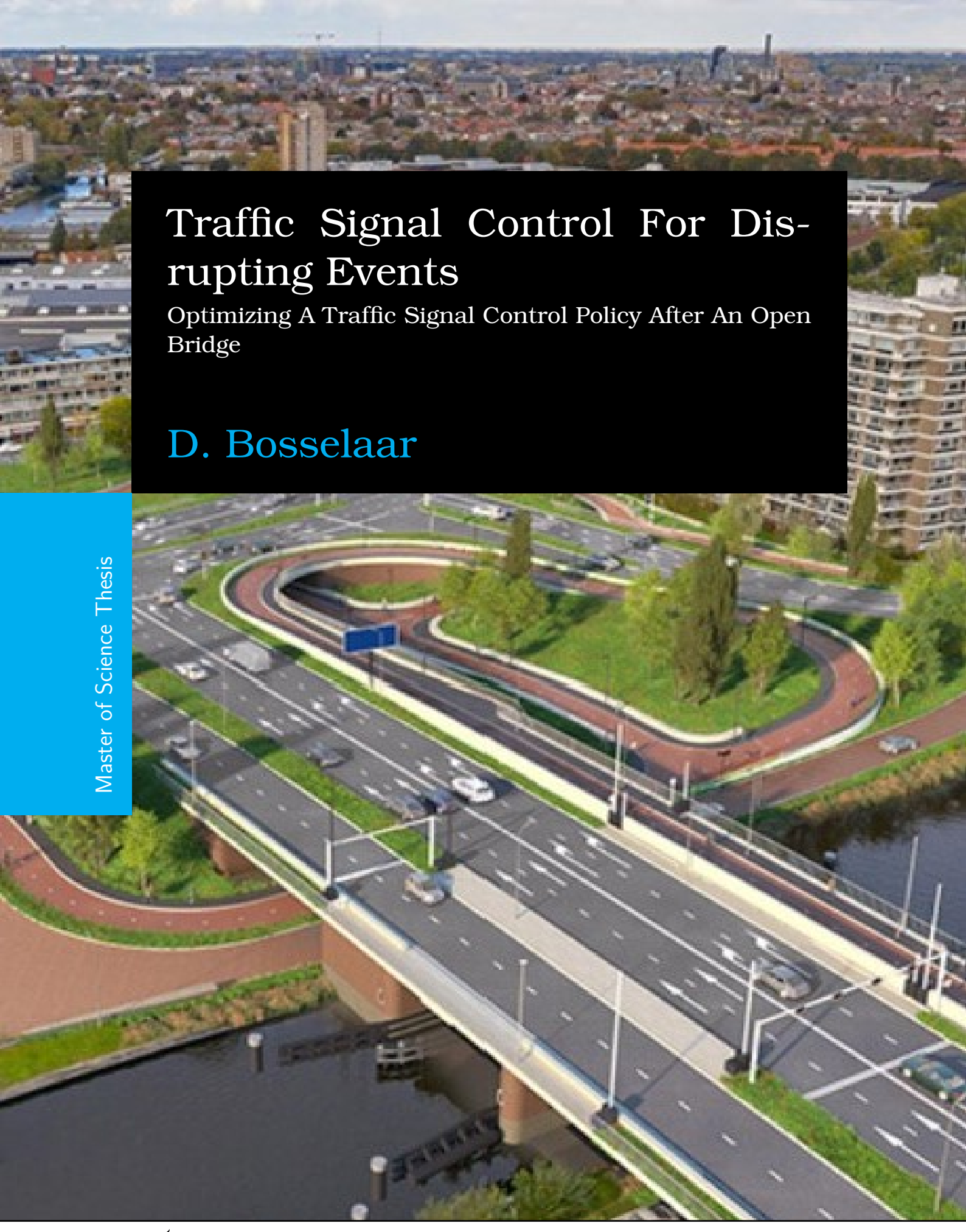

Traffic Signal Control For Disrupting Events

Optimizing A Traffic Signal Control Policy After An Open Bridge

D. Bosselaar

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Traffic Signal Control For Disrupting Events

Optimizing A Traffic Signal Control Policy After An Open Bridge

Master of Science Thesis

D. Bosselaar

July 20, 2024

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

The work in this thesis was supported by 4cast B.V. Their cooperation is hereby gratefully acknowledged.

Abstract

Traffic Signal Control (TSC) in urban areas is typically performed by (adaptive) cycle-based methods. These methods are characterized by their robustness and simplicity, but do not hold the potential to achieve optimal control. Model Predictive Control (MPC) is a control method that has proven to be a great solution for urban TSC in literature. MPC is a model-based control method that predicts and optimizes future states over a moving prediction horizon. This approach holds the potential to foresee events that disrupt traffic networks, and actively adjust signal behavior to anticipate these disruptions.

This thesis research investigates an MPC approach for urban TSC, where a bridge opening occurs multiple times a day at a busy junction in Leiden, Netherlands. A linear prediction model is used with linear inequality constraints and binary control inputs. Multiple scenarios are simulated, where the MPC controller is compared to a baseline controller developed in the COCON optimization tool. The first scenario is in regular traffic conditions, so no bridge opening. In other simulations, a 5-minute bridge opening is incorporated. Finally, an upstream intersection is included to manipulate upstream traffic flow.

The simulation results show a decrease in average delay time per road user of 20% for normal traffic conditions. When the bridge opening was included, a delay decrease of 25% was achieved. When the upstream intersection was included, a delay decrease of 41% was achieved, using a centralized approach.

Table of Contents

List of Figures

List of Tables

List of acronyms

TSC	Traffic Signal Control
MPC	Model Predictive Control
LP	Linear Programming
QP	Quadratic Programming
SQP	Sequential Quadratic Programming
ILP	Integer Linear Programming
NP	Nondeterministic Polynomial-time
NN	Neural Network
VAF	Variance Accounted For

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Azita Dabiri, for her assistance throughout my thesis project. Her critical questions and constructive feedback during our regular meetings were very valuable in guiding me in the right direction.

I am also grateful to my supervisor Mark van Raaij, from 4cast B.V, for his guidance during my thesis. His feedback and hands-on tutoring in VISSIM have been very valuable to my development and thesis process.

I extend my thanks to my colleagues at 4cast, Dick, Jeroen, Jos, Peter, Riemer, and Wilco, who were always willing to assist me on the project and made my workdays enjoyable, particularly during our Fussball matches.

Lastly, I would like to thank my brothers, Chris and Stijn, and roommates, Dennis, Kasper, Rogier, Thom, Tom, and Tom, for their support. They provided valuable feedback on the chapters I wrote, and offered great suggestions for my research.

Delft, University of Technology — D. Bosselaar
July 20, 2024

Chapter 1

Introduction

The increase in population size, with now 8 billion people living on earth [12], combined with urbanization, leads to a significant rise in the amount of motorized vehicles in urban areas [17]. This naturally causes elevated levels of congestion. Traffic congestion has negative impacts on social, economic, and environmental aspects. Shocking statistics from Inrix reveal the extent of the problem, with the Netherlands, the United States, and the United Kingdom experiencing average annual delay times of 45, 51, and 80 hours per road user, respectively [23]. This leads to substantial monetary losses, exceeding $1,000 per year on fuel cost and time losses (approximately evenly distributed). In the UK, the figure approaches a staggering $2,000 per driver [23]. In terms of the environment, there is considerable room for improvement. According to a study by Nagurney et al. [20], approximately 15% of all carbon dioxide emissions originate from transportation, along with about 90% of carbon monoxide emissions. Another major consequence of congestion is the increased frequency of accidents [26], which leads to further traffic jams and potentially poses a risk of physical harm to passengers. To address traffic congestion in urban areas, three main strategies can be employed. Firstly, incorporating additional (redundant) features into the network, such as a bypass, can effectively manage increased traffic volumes. Downsides are that this approach involves significant financial costs and requires space, which is often rare in urban areas. Secondly, road designers can look into improving the current infrastructure, such as rearranging the network or introducing bus lanes. This is much cheaper than the previously mentioned strategy, and does not require space. Thirdly, there is a method called traffic control, which will be the primary focus of this study. It is a powerful tool that enhances traffic flow by manipulation of traffic signals [15]. No redundancy is required, or improvement of the current infrastructure. With the knowledge of historic and/or real-time traffic data, traffic signals are manipulated by a virtual agent, with the ultimate goal to improve traffic flow.

