}_\theta(k_\mathrm{p}),\boldsymbol{\theta}_f\big),\quad\text{for } k=0,\ldots,N_\mathrm{p,c}-1, && (5.8)\\
&\hat{\boldsymbol{x}}(m_1m_2k_\mathrm{p})=\boldsymbol{x}(m_1m_2k_\mathrm{p}), && (5.9)
\end{aligned}$$

---

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

in which $\boldsymbol{u}_\theta(k_\mathrm{p})$ denotes the parameter variables to be optimized every operation step, $F(\cdot)$ is the prediction model that is parameterized by $\boldsymbol{\theta}_F$, and $\boldsymbol{x}(m_2 m_1 k_\mathrm{p})$ is the measured state vector at the time instant $m_1 m_2 k_\mathrm{p}$; $J(\cdot)$ is the objective function parameterized by $\boldsymbol{\theta}_J$, and $\mathscr{G}$ represents the constraint for the control inputs and states parameterized by $\boldsymbol{\theta}_\mathscr{G}$; f is the state-feedback function, which maps $\boldsymbol{u}_\theta$ to $\boldsymbol{u}_\mathrm{c}$ and which is parameterized by $\boldsymbol{\theta}_f$. Typically, only the first element of the optimized parameter vector, i.e., $\boldsymbol{u}_\theta(k_\mathrm{p})$, is implemented, and the optimization problem is solved again at the next operation step $k_\mathrm{p}+1$.

Remark 8. *For the sake of simplicity, it is assumed in the PMPC formulation* (5.3)-(5.9) *that* $\boldsymbol{u}_\theta(k_\mathrm{p})$ *remains constant during the entire PMPC prediction window. However, it is straightforward to allow* $\boldsymbol{u}_\theta(k_\mathrm{p})$ *to vary with the PMPC operation step within the PMPC prediction window. One choice is to keep the parameters constant for an interval (e.g., several time steps) and then change for the next interval over the prediction horizon, which is a hybrid option called move blocking [21].*

In conventional PMPC [60], [119], only the control inputs are parameterized and the resulting parameters $\boldsymbol{u}_\theta$ are optimized. If we simplify the state-feedback function (5.8) to an identity function such that $\boldsymbol{u}_\mathrm{c} = \boldsymbol{u}_\theta$, then the PMPC problem reduces to a conventional MPC problem. In the PMPC problem (5.3)-(5.9), in addition to the state-feedback function, the other components of the PMPC scheme are all parameterized, including the constraint sets, the objective function, and the system model. This yields an extended PMPC scheme that can cover the existing MPC methods introduced in Section **??**. However, if we take all the parameters as decision variables in (5.3)-(5.9), the resulting optimization problem will be difficult to solve, due to the large number of optimization variables and the nonlinearity and nonconvexity introduced by the parameters. This issue can be addressed by the proposed RL-based adaptive PMPC synthesis framework.

5.3.2. THE SYNTHESIS FRAMEWORK

Instead of designing a specific scheme and tailor a solution for each possible parameterization case separately, we propose to integrate all the possible solutions in an RL-PMPC synthesis framework. As shown in Figure 5.2, all parameterization cases are embedded in this framework, and a high-level RL agent is employed to adapt the parameterized components, such that the complex optimization problem with multiple parameters is avoided. Note that the RL agent in general works with a lower frequency than PMPC to adjust the parameters $\boldsymbol{\theta} = [\boldsymbol{\theta}_F^\top, \boldsymbol{\theta}_f^\top, \boldsymbol{\theta}_J^\top, \boldsymbol{\theta}_\mathscr{G}^\top, \boldsymbol{\theta}_\mathrm{s}^\top]^\top$ of all the parametric components.

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

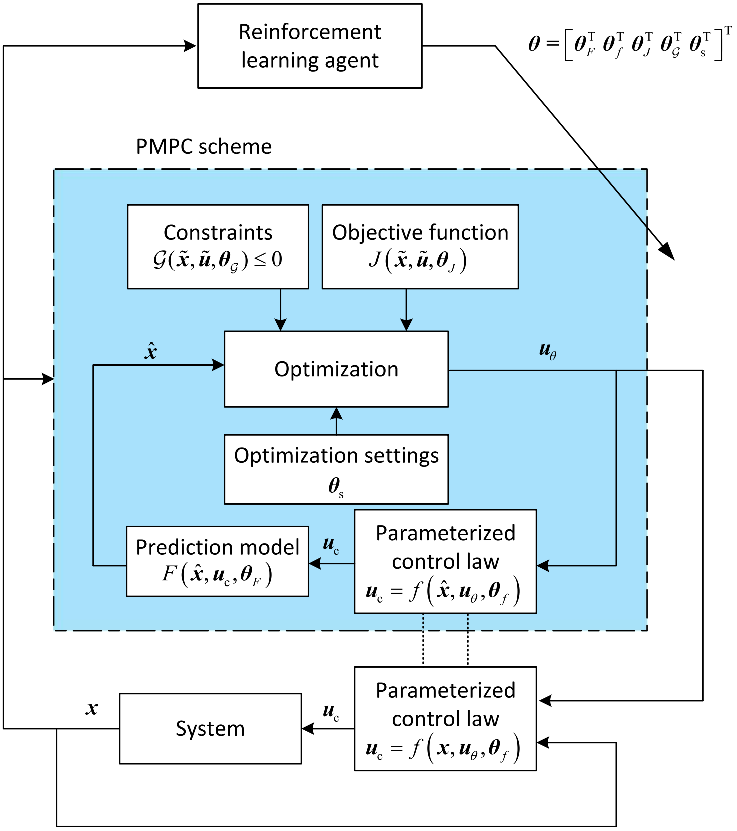
Reinforcement learning agent
$\boldsymbol{\theta} = \left[\boldsymbol{\theta}_F^{\mathrm{T}}\ \boldsymbol{\theta}_f^{\mathrm{T}}\ \boldsymbol{\theta}_J^{\mathrm{T}}\ \boldsymbol{\theta}_{\mathcal{G}}^{\mathrm{T}}\ \boldsymbol{\theta}_{\mathrm{s}}^{\mathrm{T}}\right]^{\mathrm{T}}$
PMPC scheme
Constraints
$\mathcal{G}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}}, \boldsymbol{\theta}_{\mathcal{G}}) \leq 0$
Objective function
$J(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}}, \boldsymbol{\theta}_J)$
$\hat{\boldsymbol{x}}$
Optimization
$\boldsymbol{u}_\theta$
Optimization settings
$\boldsymbol{\theta}_{\mathrm{s}}$
Prediction model
$F(\hat{\boldsymbol{x}}, \boldsymbol{u}_{\mathrm{c}}, \boldsymbol{\theta}_F)$
$\boldsymbol{u}_{\mathrm{c}}$
Parameterized control law
$\boldsymbol{u}_{\mathrm{c}} = f(\hat{\boldsymbol{x}}, \boldsymbol{u}_\theta, \boldsymbol{\theta}_f)$
$\boldsymbol{x}$
System
$\boldsymbol{u}_{\mathrm{c}}$
Parameterized control law
$\boldsymbol{u}_{\mathrm{c}} = f(\boldsymbol{x}, \boldsymbol{u}_\theta, \boldsymbol{\theta}_f)$

Figure 5.2: The synthesis framework of RL-PMPC

DEFINITION OF THE RL AGENT

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

$$T_{\mathrm{rl}} = m_3 \cdot T_{\mathrm{p}}, \quad m_3 \in \mathbb{N}^+, \tag{5.10}$$

and with the corresponding adaption step k_{rl}. The reinforcement learning process is modeled as a discrete-time stochastic control process (i.e., a Markov decision process (MDP)), which can be represented by a five-tuple $\langle S, A, P, \mathscr{R}, \gamma \rangle$. According to the RL agent within the framework (see Figure 5.2), these elements are defined as follows:

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

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

  $$a_{k_{\text{rl}}} = \boldsymbol{\theta} = [\boldsymbol{\theta}_F^\top, \boldsymbol{\theta}_f^\top, \boldsymbol{\theta}_J^\top, \boldsymbol{\theta}_{\mathscr{G}}^\top, \boldsymbol{\theta}_{\text{s}}^\top]^\top. \tag{5.11}$$

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

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

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

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

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

$$\begin{aligned} Q^\pi\left(s_{k_{\text{rl}}}, a^\pi_{k_{\text{rl}}}\right) =& \mathbb{E}_{r,s\sim E}\left[\sum_{k_{\text{rl}}=0}^{\infty} \gamma^{k_{\text{rl}}} r_{k_{\text{rl}}}(s_{k_{\text{rl}}}, a^\pi_{k_{\text{rl}}})\right] \\ =& \mathbb{E}_{r,s\sim E}\left[r_{k_{\text{rl}}}(s_{k_{\text{rl}}}, a^\pi_{k_{\text{rl}}}) + \gamma Q^\pi\left(s_{k_{\text{rl}}+1}, a^\pi_{k_{\text{rl}}+1}\right)\right], \end{aligned} \tag{5.12}$$

where the subscript $r, \boldsymbol{s} \sim E$ denotes the stochastic transitions among the states in the environment, and $a^\pi_{k_{\text{rl}}}$ is the action taken at step k_{rl} based on the policy $\pi(\cdot)$. Depending on the specific problem, various RL algorithms can be chosen. In particular, when dealing with large-scale problems with multiple parameters, the deep RL algorithms that can address continuous state space are preferred, such as Deep Q-Network (DQN) or actor-critic algorithms [65], [111], [133], [134].

Remark 9. *The learning process can be conducted offline (i.e., using a detailed simulation model to generate data), which is known as pre-training, or online (i.e., interacting with the real system), or via a combination of online and offline processes. Both variants have a similar training process (see Algorithm 2).*

Algorithm 2 Offline learning process of the RL-PMPC synthesis framework

```
 1: Initialize the DDPG agent: the critic and actor networks, the corresponding target
    networks and the experience replay buffer
 2: Initialize the PMPC scheme by determining the parameterization, and define the
    state space, action space, and reward function of the DDPG agent
 3: for episode from 1 to $M$ do
 4:   Initialize the system states
 5:   for every RL adaption step $k_{\mathrm{rl}}$ do
 6:     Observe state $s_{k_{\mathrm{rl}}}$
 7:     Take action $a_{k_{\mathrm{rl}}}$ according to state $s_{k_{\mathrm{rl}}}$ and policy $\pi(\cdot)$, and update parameters $\boldsymbol{\theta}$
        of the PMPC scheme
 8:     for every PMPC operation step $k_{\mathrm{p}}$ do
 9:       Measure state $\boldsymbol{x}(m_2 m_1 k_{\mathrm{p}})$
10:       Solve the PMPC problem (5.3), and get the optimized parameter $\boldsymbol{u}_{\theta}(k_{\mathrm{p}})$
11:       for every control step $k_{\mathrm{c}}$ do
12:         Measure state $\boldsymbol{x}(m_1 k_{\mathrm{c}})$ and calculate the control inputs $\boldsymbol{u}_{\mathrm{c}}(k_{\mathrm{c}})$ according to
            (5.8)
13:         for every step $k_{\mathrm{s}}$ do
14:           Implement control input $\boldsymbol{u}_{\mathrm{c}}(k_{\mathrm{c}})$ on the simulation model; measure and
              record states $\boldsymbol{x}(k_{\mathrm{s}}+1)$
15:         end for
16:       end for
17:     end for
18:     Observe the reward $r_{k_{\mathrm{rl}}}$ and next state $s_{k_{\mathrm{rl}}+1}$
19:     Store transition $(s_{k_{\mathrm{rl}}}, a_{k_{\mathrm{rl}}}, s_{k_{\mathrm{rl}}+1}, r_{k_{\mathrm{rl}}})$ in the replay buffer
20:     Sample a mini-batch of $N$ data points from replay buffer randomly
21:     Update the critic and actor (target) networks based on the sampled data accord-
        ing to [111]
22:   end for
23: end for
```

Algorithm 2 summarizes the overall learning process of the proposed framework, and a deep RL algorithm, i.e., deep deterministic policy gradient (DDPG) [111], is used for example[2]. The RL agent is trained for M episodes, and each episode starts from the initial state and ends in the terminal state or at the terminal time step. Note that Algorithm 2 can be easily extended to other RL algorithms. In addition, the online variant of Algorithm 2 can be obtained by changing line 14, in which the simulation model is replaced by the real system. Next, the RL-based modification of each component of the PMPC scheme is discussed separately in detail.

---

[2] The explanation of the DDPG techniques, i.e., critic and actor structure, experience replay, and target networks, is omitted here for compactness. For more details, the reader can refer to [111], [134].

CASE A: RL MODIFYING THE SYSTEM MODEL

Adjusting the model parameters can reduce the mismatch between the prediction model and the real system, thus resulting in more accurate predictions and better control performance. Therefore, the states $s_{k_{\mathrm{rl}}}$ of the RL agent at adaptation step k_{rl} can include the measured states of the real system at the corresponding time step, i.e., $\boldsymbol{x}(k_{\mathrm{s}})$ with $k_{\mathrm{s}} = m_3 m_2 m_1 k_{\mathrm{rl}}$, and other necessary information of the environment, such as the disturbances and PMPC inputs. In this section, the objective of RL (i.e., the reward function) can be either minimizing the modeling errors, as in [193] and [83], or optimizing the control performance directly. For the former case, the reward function can be defined to minimize the error between the predicted states and the measured states. For the latter case, the reward function can be defined to minimize the objective function in PMPC:

$$r_{k_{\mathrm{rl}}}(s_{k_{\mathrm{rl}}}, a_{k_{\mathrm{rl}}}) = -R(\bar{\boldsymbol{x}}(k_{\mathrm{rl}}), \bar{\boldsymbol{u}}(k_{\mathrm{rl}})), \tag{5.13}$$

where $R(\cdot)$ can be similar to the PMPC objective function $J(\cdot)$, and $\bar{\boldsymbol{x}}$ and $\bar{\boldsymbol{u}}$ include the measured states and implemented PMPC inputs during time interval $[k_{\mathrm{rl}} T_{\mathrm{rl}}, (k_{\mathrm{rl}} + 1) T_{\mathrm{rl}})$. For linear systems with parametric uncertainties (e.g., see [122], [210]), one way is to parameterize the system as:

$$F(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\theta}_F) = A(\boldsymbol{\theta}_F)\boldsymbol{x} + B(\boldsymbol{\theta}_F)\boldsymbol{u}. \tag{5.14}$$

The parameters $\boldsymbol{\theta}_F$ can be adjusted by the RL agent directly at every adaption step k_{rl}, and the action can be the corresponding parameter values. In this way, RL can adjust the varying parameters caused by the changing environment, via interacting with the environment or a simulation model. This strategy can also be extended to more general linear systems with other parametric models than (5.14).