The history of TSC traces back to the groundbreaking work of Webster and his team in 1958, as documented in their seminal paper [32]. This early research focused on developing one of the earliest single intersection signal controllers, aiming to minimize vehicle waiting time through offline optimization. The green times were based on historical traffic data. Over the years, TSC strategies evolved, with the introduction of a strategy known as the

'GreenWave method' [31]. This is a more modern fixed-time control approach that excels in networks with substantial traffic flow volumes, moving in a single direction across multiple intersections. It is a powerful control strategy but the GreenWave method's drawback lies in its lack of robustness to fluctuating traffic volumes, potentially leading to congestion in scenarios with varying flow rates. This drawback was overcome by the introduction of traffic responsive, or real-time, controllers like SCOOT and SCATS [11, 28]. Nowadays, they are the most widely used TSC methods in the world, and are known for their simplicity and robustness. Historic data is used to determine green times and cycle duration. Based on real-time traffic data, these controllers adjust green times within the cycle to further enhance traffic flow, and reduce congestion. SCOOT typically controls a single intersection, whereas SCATS is used for a network of junctions.

SCOOT and SCATS are excellent real-time control methods due to their simplicity and robustness, but they do not typically achieve optimal control. Traffic-responsive control uses current traffic data and online optimization to determine signal policies. Processing real-time data and computing signal policies both take time, which is why SCOOT and SCATS are popular approaches. Their simplicity allows for very fast online computation. Other real-time control strategies like OPAC, RHODYN, and RHODES [7, 10, 18] incorporate mathematical traffic models to predict future system states. These model-based control approaches face a trade-off between accuracy and simplicity, as the model needs to be sufficiently accurate without compromising online computation time. Balancing these factors is essential for effective model-based real-time traffic control.

MPC stands as a well established model-based real-time control method, widely applied across various fields [22]. In the realm of urban traffic control, MPC has gained attention and has been explored by many researchers, as exemplified by studies such as [21, 13], where they model traffic flow, after which an optimization problem is formulated. The application of MPC in urban traffic control aims to mitigate congestion, showcasing its adaptability in addressing complex challenges within dynamic traffic environments. By using predictive models, MPC considers future system states to optimize control decisions whilst staying within system input and state boundaries (constraints). The utilization of MPC in urban traffic management reflects a growing interest in advanced, model-based approaches that prioritize predictive capabilities for more effective real-time control.

The model choice involves significant trade-offs between accuracy and computational complexity. For example, one can choose either a microscopic or macroscopic model. Microscopic models delve into individual vehicle behavior, providing very high accuracy, but increasing complexity as the model accounts for each specific vehicle in the network. The alternative is a macroscopic model. A macroscopic model operates with traffic flows across network links, sacrificing some model accuracy, but maintaining simplicity due to the low amount of system states. Another critical trade-off is the choice between a linear and nonlinear model. Linear models may compromise prediction accuracy in the face of highly nonlinear traffic behavior [4]. However, if a suitable cost function and set of constraints are chosen, linear models allow for more efficient algorithms during online optimization, reducing computational complexity for MPC-controllers. Nonlinear urban traffic models, while typically more intricate, accommodate complex traffic scenarios, such as the time needed for vehicles to travel between intersections, yielding more accurate state predictions and potentially enhancing control performance. The selection of an appropriate model depends on the specific needs and objectives of the urban traffic control system. Examples of linear models are the 1-state link model [1],

and the link transmission model [30]. The 1-state link model assumes linear lane outflow during a green signal, independent of the saturation rate of the link. It is a very simple model but might lose some accuracy at fluctuating saturation rates. The link transmission model considers multiple lane saturation rates, which enhances accuracy. However, this comes with the cost of more computation time. Depending on both the size of the system, and the complexity, one can make the trade-off between the linear models, that both have obtained good results in a very realistic microscopic simulation environment. If the traffic system is small, or very powerful hardware is available for the optimization step, one might even opt for a nonlinear model, which has the potential to be more accurate than linear alternatives. This is done in e.g. [13], where they consider nonlinear system dynamics and linear constraints on the inputs and states.

Real-time control using a prediction model holds the potential for achieving great performance, particularly when anticipating and responding to known disruptive events. By incorporating predictive models that account for various factors such as traffic patterns, congestion, and potential disruptions, the control system gains the ability to proactively adjust signal timings and traffic flow strategies. This proactive approach allows for efficient management of traffic, minimizing delays, and optimizing overall system performance. The use of prediction models enables the system to respond swiftly and adapt to changing conditions, contributing to enhanced traffic flow and reduced disruptions. Nevertheless, an MPC approach for known disrupting event is hardly explored in literature. This thesis report explores an MPC approach for urban traffic control where a beforehand known disruptive event occurs. The disruptive event is a bridge opening, which temporarily blocks multiple lanes at a very busy junction.

1-1 Problem statement

MPC has shown promising results in the field of urban traffic control. The predictive behaviour of the approach makes it very suitable for problems in which known disruptive events occur. This thesis report will investigate an MPC approach for urban traffic control where disruptive events occur. The main goal is to reduce delay times for road users, which translates in less congestion, in a specific urban area in Leiden, Netherlands, where a bridge opening disrupts traffic flow several times a day (also during rush hours). This goal will be reached by answering the following research question:

"How can control strategies mitigate traffic congestion in urban areas, experiencing frequent beforehand known disruptive events"

The exact urban traffic network with the bridge is described in Chapter 3. The following sub-questions are answered in order to eventually reach the goal of the research:

1. **Model choice**: Which urban traffic model is both accurate, and computationally efficient for representing the dynamics of the system?

2. **Cost function and constraints**: What cost function, standard constraints, and network-specific constraints should be incorporated into the framework of the control problem?

3. **Model validation and controller testing**: How can simulations in a microscopic simulation environment be utilized to validate the model, and tune weights after setting up the controller to the model?

1-2 Thesis outline

The paper is organized as follows. In Chapter 2, an overview of urban TSC is provided, highlighting how MPC can address anticipated disruptions. Chapter 3 outlines the problem formulation and the control approach used in the case study. The results are presented and discussed in Chapter 4. Finally, conclusions are drawn, that are shared in Chapter 5, where also recommendations are provided for future work.

Chapter 2

Urban traffic modeling and control in abnormal situations

Urban traffic control using MPC requires the incorporation of a system model. Describing traffic dynamics in a model is challenging due to its nonlinear and stochastic nature, particularly in urban settings. Various models of differing accuracy and complexity have been proposed in literature, that have shown promising results. In general, a higher level of accuracy in the model yields better future state predictions, but at the expense of increased computation time during the optimization step. Notable urban traffic models include the 1-state link model [1], the link transmission model [30], and the mixed logical dynamical system [14]. These models have increasing complexity respectively. The 1-state link model assumes linear lane dynamics, and only one state is considered for each lane (representing the number of queued vehicles in that specific lane). The link transmission model, while also linear, introduces three distinct lane saturation rates and multiple states per lane. This potentially enhances accuracy, but at the cost of computation time. The mixed logical dynamical system is the most complex of the three, and represents a nonlinear model that potentially offers greater accuracy than the aforementioned linear models, but at the expense of requiring a nonlinear optimization algorithm. These algorithms are typically slower than linear alternatives.

When employing MPC for addressing abnormal events like a bridge opening, having a large prediction horizon proves advantageous, enabling the controller to anticipate earlier to such events [3]. However, the drawback of an increased horizon lies in the increased amount of decision variables to optimize. This means there are more input combinations to be explored, leading to larger computation times. Consequently, in systems dealing with scenarios like a bridge opening, utilizing a simple model is preferred. This choice is driven by the model's ability to efficiently handle the extended horizon through efficient optimization calculations. In the case study in Chapter 3, a bridge opening is implemented during simulation. Therefore, a linear model is chosen to be used in this research, the 1-state link model. It has proven to be a very suitable model in [1], where urban traffic control, using MPC, is executed in a very realistic microscopic simulation environment called SUMO [16].

This chapter is organized as follows. In Section 2-1, the 1-state link model is introduced. An in depth explanation is provided and how it can be used for traffic modeling and control. Section 2-2 describes the MPC control scheme, followed by the specific optimization problem in combination with the system 1-state link model. Finally, in Section 2-3, abnormal traffic situation control is discussed and how MPC can be used for these disrupted systems.

2-1 Urban traffic modeling using the 1-state link model

The linear 1-state link model is characterized by the fact that only one state is considered for each link, which is the number of queued vehicles. In the microscopic simulation environment utilized for system calibration and controller testing, obtaining the exact count of queued vehicles in a specific lane is not feasible. However, it is possible to gather data on queue length for each individual lane. Consequently, the only state taken into account is queue length, which essentially represents the same information, though in a different unit of measurement. A vehicle is considered to be queued up when its velocity drops below $10km/h$. When it reaches a velocity above $30km/h$, it is not considered to be queued anymore.