For nonlinear systems, consider a widely-used parametric nonlinear model [2]:

$$F(\hat{\boldsymbol{x}}, \boldsymbol{u}, \boldsymbol{\theta}_F) = F_f(\hat{\boldsymbol{x}}, \boldsymbol{u}) + F_g(\hat{\boldsymbol{x}}, \boldsymbol{u})\boldsymbol{\theta}_F, \tag{5.15}$$

where $\boldsymbol{\theta}_F$ can be the adjusted by the RL agent. Moreover, according to [184], basis functions can be used to construct linear parameter-varying models. Then the RL agent can be used to tune the weights of the chosen basis functions. This can also be applied to the cases where artificial neural networks (ANNs) are used to approximate the nonlinear system. For example, the RL agent can be used to tune the weights of the neurons of ANNs [5], or to compensate for the modeling errors between the real system and the ANNs [151]. Furthermore, some systems have different modes or dynamics under different working conditions, which can be described by switching among several system models. Then the RL agent can be used to select the suitable system model to adapt to varying conditions.

CASE B: RL MODIFYING THE PARAMETERIZED CONTROL LAWS

Parameterizing the control laws with state-feedback functions (i.e., control law $f(\cdot)$) is the main way to reduce the computation time in this framework. Several studies (see, e.g. [100], [154], [207]) consider a fixed control law that is pre-designed based on the experience or expert knowledge. This handcrafted design may work well for some scenarios, but the control performance may deteriorate when the system conditions change over time. Therefore, in this section, the RL agent within this framework allows to tune the control law $f(\cdot)$, which is parameterized by $\boldsymbol{\theta}_f$. Consider the following example, where the control law $f(\cdot)$ is a combination of several basis functions:

$$f(\hat{\boldsymbol{x}}, \boldsymbol{u}_\theta, \boldsymbol{\theta}_f) = \sum_{i=1}^{n_f} \theta_{f,i} \phi_{f,i}(\hat{\boldsymbol{x}}, \boldsymbol{u}_\theta), \tag{5.16}$$

in which $\boldsymbol{\theta}_f = [\theta_{f,1}, \ldots, \theta_{f,n_f}]^\top$ with n_f the number of the basis functions. Accordingly, the action of the RL agent can be defined by (5.11). The basis functions $\phi_{f,i}(\hat{\boldsymbol{x}}, \boldsymbol{u}_\theta), i = 1, \ldots, n_f$ should be designed a priori to handle various system conditions. They can be constructed empirically or by resorting to a learning-based method. For example, the grammatical evolution algorithm can be used to generate the control laws automatically in an offline style [90]. Furthermore, the state space and reward function for this case can be defined the same way as in Case A.

CASE C: RL MODIFYING THE OBJECTIVE FUNCTION AND CONSTRAINT SETS

It has been shown that the objective function and constraints within the PMPC scheme can be adjusted to further improve the control performance [64], while it has not been illustrated how to systematically parameterize the objective function. Arroyo *et al.* [7] also approximated the infinite cost of the objective function by using a stage cost and a value function. However, this leads to increased complexity in solving the optimization problem, since the value function introduces extra nonlinearity and nonconvexity to the optimization problem. Arroyo *et al.* [7] used an exhaustive search method to find an optimal action per time step. In this section, we propose to rewrite the objective function (5.3) as in [64], which is given by:

$$\begin{aligned}
&J(\tilde{\boldsymbol{x}}(k_\mathrm{p}), \tilde{\boldsymbol{u}}(k_\mathrm{p}), \boldsymbol{\theta}_J) \\
&= \sum_{k=0}^{N_{\mathrm{p,c}}-1} \sum_{\ell=0}^{m_1-1} L(\hat{\boldsymbol{x}}(m_1(m_2 k_\mathrm{p} + k) + \ell), \boldsymbol{u}_\mathrm{c}(m_2 k_\mathrm{p} + k), \boldsymbol{\theta}_J) \\
&\quad + T(\hat{\boldsymbol{x}}(m_1 m_2 k_\mathrm{p} + m_1 N_{\mathrm{p,c}}), \boldsymbol{\theta}_J),
\end{aligned} \tag{5.17}$$

where $L(\cdot)$ and $T(\cdot)$ are the stage cost function and terminal cost function parameterized by $\boldsymbol{\theta}_J$. One possible realization of the parameterized stage cost $L(\cdot)$ and terminal cost $T(\cdot)$ can be similar to (5.16), i.e., a combination of basis functions weighted by the parameters. In addition to the methods presented in Section 5.3.2 for constructing the basis functions, radial basis functions (RBFs) can also be considered since they have the universal approximation property [131], [150]. The selection of the centers and weights of the RBFs can be done by the RL agent within this framework, where these parameters can be integrated into $\boldsymbol{\theta}_J$, and the action of the RL agent can be the same as (5.11).

The terminal constraint set is important in MPC theory to guarantee the recursive feasibility and it can be adapted online [166]. However, Rosolia *et al.* [166] did not consider the changing environment or disturbances. In this case, all the constraint sets are integrated in the function $\mathcal{G}(\hat{\boldsymbol{x}}, \boldsymbol{u}, \boldsymbol{\theta}_\mathcal{G})$, which is parameterized by $\boldsymbol{\theta}_\mathcal{G}$. Therefore, $\boldsymbol{\theta}_\mathcal{G}$ can be tuned by the RL agent to respond to the varying environmental conditions. Furthermore, the state space and reward function for this case can be defined as in Case A.

CASE D: RL MODIFYING THE OPTIMIZATION SETTINGS

In this section, we propose to use the RL agent to modify the optimization settings, such as the length of the prediction horizon as in [153] and the optimization options of a solver. Proper tuning of the prediction horizon can result in a balance between computational complexity and performance, so as the solver options. Different optimization algorithms may lead to various results, and for each algorithm, the optimization settings such as the constraint tolerance, step size tolerance, function value tolerance, and other parameters, which significantly influence the optimization speed and accuracy, should be pre-selected. Within our proposed framework, these parameter values are allowed to be adapted according to the varying environment and control objective. The reward function of the RL agent for this case should be revised, for instance to be defined as a combination of the control performance and the computational efficiency, which is given by:

$$r_{k_{\mathrm{rl}}}(s_{k_{\mathrm{rl}}}, a_{k_{\mathrm{rl}}}) = -R(\bar{\boldsymbol{x}}(k_{\mathrm{rl}}), \bar{\boldsymbol{u}}(k_{\mathrm{rl}})) - J_{\mathscr{C}}(a_{k_{\mathrm{rl}}}), \tag{5.18}$$

where $J_{\mathscr{C}}(\cdot)$ is an index that denotes the computational efficiency of the solver (e.g., the computation time for solving the optimization problem). Accordingly, the state space of the RL agent can be defined as in Case A, and the action can be given by (5.11).

As mentioned in Remark 8, the optimization variables $\boldsymbol{u}_\theta$ can be changed in a move blocking way. Let N_{b} denote the number of the PMPC operation steps where the parameters remain constant within the prediction window. If $N_{\mathrm{b}} = N_{\mathrm{p,o}}$, then $\boldsymbol{u}_\theta$ is constant over the prediction window. In this case, the computation time is reduced, but will in general result in less optimal performance. Therefore, N_{b} can also be a parameter that is tuned by the RL agent to reach a trade-off between the performance and the computation time.

CASE E: RL MODIFYING PARAMETERIZED CONTROL INPUTS

This is a degenerate case with the PMPC module, in which the RL agent is used to tune $\boldsymbol{u}_\theta$ generated by the optimization process of the PMPC module. This is different from conventional studies [161], [178], [211], in which the RL agent directly adjusts the control inputs that are fed into the system. Considering a simple case for (5.8), where the parameterized control law is a linear function:

$$\boldsymbol{u}_{\mathrm{c}}(k_{\mathrm{c}}) = \boldsymbol{u}_\theta \hat{\boldsymbol{x}}(m_1 k_{\mathrm{c}}), \tag{5.19}$$

the corresponding RL action is

$$a_{k_{\mathrm{rl}}} = \Delta \boldsymbol{u}_\theta, \tag{5.20}$$

where $\Delta \boldsymbol{u}_\theta$ is the adjustment value to $\boldsymbol{u}_\theta$. Compared to adjusting $\boldsymbol{u}_{\mathrm{c}}$ or determining $\boldsymbol{u}_\theta$ directly, tuning the parameters $\Delta \boldsymbol{u}_\theta$ is expected to be more robust in terms of the control performance during the learning process. In particular, without the MPC scheme (i.e., the receding horizon optimization process), the framework will be reduced to an RL-based adaptive state feedback controller, as in [177]. The state space and reward function can be defined as in Case A.

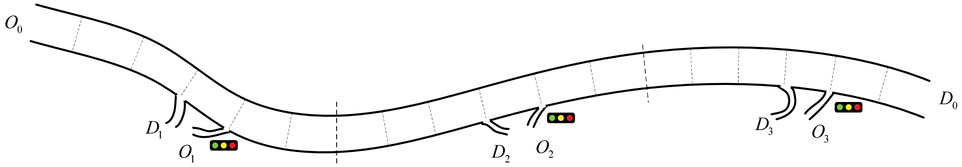

Figure 5.3: The layout of the freeway network in this case study

5.4. CASE STUDY

In this section, Case B from Section 5.3.2 is implemented on a freeway network, in order to illustrate the proposed framework. First, the freeway network is presented, followed by the traffic demand profiles, external disturbances, and weather conditions that introduce parameters uncertainties. Then the parameterized control law based on ramp metering (RM) is introduced. A DQN agent is utilized in the proposed framework to tune the parameters of the parameterized RM control law. The performance of the proposed framework is compared with conventional MPC, PMPC, and a standalone RL controller.

5.4.1. FREEWAY NETWORK

The benchmark freeway network from [118] is used in this case study, which is presented in Figure 5.3. This freeway network is divided into 18 segments of 1000 m long. There are 1 mainstream origin (O_0), 3 on-ramps (O_1, O_2, O_3), 1 unrestricted destination (D_0), and 3 unrestricted off-ramps (D_1, D_2, D_3). All the three on-ramps are regulated by a traffic light, which can control the ramp metering rate (i.e., ramp metering (RM) control). Therefore, there are 3 control signals in total for this network. In this case study, METANET is used to represent the freeway network, and the perturbed version of the same model is used as the prediction model for PMPC. Therefore, both the controlled system and the prediction model have the same simulation sampling time. METANET is a second-order macroscopic traffic flow model that has been widely used in freeway traffic control [73], [118], due to its ability to reproduce freeway traffic phenomena with relatively less computational complexity. More details about METANET can be found in [102], [130].

In this case study, a scenario of recurrent traffic demand for 2 hours during the rush hour is considered. Traffic demands from all origins (O_0, O_1, O_2, O_3) are presented in Figure 5.4. In addition, a shock wave from the downstream boundary is generated to produce extra traffic jams. Such a shock wave is an abrupt increase in traffic density that will propagate from downstream to upstream. The downstream density profile is presented in Figure 5.5. The parameters of the freeway model are taken from [118]. In this case study, environment changes are considered that can influence the parameters of the traffic network. More specifically, the weather condition is taken into consideration, which is classified into three levels: good weather, bad weather, and extreme weather. The real parameters of the freeway model and the estimated parameters of the prediction model are presented in Table 5.3, for different weather conditions. The mathematical notations of the parameters are the same as [118]. For definitions of the parameters, the reader can refer to [73], [118].

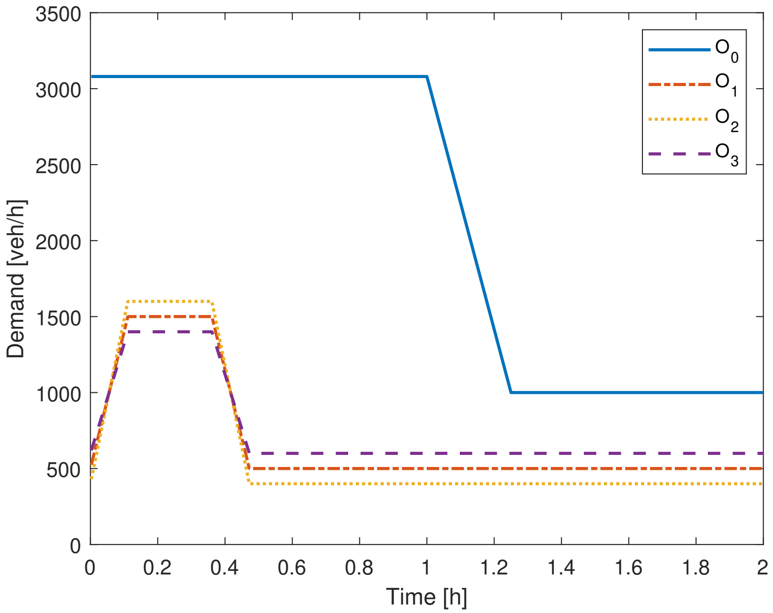

Figure 5.4: Traffic demand profiles for the origins of the freeway network

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

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

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

5.4.2. PARAMETERIZED FREEWAY TRAFFIC CONTROL LAWS

Ramp metering (RM) rates have been widely used in freeway traffic management [73]. This control measure was further parameterized by [207] in an MPC framework. The

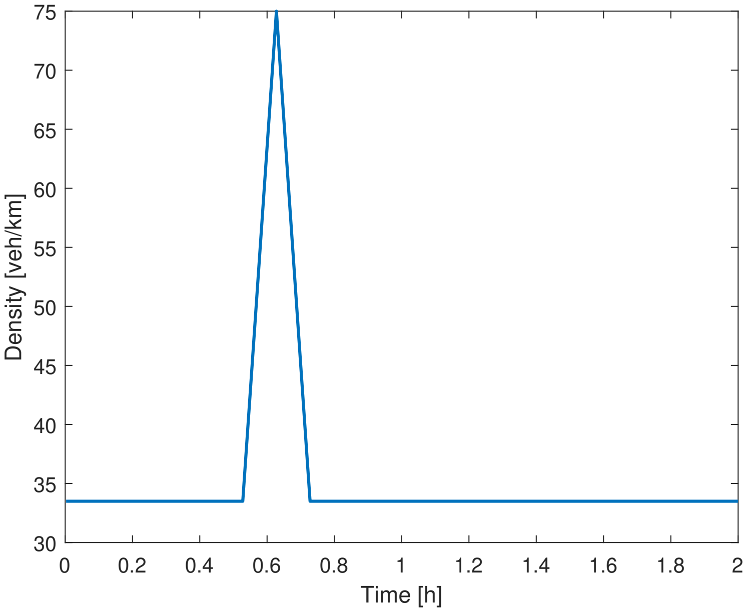

Figure 5.5: The downstream density used to generate shock wave for the freeway network

parameterized control law of RM used in this case study is based on [207], and is given by:

$$u_{\mathrm{rm},i}(k_\mathrm{c}+1) = u_{\mathrm{rm},i}(k_\mathrm{c}) + u_\theta(k_\mathrm{p})\left(\theta_f(k_\mathrm{rl}) - \rho_i(k_\mathrm{c})\right), \tag{5.21}$$

where k_c is the control step, k_p is the PMPC operation time step, k_rl is the RL operation time step, $u_{\mathrm{rm},i}(k_\mathrm{c})$ is the RM control input for the on-ramp that is linked to segment i, $\rho_i(k_\mathrm{c})$ is the measured traffic density of the downstream segment i of the on-ramp, and $u_\theta(k_\mathrm{p})$ is the parameter optimized by PMPC at operation step k_p. Note that the control law (5.21) is derived from ALINEA [146], in which the original parameter θ_f is the setpoint density obtained through experiments and history data. In this case study, the same parameterized control law (5.21) is applied to all the three on-ramps, in which $\theta_f(k_\mathrm{rl})$ is the parameter that is tuned by the RL agent at operation step k_rl.

5.4.3. CONTROLLERS

All the MPC-based controllers in this case study use the prediction model with estimated parameters given in Table 5.3. In addition, the learning-based controllers (i.e., standalone RL controller and RL-PMPC controller) are trained off-line with the prediction model, and are validated via the system with real parameters. The simulations are performed on a PC with an Intel Xeon Quad-Core E5-1620 V3 CPU with a clock speed of 3.5 GHz.

In this case study, the total time spent (TTS) by all the vehicles in the entire freeway network for all the 5 weather scenarios is taken as the performance criterion for the controllers. For the METANET model, simulation sampling time is $T_\mathrm{s} = 10\,\mathrm{s}$. The control sampling time for the parameterized control laws is $T_\mathrm{c} = 60\,\mathrm{s}$; the PMPC scheme operation sampling time is $T_\mathrm{p} = 300\,\mathrm{s}$. The control input constraints are given as:

$$0 \le u_\mathrm{rm} \le 1.$$

Therefore, the RM control inputs generated by (5.21) should be saturated within the bounds. The projection method in [90] is utilized to enforce the constraints on the control inputs, and it has been illustrated that the projection-based PMPC can achieve better or equal performance than conventional PMPC with constraints. During the entire 2-hour simulation, it is assumed that the prediction model used by the (P)MPC controllers is fixed with the estimated parameters corresponding to the initial weather condition. Meanwhile, the parameter θ_f in the parameterized control law remains constant with the value of critical density ρ_crit of the initial weather condition for the standalone PMPC controller.

STANDALONE PMPC CONTROLLER

The objective function of the PMPC scheme only contains the TTS within the prediction window. The PMPC operation sampling time is 300 s, i.e., that the parameterized optimization problem is solved every 300 s, while the parameterized control law (5.21) works on the basis of 60 s (i.e., provides control inputs according to the states every 60 s). The length of both the prediction horizon and the control horizon is 900 s. Thus, $N_\mathrm{p,s} = 90$, $N_\mathrm{p,c} = 15$, $N_\mathrm{p,o} = 3$. Furthermore, $N_\mathrm{b} = 1$, which means that the optimized parameters are allowed to change per PMPC operation step (i.e., every 300 s) within the prediction window. Therefore, the number of the optimization variables is $1 \times N_\mathrm{p,o}/N_\mathrm{b} = 3$.

The sequential quadratic programming (SQP) algorithm [15] is implemented via the `fmincon` function from Maltab to solve the nonlinear constrained optimization problem. To avoid getting stuck in local optima, multiple starting points are selected randomly to solve the optimization problem, and the best solution is taken as the final result. In this case study, the number of initial points is selected to be 40, which is determined according to the experiments, as in this way a balance is achieved between optimality and computational efficiency. In addition, the stopping criteria of the SQP algorithm are also tuned, in which the cost function tolerance, step tolerance, and constraint tolerance are all selected to be 10^{-2}.

STANDALONE MPC CONTROLLER

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

STANDALONE RL CONTROLLER

The definition of the RL agent is similar to what has been explained in Section 5.3.2. More specifically, the state space of the RL agent consists of the measured traffic state $\boldsymbol{x}$ at RL operation step k_rl, the traffic demands, downstream boundary density, previously

Table 5.2: Training parameters of the DQN agent

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

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

Accordingly, a deep Q-Network (DQN) [134] agent is used, which can address the continuous state space and the discrete action space. The neural network consists of one input layer, one output layer, and three hidden layers. The size of the input and output layers correspond to the dimensions of state and action spaces, respectively. The three hidden inner layers have 64, 256, and 64 neurons, and each of them uses a ReLU activation function. The training parameters of the DQN agent are given in Table 5.2, in which ϵ is the exploration parameter decaying from the initial value to the minimum value with the decay rate. A higher value ϵ encourages more exploration. Thus, the agent has a high probability to choose actions randomly in the early learning stage, and the probability decreases gradually as the training procedure evolves. Similarly to [178], $n-$step TD (temporal difference) is also used in this DQN agent to improve the learning and control performance, with $n = 15$.

RL-PMPC CONTROL FRAMEWORK

The RL-PMPC control framework consists of a PMPC controller and an RL agent. The PMPC controller is the same as the standalone PMPC controller defined in Section 5.4.3. The DQN agent defined in Section 5.4.3 is also used in this control framework. In addition to the state space of the standalone RL controller, the agent in this framework has extra state variables, i.e., the PMPC output $u_{\theta}(k_{\mathrm{p}})$. Furthermore, the action space is also different. The RL agent within the framework tunes the parameters $\theta_f(k_{\mathrm{rl}})$ of the control law (5.16) at every RL operation step k_{rl}. Based on numerical tuning experiments, the range of the parameter θ_f is set from 15 veh/km/lane to 40 veh/km/lane, and the action space is discretized into 11 actions distributed equidistantly within this range. The parameter selected by the RL agent is applied to all the on-ramps.

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

5.4.4. RESULTS AND DISCUSSION

The TTS and CPU time results for each controller per scenario are collected as the average values of 10 independent runs, and the results are presented in Table 5.4, in which the mean computation time is the average time required for the optimization process per control step, and the max computation time corresponds to the maximum over all the control steps of the computation time per step.

Table 5.4 shows that all the controllers improve the performance in terms of TTS with regard to the no-control case. More specifically, the standalone MPC controller provides the best TTS performance among all the controllers for scenarios 1-4. This is because the standalone MPC controller optimizes the control inputs directly for each on-ramp, and can thus provide the optimal control inputs. However, the standalone MPC controller results in the largest computation time for all the scenarios, in terms of both mean and max computation time. In comparison, the standalone PMPC controller significantly reduces the computation time with regard to the standalone MPC controller, and it provides a better TTS control performance than the standalone RL controller for Scenarios 2, 4 and 6. The computation time of the standalone RL controller is negligible since only an online neural network evaluation is required to obtain the control inputs. Nevertheless, the standalone RL controller is sensitive to the model mismatches between the prediction model (i.e., the training model) and the real system (i.e., the validation model). Therefore, even when the RL agent is trained with sufficient number of data samples, the validation performance still cannot be guaranteed (see Scenarios 2, 4 and 6 in Table 5.4).

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

Furthermore, the traffic states during the simulation interval of different controllers are depicted in Figure 5.6 to Figure 5.9, and Scenario 1 is selected to compare the performance of the controllers. As shown in Figure 5.6, the traffic demands from the on-ramps O_2 and O_3 at the time 0.25 h cause traffic congestion at segment 11 and 17, which propagates to the upstream gradually. After the time 0.5 h, the shock wave from the downstream of segment 18 causes another jam wave moving backwards. At the time 1 h, the traffic situation gets worse due to the weather change. As shown in Figure 5.7, the standalone PMPC controller can alleviate the traffic congestion caused by the on-ramp traffic demands during the first hour by regulating the on-ramp metering rate. However, the jam wave can not be eliminated through on-ramp metering, and the congestion is still severe during the second hour due to the bad weather. The standalone RL controller can improve the traffic situation during the second hour, due to its adaptation to the changing environment. Moreover, the proposed RL-PMPC controller can further improve the traffic efficiency compared with both the standalone PMPC controller and the standalone RL controller. It is not only because the RL-PMPC controller is adaptive to the changing environment, but also because it has a better sample efficiency and learns better than the standalone RL controller with limited training data.

Remark 10. *In this case study, we only consider one freeway control measure (i.e., ramp metering) to illustrate the concept of the RL-PMPC framework. If we introduce extra freeway control measures (e.g., variable speed limits), the RL-PMPC controller has more freedom to tune the corresponding parameters, which is hypothesized to further improve the control performance with regard to the standalone PMPC controller.*

5.5. CONCLUSIONS

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

In the future, stability and recursive feasibility of the novel control framework will be investigated. Moreover, the performance can be compared with existing robust MPC methods. In addition, the proposed framework can be extended to a multi-agent variant

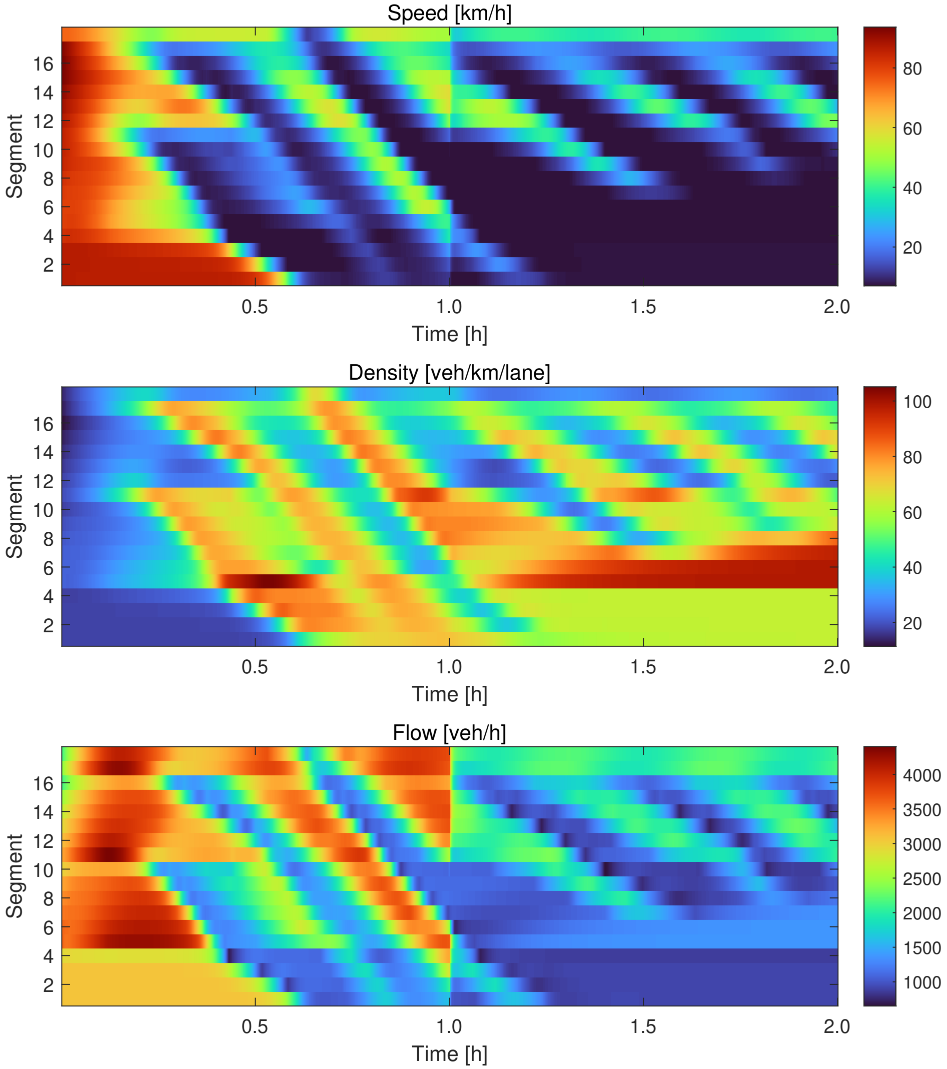

Figure 5.6: Traffic states of the freeway network during the simulation interval without control for Scenario 1.