The control inputs are the traffic signals. For lane i, $u_i \in \{0, 1\}$, where the inputs represent a red and green signal, respectively. Amber (or yellow) signals are given when the control input changes from 1 to 0. The control input is multiplied by the lane outflow rate α_i. At last, upstream system lanes have inflow or 'demand', d_i, entering the lane at the edge of the junction. Therefore, the general lane dynamics of any lane i entering the junction can be summarized as in Equation (2-1). In the equation, m represents the amount of lanes entering the junction, or incoming lanes. T_s is the sampling time.

$$x_i(k+1) = x_i(k) + \alpha_i T_s u_i(k) + d_i T_s \quad i \in \{1, \dots, m\}. \tag{2-1}$$

A traffic network typically consist of multiple intersections. This means not every lane in a traffic system is an upstream (or incoming) lane. Most networks have several 'midstream' lanes. An example junction is provided in Figure 2-1, where lanes 1-6 are 'upstream' and lanes 7-8 are 'midstream'. These midstream lanes have inflow which is often a portion of upstream lane outflow. For example, lane i is provided a green signal and has outflow of which some vehicles turn towards midstream lane j with turn fraction p_{ij}. Then the lane dynamics of j can be summarized in Equation (2-2). In the equation, n represents the amount of midstream lanes at the junction.

$$x_j(k+1) = x_j(k) + \alpha_j T_s u_j(k) - p_{ij}\alpha_i T_s u_i(k) \quad j \in \{1, \dots, n\}. \tag{2-2}$$

All variables considered in the linear lane dynamic equations are summarized in the listing below. An important note to be made is that the outflow α in both equations is a negative value, whereas the demand d is a positive value. This is important for model calibration later on, as these values need to be constrained.

- u_i [-]: Control input for lane i, where $u_i \in \{0, 1\}$.
- α_i [m/s]: Lane outflow for lane i.

- d_i [m/s]: Inflow or 'demand' entering lane i.
- $x_i(k)$ [m]: Queue length in lane i at timestep k
- p_{ij} [-]: Turn fraction from lane i towards midstream lane j.

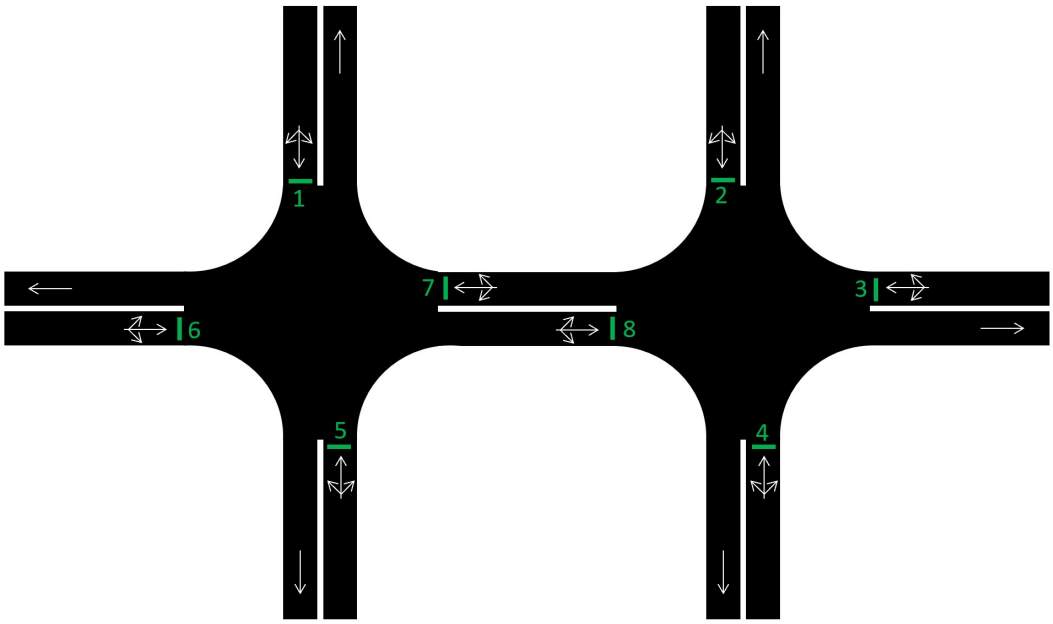

Figure 2-1: Example junction with upstream lanes 1-6 and midstream lanes 7-8.

The system can be written in standard linear form as shown in Equation (2-3). In the equation, A is the identity matrix. The input matrix B contains outflow for both upstream and midstream lanes on the diagonal, as well as inflow for midstream lanes on the non-diagonal entries. The demand of upstream lanes is added as a disturbance to the system, as it cannot be manipulated by control inputs. The matrix sizes of the system are $A \in \mathbb{R}^{(m+n)\times(m+n)}$ and $B \in \mathbb{R}^{(m+n)\times(m+n)}$, where m and n represent the amount of upstream and midstream lanes in the system respectively.

$$
\begin{aligned}
&\mathbf{x}(k+1) = A\mathbf{x}(k) + B\mathbf{u}(k) + \mathbf{d}(k),\\
&A = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 1 \end{bmatrix},\\
&B = \begin{bmatrix} \alpha_1 T_s & -p_{2,1}\alpha_2 T_s & \dots & -p_{m+n,1}\alpha_{m+n} T_s \\ -p_{1,2}\alpha_1 T_s & \alpha_2 T_s & \ddots & \vdots \\ \vdots & \ddots & \ddots & -p_{m+n,m+n-1}\alpha_{m+n} T_s \\ -p_{1,m+n}\alpha_1 T_s & \dots & -p_{m+n-1,m+n}\alpha_{m+n-1} T_s & \alpha_{n+m} T_s \end{bmatrix}
\end{aligned}
\tag{2-3}
$$

This summarizes the model dynamics of the 1-state link model. What makes the model unique are the constant outflow, independent of lane saturation, and the fact that only one state is considered for each lane.

2-2 Model predictive control

As mentioned in the introduction, according to literature, MPC has proven to be a very popular control method for urban TSC. The MPC scheme for any system is shown in Figure 2-2. It can be observed that multiple possibilities exist for e.g. input-output model, disturbance prediction, constraints, and sampling period [22]. Regardless of the parameters the designer chooses, on-line optimization is what ties them together and is therefore the common thread of MPC. A key feature of MPC is its moving horizon, where the optimization problem is continuously solved at each step, incorporating new measurements and updating predictions accordingly.