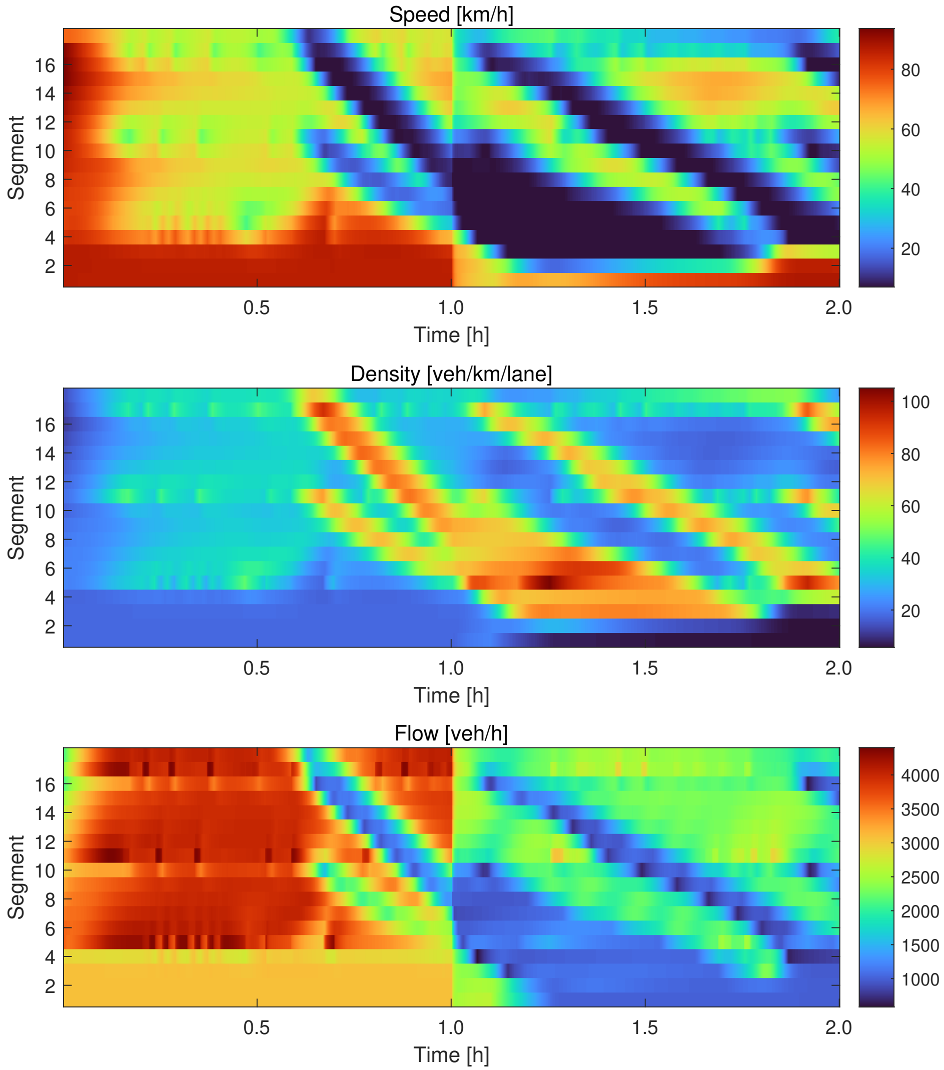

Figure 5.7: Traffic states of the freeway network during the simulation interval with the standalone PMPC controller for Scenario 1.

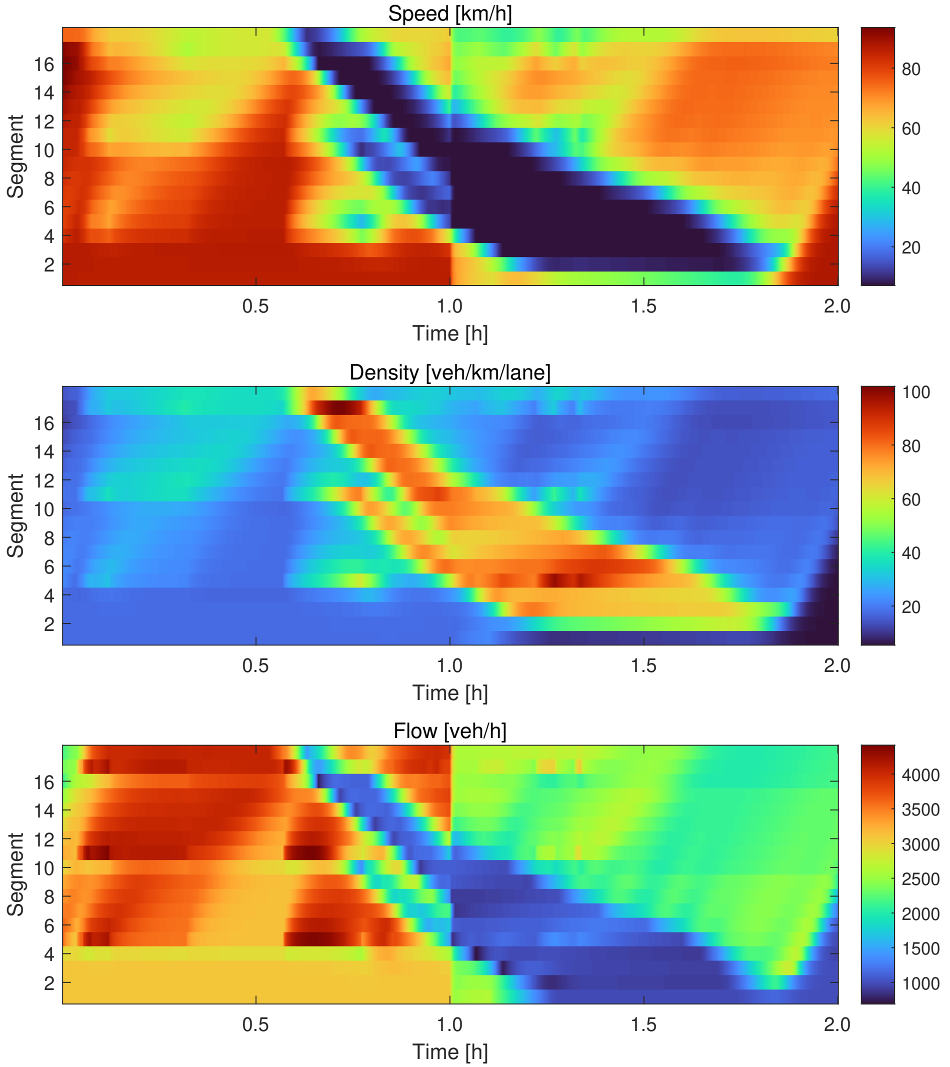

Figure 5.8: Traffic states of the freeway network during the simulation interval with the standalone RL controller for Scenario 1.

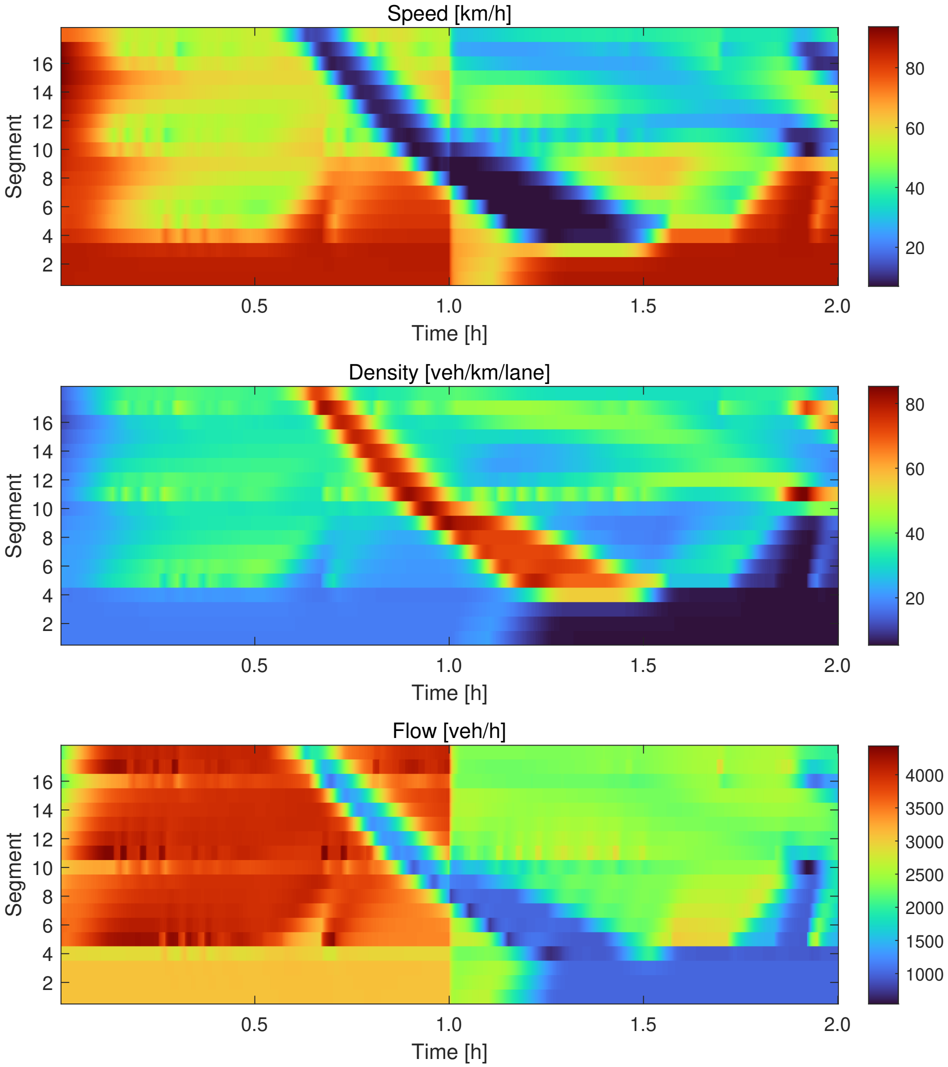

Figure 5.9: Traffic states of the freeway network during the simulation interval with the proposed RL-PMPC controller for Scenario 1.

by integrating distributed PMPC and multi-agent RL, such that it can also be applied to larger networks.

Table 5.3: Parameters of the freeway network under different weather conditions

	Weather condition	T[s]	τ[s]	κ[veh/km/lane]	η[km^2/h]	a_m	σ	v_{free}[km/h]	ρ_{crit}[veh/km/lane]	α	$\rho_{\max}$[veh/km/lane]	L_m[m]
Real	Good	10	18.0	40	65.0	1.867	0.1	102	33.5	0.1	180	1000
	Bad	10	27	60	55.0	1.667	0.3	92	26.5	0	150	900
	Extreme	10	36	80	45.0	1.467	0.5	82	19.5	-0.1	120	800
Estimated	Good	10	18.9	42	68.3	1.960	0.11	107	35.2	0.11	189	1050
	Bad	10	28.4	63	57.8	1.750	0.32	97	27.8	0	158	945
	Extreme	10	37.8	84	47.3	1.540	0.53	86	20.5	-0.11	126	840

Table 5.4: Comparison of the control performance for different controllers for different weather scenarios, in terms of TTS, mean computation time, and max computation time, in which '-' means that the corresponding item is not applicable to the controller.

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

6

CONCLUSIONS AND RECOMMENDATIONS

This chapter concludes and emphasizes the impact of this thesis, and provides recommendations for future research on the basis of this thesis.

6.1. CONCLUSIONS

In this thesis, we have addressed the challenges of MPC for traffic management in terms of computational complexity and model mismatches by developing several novel MPC-based control frameworks for urban and freeway traffic networks. More specifically, the main contributions of this thesis are summarized as follows:

- We have proposed a novel bi-level temporally-distributed MPC framework to deal with the green urban mobility issue that usually involves long-term (e.g., one year) emission constraints, and is thus computationally intractable due to the large window of the problem. The proposed framework contains a high-level MPC controller based on a rough prediction model that incorporates the long-term emission constraint into its decisions, and a low-level MPC controller based on a detailed prediction model that provides traffic signal control inputs for the urban traffic network. In this way, the computational challenge of MPC is addressed and the long-term emissions can be reduced due to the integrated long-term and short-term optimization.

- We have employed a grammatical evolution method to generate parameterized control laws for PMPC with application to urban traffic signal control. Two training frameworks have been developed to ensure that the generated control laws steer the system to the desired direction. With a properly learned parameterized control law, the resulting PMPC controller can significantly reduce computational burdens with comparable control performance, compared to the conventional MPC controller.

- We have developed a novel combined MPC-DRL framework, in which the MPC module provides a basic control performance at a lower frequency based on a prediction model, and the DRL module works at a higher frequency to compensate for the model mismatches and external disturbances through learning. Therefore, the real-time computational complexity can be reduced. Furthermore, for the case study that was considered, the combined MPC-DRL framework showed to outperform both standalone MPC and DRL methods.