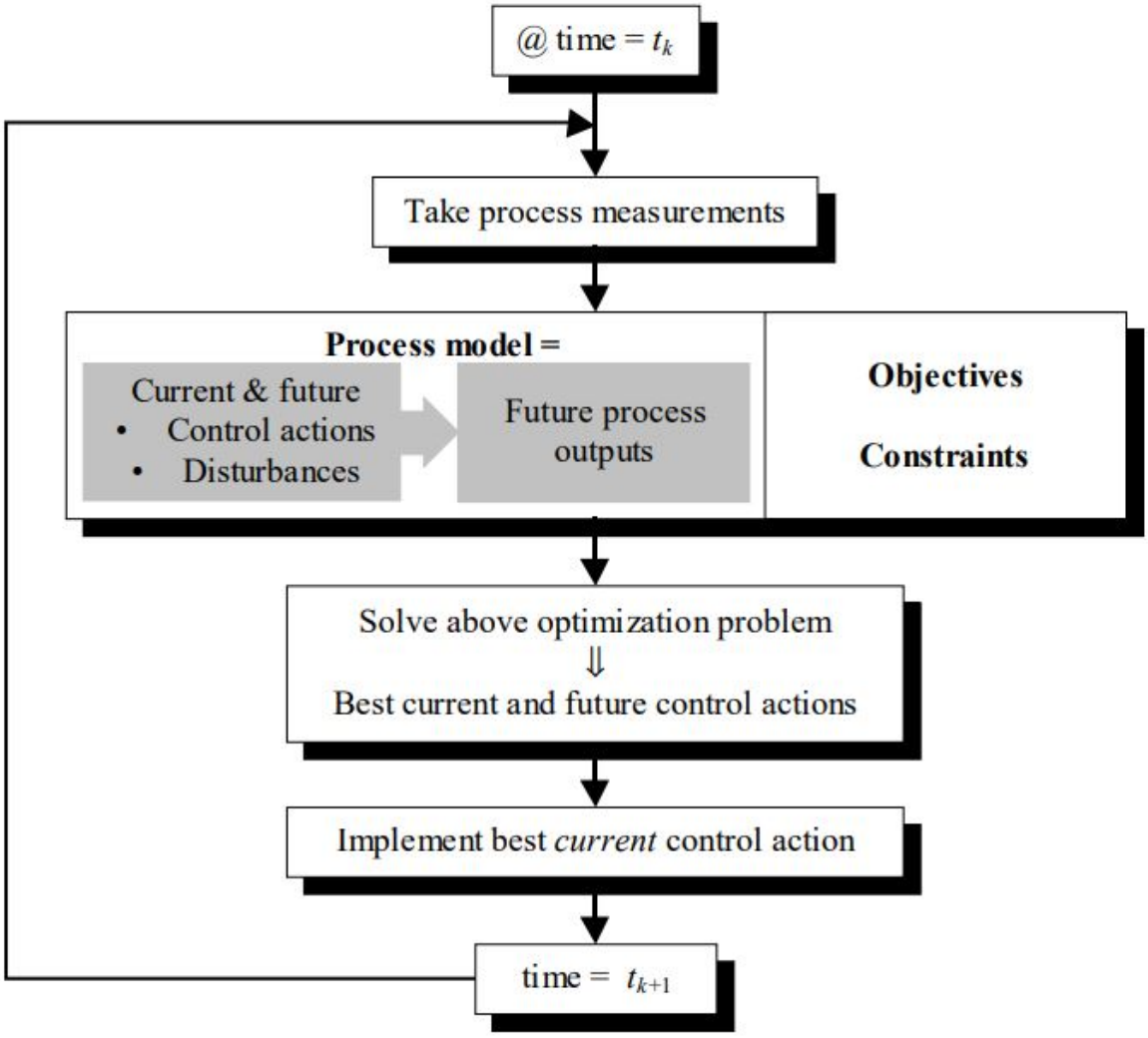

Figure 2-2: MPC control scheme [22]

The mathematical optimization problem of MPC can be formulated as in Equation (2-4). If the system has input or state constraints, one can add these to the optimization problem as desired. The overall goal is to minimize the cost function over a finite horizon, whilst staying within system and input bounds. The cost function is determined by the designer, based on the desired system behaviour.

$$\begin{aligned} \min_{u(0),\ldots,u(H-1)} \quad & \sum_{k=0}^{H-1} w_x^T x(k+1) + w_u^T u(k) \\ \text{subject to} \quad & \\ & x(k+1) = f(x(k), u(k)), \quad k = 0, \ldots, H-1, \\ & x(0) = x_{\text{measured}}, \\ & x(k+1) \in \mathcal{X}, \quad k = 0, \ldots, H-1, \\ & u(k) \in \mathcal{U}, \quad k = 0, \ldots, H-1, \end{aligned} \tag{2-4}$$

In Equation (2-4), H is the horizon, w_x and w_u are weights on states x and input u, respectively. Note that the weights and prediction horizon are design choices. Higher weights can be assigned to certain lanes to prioritize traffic flow where needed. For instance, public transport lanes often receive priority by assigning them larger weights.

Given that a system model: $x(k+1) = f(x(k), u(k))$ is available of the system, the state trajectory for any input sequence can be calculated. Depending on the linearity of the system dynamics and constraints, an optimization algorithm is used to minimize the overall cost. Examples of optimization algorithms are Linear Programming (LP) for linear objective functions and constraints, Quadratic Programming (QP) for quadratic cost functions and linear constraints, and more advanced but slower methods for nonlinear objective functions and constraints, such as Sequential Quadratic Programming (SQP).

2-2-1 Linear MPC for urban traffic control

The general concept of MPC is explained in Section 2-2. The model used for the case study is explained in Section 2-1. To achieve the ultimate goal of regulating urban traffic areas using MPC with the 1-state link prediction model, a cost function needs to be defined, as well as constraints and unknown model parameters. This subsection provides a structured overview on how this is done.

The cost of road users is typically expressed in terms of delay time. As one cannot directly control this parameter in the model, the cost is determined at cumulative queue length over all lanes, with differing weights dependent on the lane importance. Given horizon H, sampling time T_s, and lane weights w, the cost function can be formulated as in Equation (2-5).

$$J = \sum_{k=0}^{H-1} \sum_{i=1}^{m+n} w_i x_i(k+1). \tag{2-5}$$

The total cost of the optimization problem is the sum of all queued vehicles at every timestep within the prediction horizon. Weight assignment is an iterative process that is elaborated on in Chapter 4

The constraints for a traffic system consist of lane capacity and input restrictions. There is no such thing as negative queue lengths, hence intuitively $x_{min} = 0$ should be defined for every lane. In practice this is not entirely true, which is explained in Section 3-3-1. The midstream lanes have very strict maximum capacities. Otherwise, queued vehicles might block other routes on the intersection which potentially leads to a gridlock. The state constraints are defined as in (2-6a). Inputs are also constrained. The inputs for traffic signal controllers is binary, so either 0 or 1, as in (2-6b). Also, the sum of conflicting lane signals cannot exceed 1, which indicates only one green light is allowed at the same time as in (2-6c). The subscript c in u_c stands for 'conflicting', indicating a conflicting signal with respect to u_i. A so-called 'all red phase' is also introduced for conflicting lanes, which is a delay on the green time allowance to avoid collisions. These are shown in (2-6d) and (2-6e).

$$x_{\min,i} \leq x_i(k) \leq x_{\max,i} \tag{2-6a}$$
$$u_i(k) \in \{0, 1\} \tag{2-6b}$$
$$u_i(k) + u_c(k) \leq 1 \tag{2-6c}$$
$$u_i(k) + u_c(k-1) \leq 1 \tag{2-6d}$$
$$u_i(k-1) + u_c(k) \leq 1 \tag{2-6e}$$

For disturbed systems, extra input constraints should be added. Lanes blocked off by the bridge opening cannot receive a green signal, indicating $u_i \in \{0\}$.

As inputs are restricted to be binary values, Integer Linear Programming (ILP) is the most suitable optimization method for this problem. ILP theory is out of the scope of this research, but [27] is a book that explains the theory of ILP very extensively. Due to the integer inputs, an ILP problem is Nondeterministic Polynomial-time (NP)-hard. This indicates that there is no known polynomial-time algorithm that can solve all ILP problems optimally. Therefore, it can take quite some computation time, especially when the problem size grows. Some examples of efficient ILP problem solvers are GUROBI, CPLEX, or MOSEK [9, 2, 19].

2-2-2 MPC for TSC Summary

With the linear cost function, the linear inequality constraints, and the given fact that all inputs have to be integer values, the optimization step can be executed using an ILP solver. The optimal input sequence over the control horizon is calculated. Just the first control action is executed, after which the whole process starts over again at the next time-step. The updated measured state values now being $x(0)$.

Depending on the system size and prediction horizon, the optimization problem matrices can quickly become very large. For example, consider a hypothetical intersection with traffic entering from four directions, where each upstream link can turn in three directions (resulting in 12 signals). With a horizon of $H = 10$, there are over one hundred decision variables to optimize, creating a large number of input combinations for the solver to explore. The controller designer must find a good balance between performance and computation time.

2-3 Control of abnormal traffic situations

In the introduction, it was mentioned that traffic disruptions can potentially be managed effectively by using an MPC approach, due to its adaptability and predictive behaviour. However, it is not frequently proposed in the literature. One notable exception is [5], where the authors address the increasing frequency of incidents and the resulting severe congestion. Their prediction model, updated weekly with collected traffic data, is based on a Neural Network (NN).

In some other papers, MPC is used for disruption control in traffic like in [34] or [6]. However, these are small disturbances on the prediction model, and do not alter the complete dynamics of the system, which is the case for a bridge opening. This could be considered surprising as MPC has the property to incorporate beforehand known disruptions in its control policy and optimization step.

Potential abnormal traffic situations can be caused by, e.g., bridge openings, train crossings, funeral processions, emergency services, and many more. Real-time data can foresee these events happening at road junctions, and incorporating this knowledge in control policies can potentially lead to better traffic flow. In Chapter 4, a controller comparison is made where one controller incorporates the knowledge of a bridge opening beforehand and the other does not. Also, the horizon length is adjusted to make a fair comparison and draw well substantiated conclusions.