- We have proposed a synthesis framework of RL-based adaptive PMPC. In this framework, all components of the PMPC scheme, such as the cost function, the prediction model, the control law, the constraint set, and the terminal set, can be parameterized and adjusted by a high-level RL agent. In addition to the advantage of computational efficiency that is provided by PMPC, the proposed framework can also deal with external disturbances and changing environments through online learning and adaptation. Furthermore, this framework broadens the field of adaptive MPC by proposing several novel adaption strategies. This framework has been applied to a freeway network to adapt the PMPC controller under model mismatches and changing weather conditions, and the results show that the performance of PMPC can be further improved by the RL agent.

6.2. IMPACTS OF THIS THESIS

6.2.1. SOCIAL IMPACTS

The work of this thesis has several potential social impacts, which can be summarized as follows.

GREEN URBAN MOBILITY

The bi-level temporally-distributed MPC framework proposed in Chapter 2 provides a solution to address long-term emission constraints (e.g., annual limits on the overall emissions). This approach can deal with the emissions that are generated by road transportation, including urban traffic and freeway traffic. In coordination with the climate policies, it provides an approach to contribute to fulfilling the climate objectives, such as carbon neutrality.

IMPROVED TRAFFIC EFFICIENCY AND RESILIENCE

By employing learning-based MPC as discussed in Chapter 4 and Chapter 5 to regulate traffic, significant reductions in congestion can be achieved. The approaches enable optimal decision-making during rush hours and under varying weather conditions, leveraging prediction, optimization, and learning techniques. As a result, travel delays can be minimized, and the resilience of traffic systems can be enhanced, effectively addressing unexpected conditions and external disturbances.

EXTENSION TO OTHER APPLICATIONS

The methodologies developed in this thesis possess applicability beyond the field of traffic management. For instance, the bi-level temporally-distributed MPC framework can find utility in energy management for buildings, facilitating the scheduling of energy usage over extended periods. Similarly, the learning-based MPC frameworks introduced in this research have the potential to address challenges in other large-scale networks characterized by model mismatches and computational complexity, including smart grids and smart buildings. By leveraging these methods, significant contributions can be made towards realizing a more energy-efficient and environmentally sustainable society.

6.2.2. SCIENTIFIC AND TECHNICAL IMPACTS

This thesis also contributes to the state-of-the-art MPC approaches that have been verified to be effective for traffic management.

MULTI-LEVEL MPC SCHEMES

According to the literature [169], multi-level MPC schemes can be classified into four categories, as introduced in Chapter 2. The bi-level MPC structure proposed in Chapter 2 belongs to one of the categories. However, the multi-level learning-based MPC frameworks proposed in Chapter 4 and 5 are novel and extend the existing categories of multi-level MPC frameworks, by combining learning algorithms and MPC schemes in a hierarchical structure. Therefore, this thesis presents a range of innovative multi-level MPC control frameworks that effectively tackle the computational complexity associated with the conventional MPC. The proposed multi-level structures also offers the flexibility to incorporate additional control methods or techniques, such as RL-based controllers, to enhance the performance of the conventional MPC. By leveraging this multi-level approach, significant advancements can be achieved in terms of control performance and computational efficiency, thereby paving the way for further improvements in the field of MPC.

LEARNING-BASED MPC SCHEMES

This PhD thesis presents several innovative learning-based MPC schemes, namely GE-based PMPC, the combined MPC-DRL framework, and RL-based adaptive PMPC, which are different from existing literature where the learning methods are usually used to learn the model errors or prediction residuals [40], [63], [64]. The proposed learning-based MPC methods in this thesis represent significant advancements in the field of learning-based control and offer alternative approaches to enhance the performance of conventional control methods. By integrating techniques from evolutionary computation, deep reinforcement learning, and adaptive control, these schemes leverage the power of machine learning to improve the effectiveness, efficiency, and adaptability of MPC in various applications. The development and evaluation of these novel learning-based MPC schemes contribute to the expanding landscape of control methodologies and provide valuable insights for future research and applications in the field.

6.3. RECOMMENDATIONS FOR FUTURE RESEARCH

Based on the work of this thesis, several recommendations are provided for future work, which can be further divided into application recommendations and theory recommendations.

6.3.1. RECOMMENDATIONS IN TERMS OF APPLICATIONS

VARIOUS TRAFFIC CONDITIONS

In this thesis, weather conditions and shock waves are explicitly considered. However, when considering a complex traffic environments, such as a lane drop due to accidents or road closure, or by considering multiple user types (e.g., cars, public traffic, pedestrians, etc.), the traffic management problem becomes more complicated. In order to address these issues, more advanced traffic models need to be developed, such that these conditions are explicitly considered by the control system. This also poses a requirement for having more detailed traffic simulators.

CONNECTED AUTONOMOUS VEHICLES

As more and more connected autonomous vehicles (CAVs) appear on the roads, they play an important role in traffic management. CAVs can communicate with each other and with the traffic infrastructures, which therefore can help regulate traffic flows. For example, CAVs can limit the speed of traffic flow by complying with the commands given by the controller. Controlling heterogeneous traffic flows with both CAVs and vehicles with human drivers is an emerging topic. Therefore, developing an advanced traffic model that incorporate the role of CAVs is an urgent task.

ENHANCED CONTROL MEASURES AND APPROACHES

To deal with the increasingly complex traffic environments, more advanced control measures are needed. Except for conventional traffic signal controllers, infrastructure-to-vehicle (I2V) communications or vehicle-to-vehicle (V2V) communications provide more traffic management tools. Taking advantage of emerging control measures and increasing traffic data is a challenge. In addition, enhanced control approaches are also required to deal with the increasing real-time computational complexity due to the complex traffic models.

CONTROL OF CYBER-PHYSICAL NETWORKS

A traffic network is a typical realization of a cyber-physical network system, and the methods proposed in this PhD thesis can be applied to other types of cyber-physical networks, such as to smart energy grids, smart buildings, and water networks. The proposed methods provide several approaches to deal with cyber security, human interactions, and complex interdependencies of cyber-physical networks.

6.3.2. RECOMMENDATIONS IN TERMS OF THEORY

In terms of the theoretical developments based on the work of this thesis, there are a few recommendations that can further extend the proposed control methods.

EXTENSIONS OF THE RL-BASED ADAPTIVE PMPC

The RL-based adaptive PMPC approach provides the possibility to parameterize all the components of the PMPC scheme. Each case can be further studied. For example, RL-based adaptive PMPC to adjust the system model is related to conventional adaptive MPC techniques, and therefore adaptive MPC theories can be used to develop theoretical results to analyze stability and optimality for the proposed framework.

DISTRIBUTED MPC-RL FRAMEWORK

This is a natural extension of the work done in Chapter 4 of this PhD thesis, where the proposed MPC-RL combined control framework only considered a centralized model predictive control (MPC) module and a single reinforcement learning (RL) agent. Thus, the proposed framework is limited to application to relatively small networks (e.g., an urban traffic network that consists of several intersections, or a freeway traffic network that is several kilometers long with a few on-ramps). In order to increase the scalability of the combined MPC-RL framework such that it can handle city-wide networks, we can integrate distributed MPC and multi-agent RL. Both distributed MPC and multi-agent RL have already been applied to large networks [26], [37], [199], [204]. However, the combination of these two methods has not been investigated yet for large-scale networks. The main challenge may lie in the coordination and interaction between different local MPC modules and RL agents, such that the framework can work efficiently and deal with system uncertainties. In addition, the proof of stability and recursive feasibility can be a challenging task.

MULTI-TASK LEARNING-BASED CONTROL

Although RL has been widely used in various applications, current studies only consider a single scenario for an RL agent, which implies that each RL agent is trained to deal with one specific object or task. For example, an RL agent can be trained for a specific traffic network with predefined traffic demands, and it may perform well for this control task. However, its performance may deteriorate significantly when dealing with different traffic networks or different control tasks. This restricts the applications of RL methods, as well as the learning-based control methods, especially when various possible system failures and attacks should be considered. Therefore, developing a multi-task learning-based control method is of great practical significance. The main challenges lie in the hugely increased number of data samples that are required for learning and the forgetting issue when switching among different tasks. To address these issues, we may take advantage of some advanced machine learning algorithms (e.g., transfer learning), such that the knowledge acquired from one task can be leveraged and transferred across the other tasks, and we may use advanced update methods of neural networks to store the useful memory of specific tasks.

GUARANTEE FOR SAFE LEARNING AND CONTROL PERFORMANCE

In this PhD thesis, it has been shown that the proposed learning-based MPC frameworks achieved better control performance than the conventional MPC and resulted in a better learning process than the conventional RL in terms of sample efficiency and learning safety. For example, the combined MPC-RL framework provides basic performance even during the learning process, which is safer than the standalone RL, and has no state constraint violation during the implementation. However, these results are validated empirically without theoretical guarantees. The main challenges for investigating the theoretical proof of stability, safety, and optimality of the proposed method are related to the difficulty of achieving theoretical results for nonlinear MPC, especially when uncertainties and potential failures may exist. Moreover, RL introduces extra complexity into the proof. To address these issues, we may first pose assumptions, such as by simplifying the problem, or by starting with linear cases. Furthermore, we can leverage the theory from robust and stochastic MPC, fault detection, fault-tolerant control, and theoretical results of the latest MPC-RL studies [63], [64].

BIBLIOGRAPHY

[1] K. Aboudolas, M. Papageorgiou, A. Kouvelas, and E. Kosmatopoulos, “A rolling-horizon quadratic-programming approach to the signal control problem in large-scale congested urban road networks”, *Transportation Research Part C: Emerging Technologies*, vol. 18, no. 5, pp. 680–694, 2010.

[2] V. Adetola, D. DeHaan, and M. Guay, “Adaptive model predictive control for constrained nonlinear systems”, *Systems & Control Letters*, vol. 58, no. 5, pp. 320–326, 2009.

[3] A. Afram and F. Janabi-Sharifi, “Theory and applications of HVAC control systems–a review of model predictive control (MPC)”, *Building and Environment*, vol. 72, pp. 343–355, 2014.

[4] F. Ahmadizar, K. Soltanian, F. Akhlaghian Tab, and I. Tsoulos, “Artificial neural network development by means of a novel combination of grammatical evolution and genetic algorithm”, *Engineering Applications of Artificial Intelligence*, vol. 39, pp. 1–13, 2015.

[5] V. A. Akpan and G. D. Hassapis, “Nonlinear model identification and adaptive model predictive control using neural networks”, *ISA transactions*, vol. 50, no. 2, pp. 177–194, 2011.

[6] A. Alessio and A. Bemporad, “A survey on explicit model predictive control”, in *Nonlinear Model Predictive Control: Towards New Challenging Applications*, L. Magni, Ed., Berlin Heidelberg: Springer-Verlag, 2009, pp. 345–369.

[7] J. Arroyo, C. Manna, F. Spiessens, and L. Helsen, “Reinforced model predictive control (rl-mpc) for building energy management”, *Applied Energy*, vol. 309, p. 118 346, 2022.

[8] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, “Deep reinforcement learning: A brief survey”, *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

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

[10] L. D. Baskar, B. De Schutter, and H. Hellendoorn, “Traffic management for automated highway systems using model-based predictive control”, *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 2, pp. 838–847, 2012.

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

[12] F. Belletti, D. Haziza, G. Gomes, and A. M. Bayen, “Expert level control of ramp metering based on multi-task deep reinforcement learning”, *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 4, pp. 1198–1207, 2017.

[13] A. Bemporad, “Model predictive control design: New trends and tools”, in *45th IEEE Conference on Decision and Control*, 2006, pp. 6678–6683.

[14] A. Bemporad and M. Morari, “Robust model predictive control: A survey”, in *Robustness in Identification and Control*, Springer, 2007, pp. 207–226.

[15] P. T. Boggs and J. W. Tolle, “Sequential quadratic programming”, *Acta numerica*, vol. 4, pp. 1–51, 1995.

[16] A. Brabazon, “Grammatical-Evolution in Finance and Economics: A Survey”, in *Handbook of Grammatical Evolution*, Springer, 2018, pp. 263–288.

[17] A. Brandi, A. Ferrara, S. Sacone, S. Siri, C. Vivas, and F. Rubio, “Model predictive control with state estimation for freeway systems”, in *American Control Conference (ACC)*, May 2017, pp. 3536–3541.

[18] M. Brdys, M. Grochowski, T. Gminski, K. Konarczak, and M. Drewa, “Hierarchical predictive control of integrated wastewater treatment systems”, *Control Engineering Practice*, vol. 16, no. 6, pp. 751–767, 2008.

[19] F. D. Brunner, M. Lazar, and F. Allgöwer, “Stabilizing model predictive control: On the enlargement of the terminal set”, *International Journal of Robust and Nonlinear Control*, vol. 25, no. 15, pp. 2646–2670, 2015.

[20] L. Buşoniu, R. Babuška, and B. D. Schutter, “Multi-agent reinforcement learning: An overview”, *Innovations in Multi-Agent Systems and Applications-1*, pp. 183–221, 2010.

[21] R. Cagienard, P. Grieder, E. Kerrigan, and M. Morari, “Move blocking strategies in receding horizon control”, *Journal of Process Control*, vol. 17, no. 6, pp. 563–570, 2007.

[22] G. C. Calafiore and M. C. Campi, “The scenario approach to robust control design”, *IEEE Transactions on Automatic Control*, vol. 51, no. 5, pp. 742–753, 2006.

[23] E. F. Camacho and C. Bordons, *Model Predictive Control in Process Industry*. Springer, 1995.

[24] E. F. Camacho and C. B. Alba, *Model Predictive Control*. Springer Science & Business Media, 2013.

[25] P. Chanfreut, J. M. Maestre, and E. F. Camacho, “Coalitional model predictive control on freeways traffic networks”, *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 11, pp. 6772–6783, 2020.