The MPC formulation for a traffic network disturbed by a bridge opening is an adjusted version of the one in (2-4). For the disturbed network, input restrictions are given for blocked lanes as $u(k) \in \{0\}$. This has to be respected for the entire bridge opening duration. Two sets are defined:

- $\mathcal{I}$: Set of all lanes directly blocked by the bridge
- $\mathcal{T}$: Set of time indices during which the bridge is open

Then the updated problem formulation for a traffic system with a bridge opening is as in (2-7).

$$
\begin{aligned}
\min_{u(0),\ldots,u(H-1)} \quad & \sum_{k=0}^{H-1} w_x^T x(k+1) + w_u^T u(k) \\
\text{subject to} \quad & \\
& x(k+1) = f(x(k), u(k)), \quad k = 0, \ldots, H-1, \\
& x(0) = x_{\text{measured}}, \\
& x(k+1) \in \mathcal{X}, \quad k = 0, \ldots, H-1, \\
& u(k) \in \mathcal{U}, \quad k = 0, \ldots, H-1, \\
& u_i(k) \in \{0\}, \quad \forall i \in \mathcal{I}, \forall k \in \mathcal{T},
\end{aligned}
\tag{2-7}
$$

Chapter 3

Case study

This chapter delves into the characteristics of the research's focal point: the particular junction under consideration and the approach adopted for control. It is organized as follows. In Section 3-1, the setup of the case study is explained, after which several challenges one encounters trying to control the Lammebrug junction are elaborated on. Section 3-2 describes how the model from Section 2-1 can be used for the Lammebrug junction. Section 3-3 explains the control strategy used to enhance traffic flow and reduce congestion at this junction, using the control framework explained in Chapter 2. Finally, the problem formulation is summarized in Section 3-4.

3-1 Problem set-up

The Lammebrug junction, situated at the southern edge of Leiden city, is a pivotal point for significant traffic volumes traveling to, from, or through Leiden. Especially in rush hours, the traffic load is massive and hard to process. The main road entering the junction is the provincial road called N206, and it is completely blocked off during a bridge opening. At the website of the province Zuid-Holland, the following is written [24] (translated from Dutch): "We regularly receive reports about the traffic jams at the Lammebrug. The N206 runs over the Lammebrug. The N206 is important as it connects various places in the region. Additionally, the N206 is one of the main access roads to Leiden. As a result, the Lammeschansplein, located directly behind the bridge, is a very busy junction." This quote shows the urgency of the problem and the desire to improve the network.

Before discussing the exact layout of the junction, it is important to first take a look at its location on the map. This is shown in figure Figure 3-1. The specific junction of interest is located in the middle of the red rectangle. One can observe multiple N-roads (Provincial roads), from different directions, entering the junction. Also, the junction connects the A4 and A44, which are the two highways in the area. Considering this, one can imagine the large amounts of traffic volume entering the junction, particularly during rush hours. Even without the bridge opening, congestion occurs daily, and the province of Zuid-Holland is working on

a large project to place an extra route, or bypass, to reduce the pressure in this area. This strongly emphasizes the importance of efficient traffic regulation at the Lammebrug junction.

Figure 3-1: Location of Lammebridge junction on map [8]

3-1-1 Junction layout

The Lammebrug junction is a pivot point in the area, which means proper traffic regulation is very important. In Figure 3-2, a schematic overview of the junction is provided. The junction is a double intersection and consists of a total of 24 lanes (white numbers). 17 of these (1-17) are entering the system from upstream and the other 7 lanes are so called 'midstream' (18-24). The midstream lanes are in essence connectors between both intersections. All traffic is regulated by 18 traffic signal installations (green), where the 18th signal always provides green, except during bridge opening.

Traffic enters the junction from six directions. Three of these directions represent over 90% of all traffic load entering the junction. They are indicated in the figure with 'North', 'South', and 'Leiden center'. As the names indicate, these are incoming lanes from the north (A44),

city center, and south (A4). Another important observation to be made is the bus lane that is controlled by signal head 4, and should have priority.

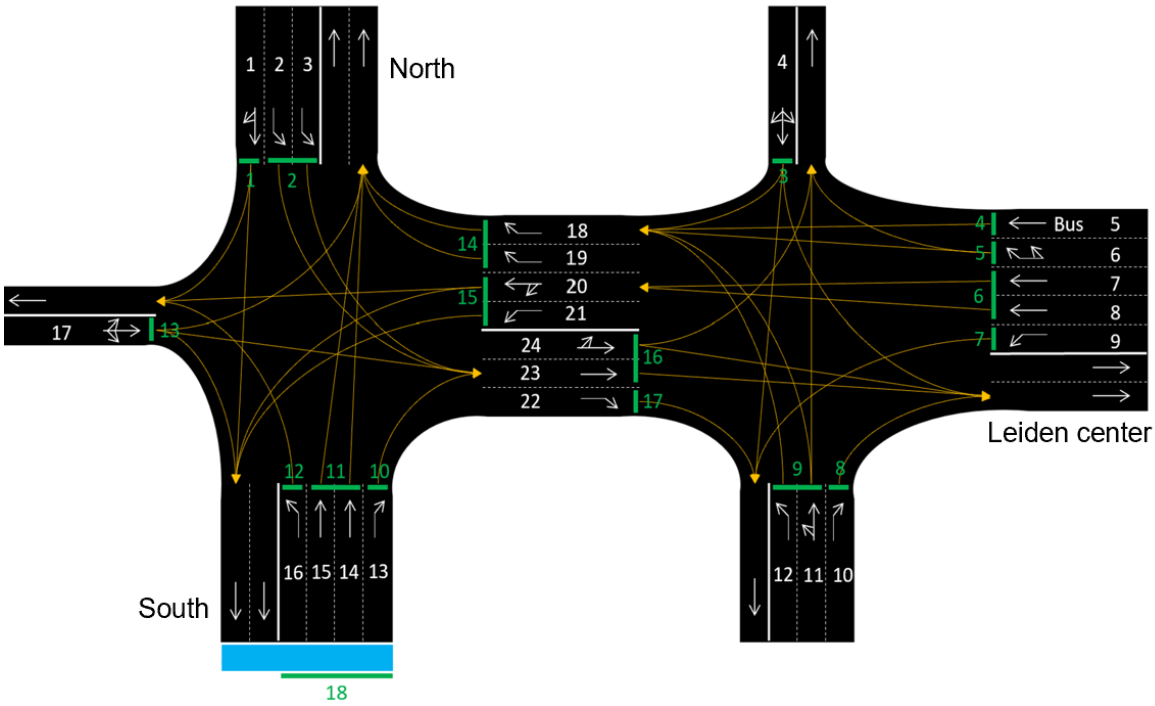

Figure 3-2: Schematic layout of the Lammebridge junction.

Already one can observe some challenges in controlling the system when looking at the junction layout. Lane 4 and 17, controlled by signal heads 3 and 13 respectively, require every other signal in their intersection to be red. This can lead to unwanted queues at busy moments during the day. Also, the midstream lanes have limited space. A typical cycle-based controller has no explicit way to fulfill constraints like this.

3-1-2 Surrounding areas

The junction highlighted in Figure 3-2 features two intersections. In this junction, outflow from the lanes controlled by e.g. signal head 2, are inflow of midstream lanes. At the southern exit/entrance in Figure 3-2, a traffic signal controlled intersection is located at about one kilometer upstream. It is located right after the highway (A4), which makes it undesirable to use these signal installations to stall traffic there, as it will spill back to cause jams at the A4. Another major traffic signal controlled intersection is located upstream of the northern exit/entrance of the Lammebrug junction. The location is shown in Figure 3-3 in the orange rectangle. One can imagine a lot of traffic enters this intersection from Katwijk, Wassenaar, or Voorschoten. The junction layout is shown in Figure 3-4, with its traffic signals that are labeled 19-27. Including this intersection in the control problem has the potential to be beneficial using either a decentralized of centralized approach. This potential benefit has been investigated in Chapter 4.