[26] T. Chu, J. Wang, L. Codecà, and Z. Li, “Multi-agent deep reinforcement learning for large-scale traffic signal control”, *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 3, pp. 1086–1095, 2019.

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

[28] C. F. Daganzo, “The cell transmission model, part ii: Network traffic”, *Transportation Research Part B: Methodological*, vol. 29, no. 2, pp. 79–93, 1995.

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

[30] B. De Schutter, “Model predictive traffic control for green mobility”, in *2014 European Control Conference*, Jun. 2014, pp. 2260–2263.

[31] B. De Schutter and B. De Moor, “Optimal traffic light control for a single intersection”, *European Journal of Control*, vol. 4, no. 3, pp. 260–276, 1998.

[32] F. Deng, J. Jin, Y. Shen, and Y. Du, “Advanced self-improving ramp metering algorithm based on multi-agent deep reinforcement learning”, in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, IEEE, 2019, pp. 3804–3809.

[33] S. Di Cairano, T. Bäthge, and R. Findeisen, “Modular design for constrained control of actuator-plant cascades”, in *2019 American Control Conference (ACC)*, IEEE, 2019, pp. 1755–1760.

[34] M. Diehl, H. G. Bock, H. Diedam, and P.-B. Wieber, “Fast direct multiple shooting algorithms for optimal robot control”, in *Fast motions in biomechanics and robotics*, Springer, 2006, pp. 65–93.

[35] A. Downs, “The law of peak-hour expressway congestion”, *Traffic Quarterly*, vol. 16, no. 3, 1962.

[36] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, *et al.*, “Challenges of real-world reinforcement learning: Definitions, benchmarks and analysis”, *Machine Learning*, vol. 110, no. 9, pp. 2419–2468, 2021.

[37] W. Dunham, B. Hencey, A. R. Girard, and I. Kolmanovsky, “Distributed model predictive control for more electric aircraft subsystems operating at multiple time scales”, *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2177–2190, 2019.

[38] ECA special report 23/2018, “Air pollution: Our health still insufficiently protected”, 2018.

[39] L. Echeverrıa, J. I. Giménez-Nadal, and J. A. Molina, “Who uses green mobility? exploring profiles in developed countries”, *Transportation Research Part A: Policy and Practice*, vol. 163, pp. 247–265, 2022.

[40] A. N. Elmachtoub and P. Grigas, “Smart "predict, then optimize"”, *INFORMS*, 2021.

[41] M. Eom and B.-I. Kim, “The traffic signal control problem for intersections: A review”, *European Transport Research Review*, vol. 12, pp. 1–20, 2020.

[42] D. Ernst, M. Glavic, F. Capitanescu, and L. Wehenkel, “Reinforcement learning versus model predictive control: A comparison on a power system problem”, *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 2, pp. 517–529, 2008.

[43] European Commission, *Regulation (EU) 2021/1119 of the european parliament and of the council of 30 june 2021 establishing the framework for achieving climate neutrality and amending regulations (EC) no 401/2009 and (EU) 2018/1999 ('European Climate Law')*, 2021.

[44] European Court of Auditors, "Urban mobility in the EU", 2019.

[45] European Economic Area, *Air quality e-reporting*, 2021.

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

[47] M. Fenton, J. McDermott, D. Fagan, S. Forstenlechner, E. Hemberg, and M. O'Neill, "Ponyge2: Grammatical evolution in python", in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2017, pp. 1194–1201.

[48] A. Ferrara, G. P. Incremona, and G. Piacentini, "A hierarchical mpc and sliding mode based two-level control for freeway traffic systems with partial demand information", *European Journal of Control*, vol. 59, pp. 152–164, 2021.

[49] A. Ferrara, A. N. Oleari, S. Sacone, and S. Siri, "Freeways as systems of systems: A distributed model predictive control scheme", *IEEE Systems Journal*, vol. 9, no. 1, pp. 312–323, 2014.

[50] A. Ferrara, S. Sacone, and S. Siri, "Event-triggered model predictive schemes for freeway traffic control", *Transportation Research Part C: Emerging Technologies*, vol. 58, pp. 554–567, 2015.

[51] C. A. Floudas and P. M. Pardalos, "State of the art in global optimization: Computational methods and applications", 2013.

[52] J. R. D. Frejo, A. Núñez, B. De Schutter, and E. F. Camacho, "Hybrid model predictive control for freeway traffic using discrete speed limit signals", *Transportation Research Part C: Emerging Technologies*, vol. 46, pp. 309–325, 2014.

[53] J. R. D. Frejo and E. F. Camacho, "Global versus local mpc algorithms in freeway traffic control with ramp metering and variable speed limits", *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1556–1565, 2012.

[54] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods", in *International conference on machine learning*, PMLR, 2018, pp. 1587–1596.

[55] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration", in *International conference on machine learning*, PMLR, 2019, pp. 2052–2062.

[56] N. H. Gartner, "OPAC: A demand-responsive strategy for traffic signal control", *Transportation Research Record*, vol. 906, pp. 75–84, 1983.

[57] A. H. Ghods, L. Fu, and A. Rahimi-Kian, "An efficient optimization approach to real-time coordinated and integrated freeway traffic control", *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 4, pp. 873–884, 2010.

[58] G. Gomes, R. Horowitz, A. A. Kurzhanskiy, P. Varaiya, and J. Kwon, "Behavior of the cell transmission model and effectiveness of ramp metering", *Transportation Research Part C: Emerging Technologies*, vol. 16, no. 4, pp. 485–513, 2008.

[59] D. Görges, "Relations between model predictive control and reinforcement learning", *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 4920–4928, 2017.

[60] P. Goulart, E. Kerrigan, and J. Maciejowski, "Optimization over state feedback policies for robust control with constraints", *Automatica*, vol. 42, no. 4, pp. 523–533, 2006.

[61] M. Gregurić, K. Kušić, and E. Ivanjko, "Impact of deep reinforcement learning on variable speed limit strategies in connected vehicles environments", *Engineering Applications of Artificial Intelligence*, vol. 112, p. 104 850, 2022.

[62] M. Gregurić, K. Kušić, F. Vrbanić, and E. Ivanjko, "Variable speed limit control based on deep reinforcement learning: A possible implementation", in *2020 International Symposium ELMAR*, IEEE, 2020, pp. 67–72.

[63] S. Gros and M. Zanon, "Learning for MPC with stability & safety guarantees", *Automatica*, vol. 146, p. 110 598, 2022.

[64] S. Gros and M. Zanon, "Data-driven economic nmpc using reinforcement learning", *IEEE Transactions on Automatic Control*, vol. 65, no. 2, pp. 636–648, 2019.

[65] T. Haarnoja, A. Zhou, K. Hartikainen, *et al.*, "Soft actor-critic algorithms and applications", *arXiv preprint arXiv:1812.05905*, 2018.

[66] J. Haddad, M. Ramezani, and N. Geroliminis, "Cooperative traffic control of a mixed network with two urban regions and a freeway", *Transportation Research Part B: Methodological*, vol. 54, pp. 17–36, 2013.

[67] H.-G. Han, S.-J. Fu, H.-Y. Sun, and J.-F. Qiao, "Hierarchical nonlinear model predictive control with multi-time-scale for wastewater treatment process", *Journal of Process Control*, vol. 108, pp. 125–135, 2021.

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

[69] Y. Han, M. Wang, Z. He, Z. Li, H. Wang, and P. Liu, "A linear lagrangian model predictive controller of macro-and micro-variable speed limits to eliminate freeway jam waves", *Transportation Research Part C: Emerging Technologies*, vol. 128, p. 103 121, 2021.

[70] Y. Han, M. Wang, L. Li, C. Roncoli, J. Gao, and P. Liu, "A physics-informed reinforcement learning-based strategy for local and coordinated ramp metering", *Transportation Research Part C: Emerging Technologies*, vol. 137, p. 103 584, 2022.

[71] N. Hansen, X. Wang, and H. Su, "Temporal difference learning for model predictive control", *arXiv preprint arXiv:2203.04955*, 2022.

[72] A. Haydari and Y. Yilmaz, "Deep reinforcement learning for intelligent transportation systems: A survey", *IEEE Transactions on Intelligent Transportation Systems*, 2020.

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

[74] A. Hegyi, B. De Schutter, and J. Hellendoorn, “Optimal coordination of variable speed limits to suppress shock waves”, *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 1, pp. 102–112, 2005.

[75] T. A. N. Heirung, B. E. Ydstie, and B. Foss, “Dual adaptive model predictive control”, *Automatica*, vol. 80, pp. 340–348, 2017.

[76] E. Hemberg, L. Ho, M. O’Neill, and H. Claussen, “A comparison of grammatical genetic programming grammars for controlling femtocell network coverage”, *Genetic Programming and Evolvable Machines*, vol. 14, no. 1, pp. 65–93, 2013.

[77] J. Henry, J. Farges, and J. Tuffal, “The PRODYN real time traffic algorithm”, in *Control in Transportation Systems*, Elsevier, 1984, pp. 305–310.

[78] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, “Learning-based model predictive control: Toward safe learning in control”, *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, pp. 269–296, 2020.

[79] S. Hoogendoorn, J. Van Kooten, and R. Adams, “Lessons learned from field operational test of integrated network management in amsterdam”, *Transportation Research Record*, vol. 2554, no. 1, pp. 111–119, 2016.

[80] J. Hopcroft, R. Motwani, and J. Ullman, *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 3rd Edition, 2006.

[81] R. Horowitz, A. May, A. Skabardonis, *et al.*, “Design, field implementation and evaluation of adaptive ramp metering algorithms”, *California PATH Research Report UCB-ITS-PRR-2005-2*, 2005.

[82] M. M. Hosseini, L. Rodriguez-Garcia, and M. Parvania, “Hierarchical combination of deep reinforcement learning and quadratic programming for distribution system restoration”, *IEEE Transactions on Sustainable Energy*, vol. 14, no. 2, pp. 1088–1098, 2023.

[83] J. Hu, Q. Wang, Y. Ye, and Y. Tang, “Toward online power system model identification: A deep reinforcement learning approach”, *IEEE Transactions on Power Systems*, 2022.

[84] P. Hunt, D. Robertson, R. Bretherton, and M. Royle, “The SCOOT on-line traffic signal optimisation technique”, *Traffic Engineering & Control*, vol. 23, no. 4, pp. 190–192, 1982.

[85] A. Jamshidnejad, D. Sun, A. Ferrara, and B. De Schutter, “A novel bi-level temporally-distributed MPC: An application for persistent green urban mobility”, *Transportation Research Part C: Emerging Technologies*, 2023.

[86] A. Jamshidnejad, I. Papamichail, H. Hellendoorn, M. Papageorgiou, and B. D. Schutter, “Gradient-based model-predictive control for green urban mobility in traffic networks”, in *2016 IEEE 19th International Conference on Intelligent Transportation Systems*, Nov. 2016, pp. 1077–1082.

[87] A. Jamshidnejad, I. Papamichail, M. Papageorgiou, and B. De Schutter, "Sustainable model-predictive control in urban traffic networks: Efficient solution based on general smoothening methods", *IEEE Transactions on Control Systems Technology*, vol. 26, no. 3, pp. 813–827, 2017.

[88] A. Jamshidnejad, S. Lin, Y. Xi, and B. De Schutter, "Corrections to "Integrated urban traffic control for the reduction of travel delays and emissions"", *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 5, pp. 1978–1983, 2018.

[89] J. Jeschke and B. De Schutter, "Parametrized model predictive control approaches for urban traffic networks", *IFAC-PapersOnLine*, vol. 54, no. 2, pp. 284–291, 2021.

[90] J. Jeschke, D. Sun, A. Jamshidnejad, and B. De Schutter, "Grammatical-evolution-based parameterized model predictive control for urban traffic networks", *Control Engineering Practice*, vol. 132, p. 105 431, 2023.

[91] X. Jin, T. Jiang, Y. Mu, *et al.*, "Scheduling distributed energy resources and smart buildings of a microgrid via multi-time scale and model predictive control method", *IET Renewable Power Generation*, vol. 13, no. 6, pp. 816–833, 2019.

[92] T. Johannink, S. Bahl, A. Nair, *et al.*, "Residual reinforcement learning for robot control", in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 6023–6029.

[93] S. Kachroudi and S. Mammar, "The effects of the control and prediction horizons on the urban traffic regulation", in *13th International IEEE Conference on Intelligent Transportation Systems*, IEEE, 2013.

[94] P. Karamanakos, T. Geyer, and R. Kennel, "A computationally efficient model predictive control strategy for linear systems with integer inputs", *IEEE Transactions on Control Systems Technology*, vol. 24, no. 4, pp. 1463–1471, 2015.

[95] M. Keyvan-Ekbatani, X. Gao, V. V. Gayah, and V. L. Knoop, "Traffic-responsive signals combined with perimeter control: Investigating the benefits", *Transportmetrica B: Transport Dynamics*, vol. 7, no. 1, pp. 1402–1425, 2019.

[96] B.-Y. Kim and H.-S. Ahn, "Distributed coordination and control for a freeway traffic network using consensus algorithms", *IEEE Systems Journal*, vol. 10, no. 1, pp. 162–168, 2014.

[97] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization", *arXiv preprint arXiv:1412.6980*, 2014.

[98] J. Köhler, P. Kötting, R. Soloperto, F. Allgöwer, and M. A. Müller, "A robust adaptive model predictive control framework for nonlinear uncertain systems", *International Journal of Robust and Nonlinear Control*, vol. 31, no. 18, pp. 8725–8749, 2021.

[99] Q.-J. Kong, Y. Xu, S. Lin, D. Wen, F. Zhu, and Y. Liu, "UTN-model-based traffic flow prediction for parallel-transportation management systems", *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1541–1547, Sep. 2013.