Figure 3-3: Location of upstream intersection on map [8]

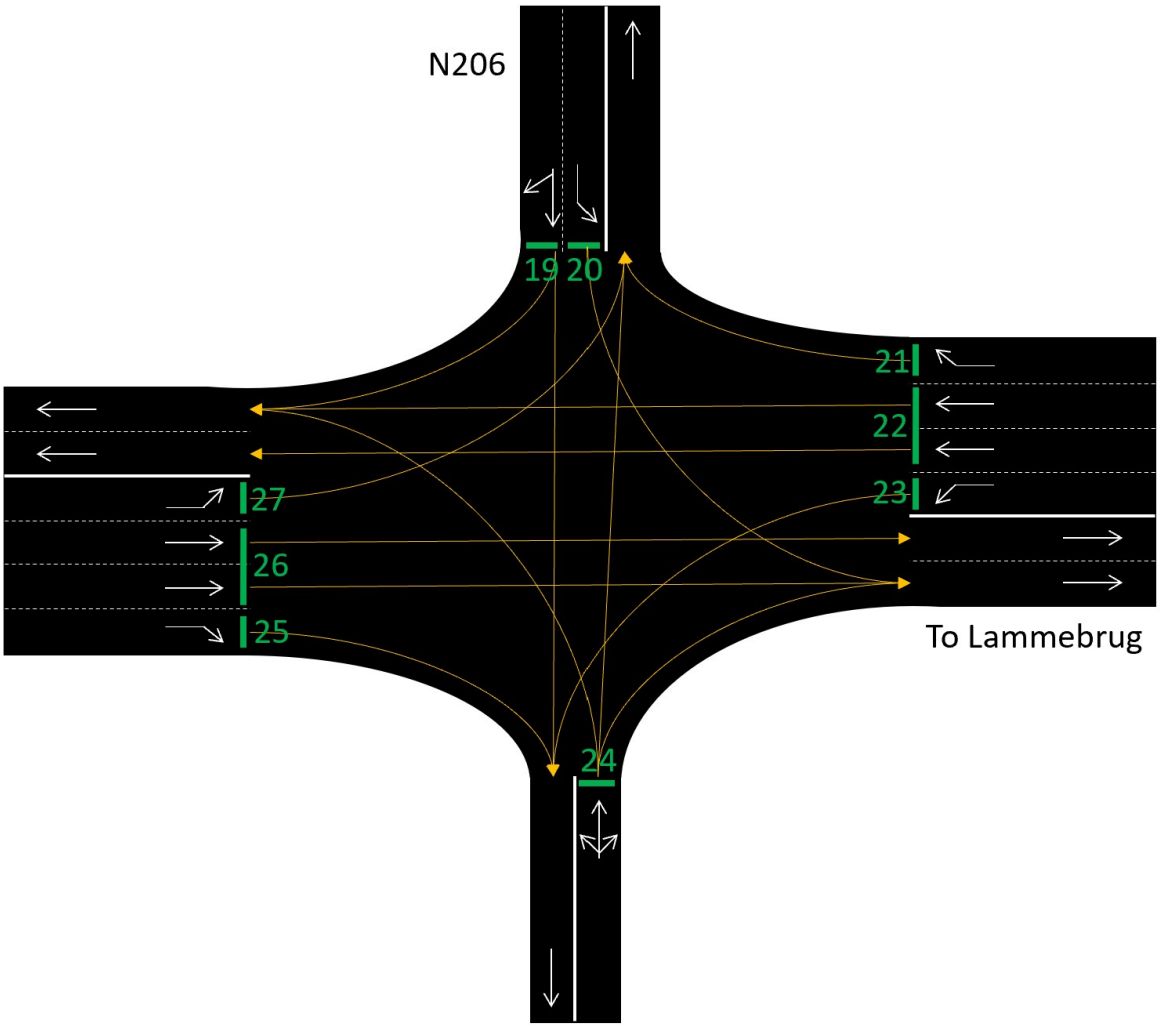

Figure 3-4: Schematic layout of the upstream junction

3-2 Lammebrug junction model

The linear model described in Section 2-1 can be used to model the dynamics for the Lammebrug junction. A total of 18 lanes are to be controlled of which 4 are midstream lanes. Some parameters require calibration, these are outflow rates and turn fractions. The state matrix $A = I_{18}$ is assumed to be correct. Lane inflow is provided in Table 3-1

Table 3-1: Traffic inflow to system

signal head	Inflow [veh/h]
1	802
2	157
3	10
4	17
5	118
6	322
7	6
8	13
9	236
10	276
11	884
12	4
13	2

3-2-1 Model parameters

The model parameters to be determined are outflow rates and turn fractions. First an initial guess is made, after which model calibration done using a least squares approach.

For the outflow for any lane i (α_i) the initial estimated value is $3m/s$. This indicates a vehicle of length $6m$ passes by every $2s$ during a green signal. The initial guess for turn fractions from upstream lanes to midstream lanes are all provided in Table 3-2.

For model calibration, a least squares approach is used. 20 hours of simulation is performed with a sampling time of $T_S = 5s$. This means a total of 14400 data points are available. 15 hours of this simulation data is used for model calibration, the other 5 hours are used for validation. The following constraints were set on B matrix entries:

- Outflow is constrained at $\alpha_i \leq -2$ for every lane.
- Midstream lane inflow is constrained at $-p_{ij} * \alpha_i \leq -\alpha_i - 0.4$.
- Any entry representing two uncorrelated lanes is set to 0.

The Variance Accounted For (VAF) on the validation set for the uncalibrated model was 0.71. After model calibration, the VAF was increased to 0.83. The updated outflow rates are $\alpha_i = -2m/s$ for every lane. The updated turn fractions can be found in Table 3-3

Table 3-2: Turn fractions from lane i to j (p_{ij})

lane i	**lane** j	p_{ij} **(-)**
3	14	0.10
5	14	0.86
9	14	0.49
3	15	0.85
9	15	0.51
2	16	0.66
10	16	0.69
13	16	0.17
2	17	0.34
10	17	0.31
13	17	0.08

Table 3-3: Calibrated turn fractions from lane i to j (p_{ij})

lane i	**lane** j	p_{ij} **(-)**
3	14	0.38
5	14	0.8
9	14	0.06
3	15	0.35
9	15	0.11
2	16	0.73
10	16	0.25
13	16	0.48
2	17	0.8
10	17	0.71
13	17	0.45

3-3 Control strategy

Given the calibrated model, a control strategy can be determined. This involves setting up constraints and control parameters like weights and prediction horizon.

3-3-1 Constraints

In Chapter 2, all constraints were provided in (2-6). Intuitively, the minimal state value for any lane is $0m$, as a queue length cannot be negative. However, if the outflow rate per sampling time is larger than the length of a single vehicle, the controller would never provide a green signal to a single queued vehicle. In practical terms this means the minimal state value is between 0 and the outflow rate: $\alpha \leq x_{min} \leq 0$ [1]. Now queues of just a single vehicle will be provided a green signal when beneficial, but it will never provide a green signal when the queue length is $0m$. The midstream lanes not only have a minimal state value, but also a maximum which is a queue length of $90m$.

Regarding input constraints, as mentioned before, the inputs are either 0 (red signal) or 1 (green signal). Other input constraints are the restriction of two conflicting streams receiving a green signal. In total, there are 36 conflicting areas at the Lammebrug junction. These can be found in Table 3-4.

Table 3-4: Conflicting signals

Signal head	Conflicts with:
1	12,13,15
2	10,11,13,15
3	4,5,6,7,8,9,16,17
4	5,6,9,16
5	9,16
6	9,16
7	9,16,17
8	16
9	16
10	13
11	13,14,15
12	13,15
13	14,15

36 conflicting signals indicate 108 inequality constraints for each timestep, as mentioned in Chapter 2.

3-3-2 Controller parameters

With the model and constraints, it is almost possible to put together an optimization problem. The only parameter to be determined is the sampling time T_s (s), as the model inflow and outflow are dependent on this value. For each timestep, the signal value is determined to be

either green or red. The minimal green time is $5s$ [1], hence a sampling time of $T_s = 5s$ is chosen.

The prediction horizon H is a parameter to be chosen by the designer. Increasing the horizon tends to achieve better controller performance, but comes with the cost of higher computation times. As a starting point, it is set to $H = 6$, which indicates a prediction of $30s$ ahead. The horizon is increased in different simulations up to $H = 12$, in order to compare the trade-off between performance and computational expenses.

In the cost function in Equation (2-5), one can observe a weight on each individual queue length. Every weight w_i is a parameter to be chosen and tuned by the designer, based on preferences by the road manager. The initial weights are set as the amount of lanes that are controlled by the the signal. For example, signal 11 provides a signal for 2 lanes, hence $w_{11} = 2$. One exception is made for the bus lane, as this one has priority over regular traffic. Its weight, w_4, is set to 5. This ensures busses to receive immediate green signal as they approach the junction.

3-4 Summary

The main insights from the problem formulation are the following. Based on the junction layout, an 18-state system is developed for the Lammebrug junction with the primary goal of minimizing queue lengths. Achieving this objective requires accurate system parameters. Initially, these parameters are estimated by the designer. Using a least squares approach, the lane outflow and turn fractions are calibrated. With an accurate model, cost function, and addition of constraints on both states and inputs, MPC can be performed effectively. When the controller is working well, the network can be expanded by incorporating an upstream intersection. This provides the opportunity to regulate traffic inflow upstream to the road owners desires.