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

[101] A. Kotsialos, M. Papageorgiou, and A. Messner, "Integrated optimal control of motorway traffic networks", in *Proceedings of the 1999 American Control Conference (Cat. No. 99CH36251)*, IEEE, vol. 3, 1999, pp. 2183–2187.

[102] A. Kotsialos, M. Papageorgiou, C. Diakaki, Y. Pavlis, and F. Middelham, "Traffic flow modeling of large-scale motorway networks using the macroscopic modeling tool METANET", *IEEE Transactions on Intelligent Transportation Systems*, vol. 3, no. 4, pp. 282–292, 2002.

[103] A. Kotsialos, M. Papageorgiou, M. Mangeas, and H. Haj-Salem, "Coordinated and integrated control of motorway networks via non-linear optimal control", *Transportation Research Part C: Emerging Technologies*, vol. 10, no. 1, pp. 65–84, 2002.

[104] J. Koza, "Genetic programming as a means for programming computers by natural selection", *Statistics and Computing*, vol. 4, no. 2, pp. 87–112, 1994.

[105] S. N. Kumpati, P. Kannan, *et al.*, "Identification and control of dynamical systems using neural networks", *IEEE Transactions on neural networks*, vol. 1, no. 1, pp. 4–27, 1990.

[106] W. Langson, I. Chryssochoos, S. Raković, and D. Q. Mayne, "Robust model predictive control using tubes", *Automatica*, vol. 40, no. 1, pp. 125–133, 2004.

[107] C. Lee, B. Hellinga, and F. Saccomanno, "Evaluation of variable speed limits to improve traffic safety", *Transportation Research Part C: Emerging Technologies*, vol. 14, no. 3, pp. 213–228, 2006.

[108] Y. Li, K. Hua, and Y. Cao, "Using stochastic programming to train neural network approximation of nonlinear MPC laws", *Automatica*, vol. 146, p. 110 665, 2022.

[109] Z. Li, P. Liu, C. Xu, H. Duan, and W. Wang, "Reinforcement learning-based variable speed limit control strategy to reduce traffic congestion at freeway recurrent bottlenecks", *IEEE transactions on intelligent transportation systems*, vol. 18, no. 11, pp. 3204–3217, 2017.

[110] M. J. Lighthill and G. B. Whitham, "On kinematic waves ii. a theory of traffic flow on long crowded roads", *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, vol. 229, no. 1178, pp. 317–345, 1955.

[111] T. P. Lillicrap, J. J. Hunt, A. Pritzel, *et al.*, "Continuous control with deep reinforcement learning", *arXiv preprint arXiv:1509.02971*, 2015.

[112] S. Lin, B. De Schutter, Y. Xi, and H. Hellendoorn, "Efficient network-wide model-based predictive control for urban traffic networks", *Transportation Research Part C: Emerging Technologies*, vol. 24, pp. 122–140, 2012.

[113] 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.

[114] S. Lin, B. De Schutter, Y. Xi, and H. Hellendoorn, “Fast model predictive control for urban road networks via milp”, *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 3, pp. 846–856, 2011.

[115] S. Lin, B. De Schutter, Y. Xi, and H. Hellendoorn, “Integrated urban traffic control for the reduction of travel delays and emissions”, *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1609–1619, 2013.

[116] J. Little, M. Kelson, and N. Gartner, “MAXBAND: A versatile program for setting signals on arteries and triangular networks”, *Sloan School of Management, Massachusetts Institute of Technology*, 1981.

[117] D. Liu, J. Wu, H. Liu, P. Song, and K. Wang, “Multi-time scale energy management strategy of micro energy grid based on model predictive control”, in *2020 IEEE Sustainable Power and Energy Conference (iSPEC)*, IEEE, 2020, pp. 1511–1516.

[118] S. Liu, A. Sadowska, and B. De Schutter, “A scenario-based distributed model predictive control approach for freeway networks”, *Transportation Research Part C: Emerging Technologies*, vol. 136, p. 103 261, 2022.

[119] J. Lofberg, “Approximations of closed-loop minimax MPC”, in *42nd IEEE International Conference on Decision and Control*, IEEE, vol. 2, 2003, pp. 1438–1442.

[120] J. de Lope, D. Maravall, *et al.*, “Robust high performance reinforcement learning through weighted k-nearest neighbors”, *Neurocomputing*, vol. 74, no. 8, pp. 1251–1259, 2011.

[121] P. A. Lopez, M. Behrisch, L. Bieker-Walz, *et al.*, “Microscopic traffic simulation using SUMO”, in *IEEE Intelligent Transportation Systems Conference (ITSC)*, vol. 21, Maui, Hawaii: IEEE, 2018, pp. 2575–2582, ISBN: 9781728103235.

[122] M. Lorenzen, F. Allgöwer, and M. Cannon, “Adaptive model predictive control with robust constraint satisfaction”, *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 3313–3318, 2017.

[123] C. Lu, J. Huang, L. Deng, and J. Gong, “Coordinated ramp metering with equity consideration using reinforcement learning”, *Journal of Transportation Engineering, Part A: Systems*, vol. 143, no. 7, p. 04 017 028, 2017.

[124] J. Maciejowski, *Predictive Control with Constraints*. Prentice Hall, 2002.

[125] J. M. Maestre, R. R. Negenborn, *et al.*, *Distributed Model Predictive Control Made Easy*. Springer, 2014, vol. 69.

[126] D. Manolis, T. Pappa, C. Diakaki, I. Papamichail, and M. Papageorgiou, “Centralised versus decentralised signal control of large-scale urban road networks in real time: A simulation study”, *IET Intelligent Transport Systems*, vol. 12, no. 8, pp. 891–900, 2018.

[127] A. Marco, P. Hennig, J. Bohg, S. Schaal, and S. Trimpe, “Automatic LQR tuning based on gaussian process global optimization”, in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2016, pp. 270–277.

[128] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, “Constrained model predictive control: Stability and optimality”, *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

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

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

[131] C. A. Micchelli, *Interpolation of scattered data: distance matrices and conditionally positive definite functions*. Springer, 1984.

[132] P. Mirchandani and L. Head, “A real-time traffic signal control system: Architecture, algorithms, and analysis”, *Transportation Research Part C: Emerging Technologies*, vol. 9, no. 6, pp. 415–432, 2001.

[133] V. Mnih, A. P. Badia, M. Mirza, *et al.*, “Asynchronous methods for deep reinforcement learning”, in *International conference on machine learning*, PMLR, 2016, pp. 1928–1937.

[134] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, “Playing atari with deep reinforcement learning”, *arXiv preprint arXiv:1312.5602*, 2013.

[135] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, “Human-level control through deep reinforcement learning”, *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[136] M. Morari and J. H. Lee, “Model predictive control: Past, present and future”, *Computers & Chemical Engineering*, vol. 23, no. 4-5, pp. 667–682, 1999.

[137] A. Muralidharan and R. Horowitz, “Computationally efficient model predictive control of freeway networks”, *Transportation Research Part C: Emerging Technologies*, vol. 58, pp. 532–553, 2015.

[138] R. R. Negenborn, B. De Schutter, M. A. Wiering, and H. Hellendoorn, “Learning-based model predictive control for markov decision processes”, *IFAC Proceedings Volumes*, vol. 38, no. 1, pp. 354–359, 2005.

[139] M. Nicolau and A. Agapitos, “Understanding grammatical evolution: Grammar design”, in *Handbook of Grammatical Evolution*, Springer, 2018, pp. 23–53.

[140] M. Nicolau, M. O’Neill, and A. Brabazon, “Termination in grammatical evolution: Grammar design, wrapping, and tails”, in *IEEE Congress on Evolutionary Computation*, 2012, pp. 1–8.

[141] M. O’Neill and C. Ryan, “Grammatical evolution”, *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 4, pp. 349–358, 2001.

[142] A. N. Oleari, J. R. D. Frejo, E. F. Camacho, and A. Ferrara, “A model predictive control scheme for freeway traffic systems based on the classification and regression trees methodology”, in *2015 European Control Conference (ECC)*, IEEE, 2015, pp. 3459–3464.

[143] L. B. D. Oliveira and E. Camponogara, “Multi-agent model predictive control of signaling split in urban traffic networks”, *Transportation Research Part C: Emerging Technologies*, vol. 18, no. 1, pp. 120–139, 2010.

[144] I. Outlook, *Itf transport outlook 2017*, 2017.

[145] G. Pannocchia, J. B. Rawlings, and S. J. Wright, “Conditions under which suboptimal nonlinear MPC is inherently robust”, *Systems & Control Letters*, vol. 60, no. 9, pp. 747–755, 2011.

[146] M. Papageorgiou, H. Hadj-Salem, J.-M. Blosseville, *et al.*, “Alinea: A local feedback control law for on-ramp metering”, *Transportation Research Record*, vol. 1320, no. 1, pp. 58–67, 1991.

[147] M. Papageorgiou, E. Kosmatopoulos, and I. Papamichail, “Effects of variable speed limits on motorway traffic flow”, *Transportation Research Record*, vol. 2047, no. 1, pp. 37–48, 2008.

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

[149] G. J. Pappas, G. Lafferriere, and S. Sastry, “Hierarchically consistent control systems”, *IEEE Transactions on Automatic Control*, vol. 45, no. 6, pp. 1144–1160, 2000.

[150] J. Park and I. W. Sandberg, “Universal approximation using radial-basis-function networks”, *Neural Computation*, vol. 3, no. 2, pp. 246–257, 1991.

[151] A. Perrusquıa and W. Yu, “Identification and optimal control of nonlinear systems using recurrent neural networks and reinforcement learning: An overview”, *Neurocomputing*, vol. 438, pp. 145–154, 2021.

[152] L. E. Peterson, “K-nearest neighbor”, *Scholarpedia*, vol. 4, no. 2, p. 1883, 2009.

[153] D. Piga, M. Forgione, S. Formentin, and A. Bemporad, “Performance-oriented model learning for data-driven mpc design”, *IEEE control systems letters*, vol. 3, no. 3, pp. 577–582, 2019.

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

[155] R. Poli, W. B. Langdon, N. F. McPhee, and J. Koza, *A Field Guide to Genetic Programming*. Lulu.com, 2008.

[156] S. Qin and T. Badgwell, “A survey of industrial model predictive control technology”, *Control Engineering Practice*, vol. 11, no. 7, pp. 733–764, 2003.

[157] S. V. Raković, B. Kouvaritakis, R. Findeisen, and M. Cannon, “Homothetic tube model predictive control”, *Automatica*, vol. 48, no. 8, pp. 1631–1638, 2012.

[158] J. Rawlings and D. Mayne, *Model Predictive Control: Theory and Design*. Nob Hill Publishing, 2009.

[159] B. Recht, “A tour of reinforcement learning: The view from continuous control”, *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, pp. 253–279, 2019.

[160] J. Reilly and A. M. Bayen, “Distributed optimization for shared state systems: Applications to decentralized freeway control via subnetwork splitting”, *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 6, pp. 3465–3472, 2015.

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

[162] S. Richter, C. N. Jones, and M. Morari, "Computational complexity certification for real-time mpc with input constraints based on the fast gradient method", *IEEE Transactions on Automatic Control*, vol. 57, no. 6, pp. 1391–1403, 2012.

[163] D. Robertson, "Research on the TRANSYT and SCOOT methods of signal coordination", *ITE Journal*, vol. 56, no. 1, pp. 36–40, 1986.

[164] R. P. Roess, E. S. Prassas, and W. R. McShane, *Traffic engineering*. Pearson/Prentice Hall, 2004.

[165] C. Roncoli, I. Papamichail, and M. Papageorgiou, "Hierarchical model predictive control for multi-lane motorways in presence of vehicle automation and communication systems", *Transportation Research Part C: Emerging Technologies*, vol. 62, pp. 117–132, 2016.

[166] U. Rosolia and F. Borrelli, "Learning model predictive control for iterative tasks: A data-driven control framework", *IEEE Transactions on Automatic Control*, vol. 63, no. 7, pp. 1883–1896, 2017.

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

[168] C. Ryan, M. O'Neill, and J. Collins, "Introduction to 20 years of grammatical evolution", in *Handbook of Grammatical Evolution*, Springer, 2018, pp. 1–21.

[169] R. Scattolini, "Architectures for distributed and hierarchical model predictive control–a review", *Journal of Process Control*, vol. 19, no. 5, pp. 723–731, 2009.

[170] T. Schmidt-Dumont and J. H. van Vuuren, "Decentralised reinforcement learning for ramp metering and variable speed limits on highways", *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 8, p. 1, 2015.

[171] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms", *arXiv preprint arXiv:1707.06347*, 2017.

[172] A. Sims, "The Sydney coordinated adaptive traffic system", in *Engineering Foundation Conference on Research Directions in Computer Control of Urban Traffic Systems*, 1979.

[173] S. Siri, C. Pasquale, S. Sacone, and A. Ferrara, "Freeway traffic control: A survey", *Automatica*, vol. 130, p. 109 655, 2021.

[174] J. Skaf, S. Boyd, and A. Zeevi, "Shrinking-horizon dynamic programming", *International Journal of Robust and Nonlinear Control*, vol. 20, no. 17, pp. 1993–2002, 2010.

[175] Z. Su, A. Jamshidi, A. Núñez, S. Baldi, and B. De Schutter, "Multi-level condition-based maintenance planning for railway infrastructures–a scenario-based chance-constrained approach", *Transportation Research Part C: Emerging Technologies*, vol. 84, pp. 92–123, 2017.