Chapter 4

Results

In this chapter, simulation results are shown. Simulations are executed in VISSIM using the COM interface, which allows the integration of custom algorithms from software, like MATLAB or Python. In this research MATLAB is used, the code is provided in Appendix B. VISSIM is a microscopic simulation environment that accurately represents traffic dynamics and individual vehicle behavior [25]. Therefore, it can very accurately measure system states (queue lengths). These queue lengths are fed into the MATLAB workspace and used as $x(0)$ for the MPC optimization step. In MATLAB the optimal input sequence for the horizon is determined of which the first control inputs are fed into the VISSIM model and executed. A visualization of this process is provided in Figure 4-1. It repeats every timestep within the simulation. As a baseline, a traffic signal optimization tool is used called COCON [33]. COCON control is typically used at road junctions to handle peak load during rush hours.

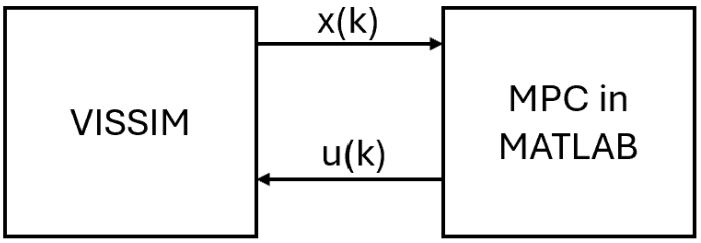

Figure 4-1: Visualization on MATLAB VISSIM communication.

The chapter is organized as follows. The first simulation results include normal traffic without the bridge opening in Section 4-1. In Section 4-2, simulation results are shared where a bridge opens for 5 minutes after half an hour of simulation time. Section 4-3 shows results of the extended network, where the upstream intersection is also included. Finally the results are discussed in Section 4-4.

4-1 Normal traffic simulations

For the first scenario, the bridge stays closed during a 2 hour morning rush simulation. Several simulations are done with different traffic inflow profiles and horizons.

For the initial simulation, the lane queue weights are assigned based on the number of lanes they control, with the exception of the bus lane, which is given a weight of 5. Traffic inflow is set according to Table 3-1. In VISSIM, vehicle inputs are stochastic, meaning the average inflow over time follows the values in Table 3-1, but it is normally distributed, resulting in fluctuations with peaks. The horizon is set to $H = 6$, with a sampling time of $T_s = 5s$. This indicates that the model plans 30 seconds ahead.

The average delay per road user for the COCON controlled network (baseline) is $38.1s$. For the MPC-controlled systems this is 10% less at $34.3s$. Yet there is undesired behaviour occurring. signal 3 and 13 never provide green time during the entire simulation. This can be explained by the large amount of conflicting signals they both have, and their respective weights should be increased. After multiple simulations, where weights are tuned based on observations, the ideal values were found as in Table 4-1.

Table 4-1: Individual lane weights.

Weight	Value
w_1	1
w_2	2
w_3	2
w_4	5
w_5	1.5
w_6	1.5
w_7	0.5
w_8	0.5
w_9	1.5
w_{10}	0.5
w_{11}	2
w_{12}	0.5
w_{13}	2
w_{14}	0.5
w_{15}	2.5
w_{16}	1.5
w_{17}	0.5
w_{18}	5

The controller is tested in four different traffic inflow scenarios. Regular inflow from Table 3-1, 20% increased inflow, 20% decreased inflow, and fluctuating inflow. The latter one is the most realistic scenario as traffic inflow in reality fluctuates much. In this case, the traffic inflow fluctuates between 80% and 120% of the known average inflow during morning rush. It ramps up in the first hour, and ramps down in the second hour. For every traffic inflow regime, the performance of four different horizons are explored. Also the average computation time per timestep is measured. The maximum amount of nodes to be explored is set to 1.5E6 for the

'intlinprog' solver in MATLAB [29]. The average delay time per road user for every horizon and inflow regime can be found in Tables 4-2, 4-3, 4-4, and 4-5.

As was mentioned before, the problem contains $H * 18$ decision variables to optimize whilst staying within input and state constraints. For $H = 6$, the controller has 108 inputs to optimize and 1128 linear inequality constraints to respect. For $H = 12$, there are 216 decision variables to optimize whilst respecting 2220 inequality constraints. Increasing system size leads to a bigger optimization problem, the computation times of different horizons are shared in Table 4-6.

Table 4-2: Simulation results with regular traffic inflow

Controller	Average delay per road user in seconds
COCON baseline	38.1
H = 6	36.4
H = 8	32.5
H = 10	32.1
H = 12	30.5

Table 4-3: Simulation results with low traffic inflow. 80% of regular inflow.

Controller	Average delay per road user in seconds
COCON baseline	34.9
H = 6	31.5
H = 8	26.9
H = 10	26.4
H = 12	26.3

Table 4-4: Simulation results with high traffic inflow. 120% of regular inflow.

Controller	Average delay per road user in seconds
COCON baseline	48.8
H = 6	44.4
H = 8	41.8
H = 10	40.6
H = 12	39.6

Table 4-5: Simulation results with fluctuating traffic inflow. 80-120% of regular inflow.

Controller	Average delay per road user in seconds
COCON baseline	40.4
H = 6	34.7
H = 8	32.4
H = 10	30.5
H = 12	30.7

Table 4-6: Different horizons and their average computation time for one optimization step.

H	Regular inflow	Low inflow	High inflow	Fluctuating inflow
6	$0.04s$	$0.03s$	$0.06s$	$0.03s$
8	$0.11s$	$0.07s$	$0.12s$	$0.09s$
10	$0.40s$	$018s$	$0.43s$	$0.36s$
12	$1.36s$	$0.75s$	$1.34s$	$1.35s$

4-1-1 Signal comparison

Comparing the green times and signal sequences from the MPC approach with the COCON baseline can provide valuable insights. In this subsection, three MPC-controlled signals are compared to the corresponding signals of the baseline. The lanes to be compared are 1, 3, and 11. Lane 1 and 11 both experience very high traffic volumes, directly leaving the network without passing midstream lanes. lane 3 is compared too, characterized by significantly lower traffic volume. Also, lane 3 conflicts with 8 other signals, and a big share of this traffic travels through midstream lanes. The scenario used for this comparison is the one with fluctuating traffic inflow as described in Table 4-5, as this is the most realistic traffic profile. The horizon is chosen as 8.

In Figure 4-2, one can observe both the signal sequences of the baseline, and MPC-controlled network of lane 1. The total green time provided in the 2 hour simulation by the baseline is 65:45 minutes. The MPC-controller provided a total of 51:20 minutes green time. The average vehicle delay in lane 1 is $26s$ for both controllers.

In Figure 4-3, one can observe both the signal sequences of the baseline, and MPC-controlled network of lane 3. The total green time provided in the 2 hour simulation by the baseline is 05:30 minutes. The MPC-controller provided a total of 01:20 minutes green time. The average vehicle delay in lane 3 for the baseline is $84s$, whereas the average vehicle delay for the MPC in lane 3 is $162s$.

In Figure 4-4, one can observe both the signal sequences of the baseline, and MPC-controlled network of lane 11. The total green time provided in the 2 hour simulation by the baseline is 44:00 minutes. The MPC-controller provided a total of 41:05 minutes green time. The average vehicle delay in lane 11 for the baseline is $32s$, whereas the average vehicle delay for the MPC in lane 11 is $18s$.

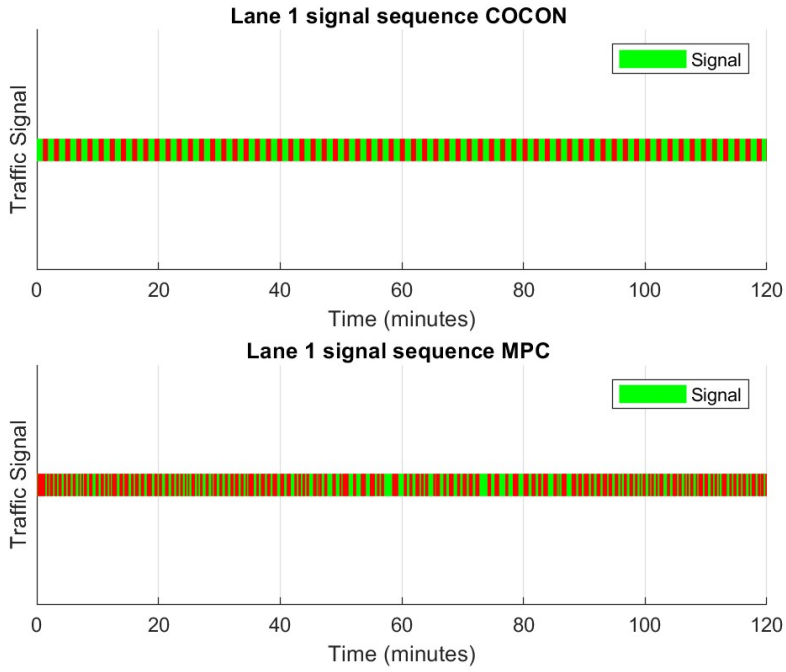

Figure 4-2: Traffic signal sequence comparison for lane 1.