[176] Z. Su, A. Jamshidi, A. Núñez, S. Baldi, and B. De Schutter, “Integrated condition-based track maintenance planning and crew scheduling of railway networks”, *Transportation Research Part C: Emerging Technologies*, vol. 105, pp. 359–384, 2019.

[177] D. Sun, A. Jamshidnejad, and B. De Schutter, “Adaptive parameterized control for coordinated traffic management using reinforcement learning”, in *IFAC World Congress*, 2023.

[178] D. Sun, A. Jamshidnejad, and B. De Schutter, “A novel framework combining MPC and deep reinforcement learning with application to freeway traffic control”, *IEEE Transactions on Intelligent Transportation Systems*, under review, 2023.

[179] D. Sun, A. Jamshidnejad, and B. De Schutter, “Adaptive parameterized model predictive control based on reinforcement learning: A synthesis framework”, *Engineering Applications of Artificial Intelligence*, under review, 2023.

[180] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[181] M. Tanaskovic, L. Fagiano, and V. Gligorovski, “Adaptive model predictive control for linear time varying MIMO systems”, *Automatica*, vol. 105, pp. 237–245, 2019.

[182] T. Tettamanti, T. Luspay, B. Kulcsar, T. Péni, and I. Varga, “Robust control for urban road traffic networks”, *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 1, pp. 385–398, 2013.

[183] T. Tettamanti, I. Varga, B. Kulcsár, and J. Bokor, “Model predictive control in urban traffic network management”, in *2008 16th Mediterranean Conference on Control and Automation*, IEEE, 2008, pp. 1538–1543.

[184] R. Tóth, *Modeling and identification of linear parameter-varying systems*. Berlin, Heidelberg: Springer–Verlag, 2010.

[185] I. Tsoulos, D. Gavrilis, and E. Glavas, “Neural network construction and training using grammatical evolution”, *Neurocomputing*, vol. 72, no. 1-3, pp. 269–277, 2008.

[186] G. E. Uhlenbeck and L. S. Ornstein, “On the theory of the brownian motion”, *Physical review*, vol. 36, no. 5, p. 823, 1930.

[187] U. UNEP, “Emissions gap report 2020”, *UN environment programme*, 2020.

[188] E. Van Henten and J. Bontsema, “Time-scale decomposition of an optimal control problem in greenhouse climate management”, *Control Engineering Practice*, vol. 17, no. 1, pp. 88–96, 2009.

[189] P. Varaiya, “Smart cars on smart roads: Problems of control”, *IEEE Transactions on Automatic Control*, vol. 38, no. 2, pp. 195–207, 1993.

[190] E. Walraven, M. T. Spaan, and B. Bakker, “Traffic flow optimization: A reinforcement learning approach”, *Engineering Applications of Artificial Intelligence*, vol. 52, pp. 203–212, 2016.

[191] C. Wang, Y. Xu, J. Zhang, and B. Ran, "Integrated traffic control for freeway recurrent bottleneck based on deep reinforcement learning", *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[192] C. Wang, J. Zhang, L. Xu, L. Li, and B. Ran, "A new solution for freeway congestion: Cooperative speed limit control using distributed reinforcement learning", *IEEE Access*, vol. 7, pp. 41 947–41 957, 2019.

[193] S. Wang, R. Diao, C. Xu, D. Shi, and Z. Wang, "On multi-event co-calibration of dynamic model parameters using soft actor-critic", *IEEE Transactions on Power Systems*, vol. 36, no. 1, pp. 521–524, 2020.

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

[195] F. Webster, "Traffic signal settings", Road Research Laboratory, Tech. Rep., 1958.

[196] G. S. van de Weg, A. Hegyi, S. P. Hoogendoorn, and B. De Schutter, "Efficient freeway mpc by parameterization of alinea and a speed-limited area", *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 1, pp. 16–29, 2018.

[197] G. S. van de Weg, H. L. Vu, A. Hegyi, and S. P. Hoogendoorn, "A hierarchical control framework for coordination of intersection signal timings in all traffic regimes", *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 5, pp. 1815–1827, 2018.

[198] A. Wegener, M. Piórkowski, M. Raya, H. Hellbrück, S. Fischer, and J. Hubaux, "TraCI: An interface for coupling road traffic and network simulators", in *Proceedings of the 11th communications and networking simulation symposium*, 2008, pp. 155–163.

[199] M. A. Wiering *et al.*, "Multi-agent reinforcement learning for traffic light control", in *Machine Learning: Proceedings of the Seventeenth International Conference (ICML'2000)*, 2000, pp. 1151–1158.

[200] N. Wu, D. Li, Y. Xi, and B. D. Schutter, "Distributed event-triggered model predictive control for urban traffic lights", *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 8, pp. 4975–4985, 2020.

[201] Y. Wu, H. Tan, L. Qin, and B. Ran, "Differential variable speed limits control for freeway recurrent bottlenecks via deep actor-critic algorithm", *Transportation Research Part C: Emerging Technologies*, vol. 117, p. 102 649, 2020.

[202] H. Xie, X. Xu, Y. Li, W. Hong, and J. Shi, "Model predictive control guided reinforcement learning control scheme", in *2020 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2020, pp. 1–8.

[203] B. Ye, W. Wu, L. Li, and W. Mao, "A hierarchical model predictive control approach for signal splits optimization in large-scale urban road networks", *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 8, pp. 2182–2192, 2016.

[204] B. Ye, W. Wu, and W. Mao, "Distributed model predictive control method for optimal coordination of signal splits in urban traffic networks", *Asian Journal of Control*, vol. 17, no. 3, pp. 775–790, 2015.

[205] B. Ye, W. Wu, K. Ruan, *et al.*, “A survey of model predictive control methods for traffic signal control”, *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 3, pp. 623–640, 2019.

[206] M. Zanon and S. Gros, “Safe reinforcement learning using robust mpc”, *IEEE Transactions on Automatic Control*, vol. 66, no. 8, pp. 3638–3652, 2020.

[207] S. Zegeye, B. De Schutter, H. Hellendoorn, E. Breunesse, and A. Hegyi, “A predictive traffic controller for sustainable mobility using parameterized control policies”, *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 3, pp. 1420–1429, 2012.

[208] S. Zegeye, B. D. Schutter, J. Hellendoorn, E. Breunesse, and A. Hegyi, “Integrated macroscopic traffic flow, emission, and fuel consumption model for control purposes”, *Transportation Research Part C: Emerging Technologies*, vol. 31, pp. 158–171, Jun. 2013.

[209] H. Zhang, S. Li, and Y. Zheng, “Q-learning-based model predictive control for nonlinear continuous-time systems”, *Industrial & Engineering Chemistry Research*, vol. 59, no. 40, pp. 17 987–17 999, 2020.

[210] K. Zhang and Y. Shi, “Adaptive model predictive control for a class of constrained linear systems with parametric uncertainties”, *Automatica*, vol. 117, p. 108 974, 2020.

[211] 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.

[212] M. Zhong, M. Johnson, Y. Tassa, T. Erez, and E. Todorov, “Value function approximation and model predictive control”, in *2013 IEEE symposium on adaptive dynamic programming and reinforcement learning (ADPRL)*, IEEE, 2013, pp. 100–107.

[213] Y. Zhou, K. Ozbay, P. Kachroo, and F. Zuo, “Ramp metering for a distant downstream bottleneck using reinforcement learning with value function approximation”, *Journal of Advanced Transportation*, vol. 2020, 2020.

[214] F. Zhu and S. V. Ukkusuri, “Accounting for dynamic speed limit control in a stochastic traffic environment: A reinforcement learning approach”, *Transportation research part C: emerging technologies*, vol. 41, pp. 30–47, 2014.

[215] A. K. Ziliaskopoulos, “A linear programming model for the single destination system optimum dynamic traffic assignment problem”, *Transportation Science*, vol. 34, no. 1, pp. 37–49, 2000.

CURRICULUM VITÆ

Dingshan Sun was born on August 21, 1994 in Xiantao, Hubei Province, China. He obtained his B.Sc. degree from Northeastern University, China, in 2015, and obtained his M.Sc. degree from Shanghai Jiao Tong University, China, in 2018. Since March 2019, he has been pursuing his Ph.D. degree in Delft Center for Systems and Control, Delft University of Technology. In his Ph.D. project, he worked on multi-level and learning-based model predictive control for traffic management under the supervision of Prof. dr. ir. Bart De Schutter and Dr. Anahita Jamshidnejad. His research interests include (parameterized) model predictive control, reinforcement learning, learning-based control, and multi-level control with applications to control of urban and freeway traffic networks.

TRAIL THESIS SERIES

The following list contains the most recent dissertations in the TRAIL Thesis Series. For a complete overview of more than 275 titles see the TRAIL website: www.rsTRAIL.nl.
The TRAIL Thesis Series is a series of the Netherlands TRAIL Research School on transport, infrastructure and logistics.

Sun, D., *Multi-level and Learning-based Model Predictive Control for Traffic Management*, T2023/16, October 2023, TRAIL Thesis Series, the Netherlands

Brederode, L.J.N., *Incorporating Congestion Phenomena into Large Scale Strategic Transport Model Systems*, T2023/15, October 2023, TRAIL Thesis Series, the Netherlands

Hernandez, J.I., *Data-driven Methods to study Individual Choice Behaviour: with applications to discrete choice experiments and Participatory Value Evaluation experiments*, T2023/14, October 2023, TRAIL Thesis Series, the Netherlands

Aoun, J., *Impact Assessment of Train-Centric Rail Signaling Technologies*, T2023/13, October 2023, TRAIL Thesis Series, the Netherlands

Pot, F.J., *The Extra Mile: Perceived accessibility in rural areas*, T2023/12, September 2023, TRAIL Thesis Series, the Netherlands

Nikghadam, S., *Cooperation between Vessel Service Providers for Port Call Performance Improvement*, T2023/11, July 2023, TRAIL Thesis Series, the Netherlands

Li, M., *Towards Closed-loop Maintenance Logistics for Offshore Wind Farms: Approaches for strategic and tactical decision-making*, T2023/10, July 2023, TRAIL Thesis Series, the Netherlands

Berg, T. van den, *Moral Values, Behaviour, and the Self: An empirical and conceptual analysis*, T2023/9, May 2023, TRAIL Thesis Series, the Netherlands

Shelat, S., *Route Choice Behaviour under Uncertainty in Public Transport Networks: Stated and revealed preference analyses*, T2023/8, June 2023, TRAIL Thesis Series, the Netherlands

Zhang, Y., *Flexible, Dynamic, and Collaborative Synchromodal Transport Planning Considering Preferences*, T2023/7, June 2023, TRAIL Thesis Series, the Netherlands

Kapetanović, M., *Improving Environmental Sustainability of Regional Railway Services*, T2023/6, June 2023, TRAIL Thesis Series, the Netherlands

Li, G., *Uncertainty Quantification and Predictability Analysis for Traffic Forecasting at Multiple Scales*, T2023/5, April 2023, TRAIL Thesis Series, the Netherlands

Harter, C., *Vulnerability through Vertical Collaboration in Transportation: A complex networks approach*, T2023/4, March 2023, TRAIL Thesis Series, the Netherlands

Razmi Rad, S., *Design and Evaluation of Dedicated Lanes for Connected and Automated Vehicles*, T2023/3, March 2023, TRAIL Thesis Series, the Netherlands

Eikenbroek, O., *Variations in Urban Traffic*, T2023/2, February 2023, TRAIL Thesis Series, the Netherlands

Wang, S., *Modeling Urban Automated Mobility on-Demand Systems: an Agent-Based Approach*, T2023/1, January 2023, TRAIL Thesis Series, the Netherlands

Szép, T., *Identifying Moral Antecedents of Decision-Making in Discrete Choice Models*, T2022/18, December 2022, TRAIL Thesis Series, the Netherlands

Zhou, Y., *Ship Behavior in Ports and Waterways: An empirical perspective*, T2022/17, December 2022, TRAIL Thesis Series, the Netherlands

Yan, Y., *Wear Behaviour of A Convex Pattern Surface for Bulk Handling Equipment*, T2022/16, December 2022, TRAIL Thesis Series, the Netherlands

Giudici, A., *Cooperation, Reliability, and Matching in Inland Freight Transport*, T2022/15, December 2022, TRAIL Thesis Series, the Netherlands

Nadi Najafabadi, A., *Data-Driven Modelling of Routing and Scheduling in Freight Transport*, T2022/14, October 2022, TRAIL Thesis Series, the Netherlands

Heuvel, J. van den, *Mind Your Passenger! The passenger capacity of platforms at railway stations in the Netherlands*, T2022/13, October 2022, TRAIL Thesis Series, the Netherlands

Haas, M. de, *Longitudinal Studies in Travel Behaviour Research*, T2022/12, October 2022, TRAIL Thesis Series, the Netherlands

Dixit, M., *Transit Performance Assessment and Route Choice Modelling Using Smart Card Data*, T2022/11, October 2022, TRAIL Thesis Series, the Netherlands

Du, Z., *Cooperative Control of Autonomous Multi-Vessel Systems for Floating Object Manipulation*, T2022/10, September 2022, TRAIL Thesis Series, the Netherlands

Larsen, R.B., *Real-time Co-planning in Synchromodal Transport Networks using Model Predictive Control*, T2022/9, September 2022, TRAIL Thesis Series, the Netherlands

Zeinaly, Y., *Model-based Control of Large-scale Baggage Handling Systems: Leveraging the theory of linear positive systems for robust scalable control design*, T2022/8, June 2022, TRAIL Thesis Series, the Netherlands

Fahim, P.B.M., *The Future of Ports in the Physical Internet*, T2022/7, May 2022, TRAIL Thesis Series, the Netherlands

Huang, B., *Assessing Reference Dependence in Travel Choice Behaviour*, T2022/6, May 2022, TRAIL Thesis Series, the Netherlands