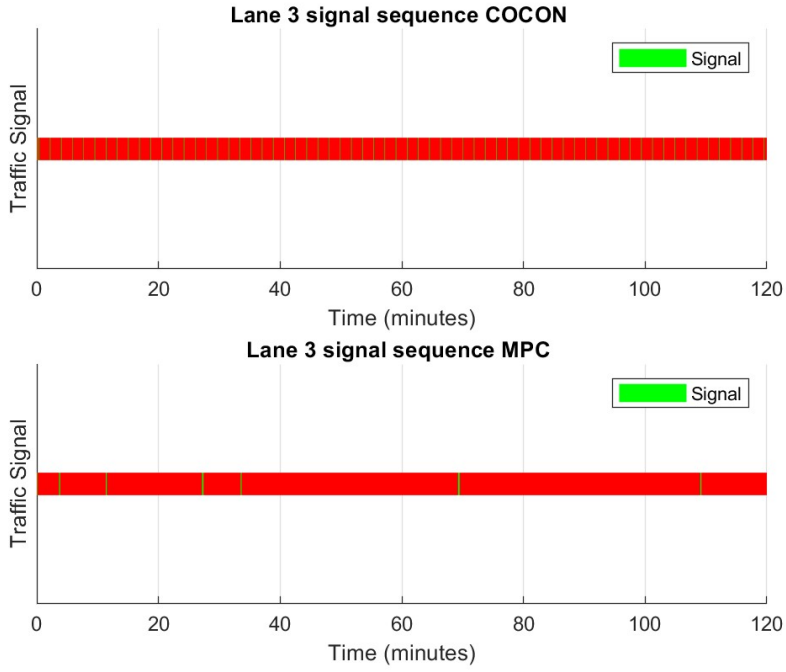

Figure 4-3: Traffic signal sequence comparison for lane 3.

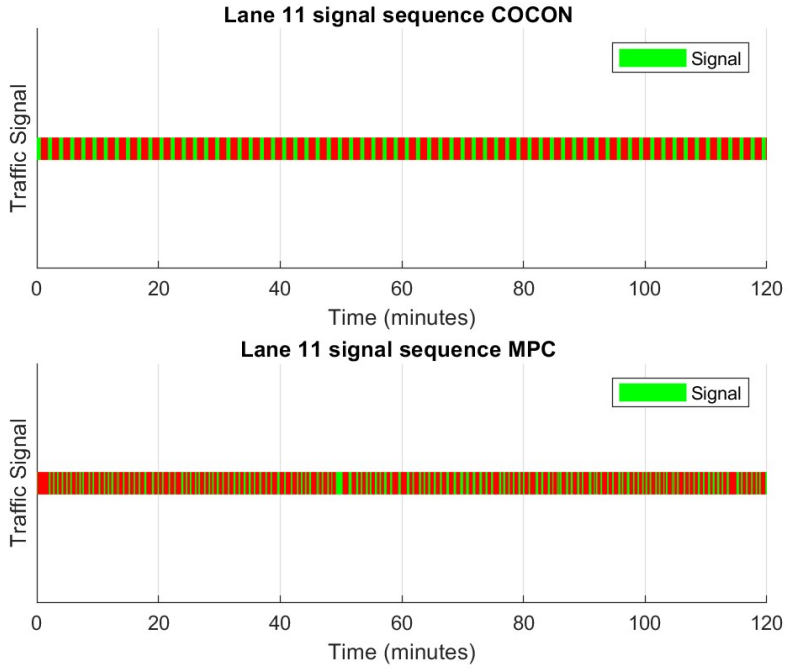

Figure 4-4: Traffic signal sequence comparison for lane 11.

4-2 Bridge opening

In this section, the same scenarios as the previous section are simulated, but now the bridge opening is included. After half an hour of simulation time, the bridge opens for 5 minutes. The average delay time per road user for every horizon and inflow regime can be found in Tables 4-7, 4-8, 4-9, and 4-10.

Table 4-7: Simulation results with regular traffic inflow and bridge opening after half an hour

Controller	Average delay per road user in seconds
COCON baseline	53.6
H = 6	48.2
H = 8	46.1
H = 10	43.3
H = 12	43.6

Table 4-8: Simulation results with low traffic inflow. 80% of regular inflow, and bridge opening after half an hour

Controller	Average delay per road user in seconds
COCON baseline	47.8
H = 6	38.2
H = 8	37.5
H = 10	36.4
H = 12	36.2

Table 4-9: Simulation results with high traffic inflow. 120% of regular inflow, and bridge opening after half an hour

Controller	Average delay per road user in seconds
COCON baseline	76.9
H = 6	71.1
H = 8	66.3
H = 10	65.9
H = 12	65.0

Table 4-10: Simulation results with fluctuating traffic inflow. 80-120% of regular inflow, and bridge opening after half an hour

Controller	Average delay per road user in seconds
COCON baseline	55.0
H = 6	47.9
H = 8	44.8
H = 10	44.7
H = 12	41.8

4-2-1 Signal comparison

Signal sequences are provided of lane 1 and 15. Both are given a red signal during bridge opening. Moreover, lane 1 and 15 are conflicting signals. In the 10 minutes following the bridge opening, it could potentially be very valuable to observe what happens with these signals. In these 10 minutes, lane 1 has a total green time of 5:45 minutes. Lane 15 has a total of 3:20 minutes green time. There is also 55 seconds during these 10 minutes where both lane 1 and lane 15 are provided a red signal. This can be explained by the 'all red phase' constraints in (2-6d) and (2-6e). The signal sequences of lane 1 and 15 are provided in Figure 4-5, and Figure 4-6 respectively.

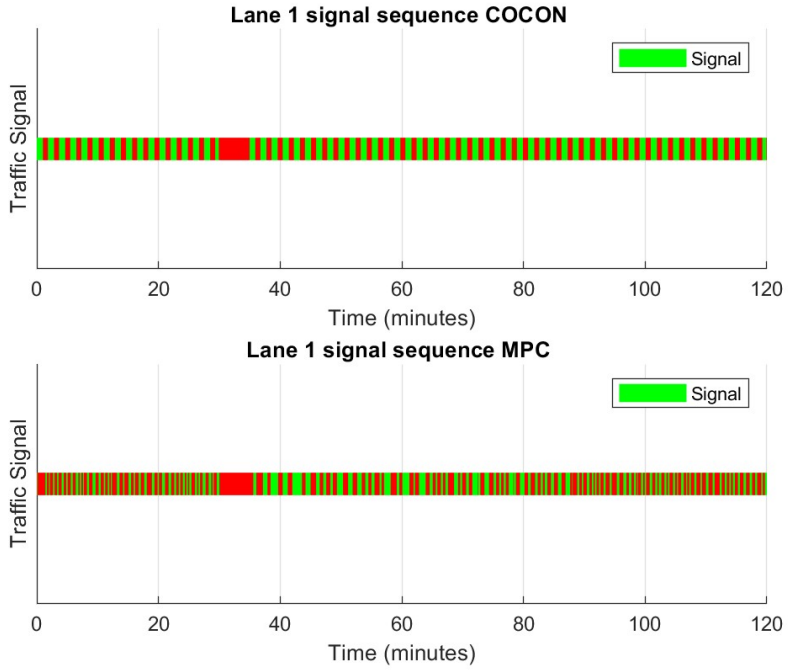

Figure 4-5: Traffic signal sequence comparison for lane 1, bridge opening at 30 minutes.

4-2-2 Bridge knowledge

In this section, the comparison is made between two types of MPC-controllers. The first type incorporates the knowledge of the bridge opening/closing time into the corresponding optimization problem. The second type is not aware of the bridge opening/closing occurring until the event actually happens.

In previous simulations the total time was set to 2 hours, with a bridge opening of 5 minutes after half an hour. To have a better comparison, the simulation time is now set to 40 minutes with a 5-minute bridge opening after 15 minutes of simulation time. The results can be found in Table 4-11

The signal sequence for the MPC-controllers with horizon $H = 12$ is provided in Figure 4-7. The first one is not aware of the bridge opening until the event happens, whereas the second

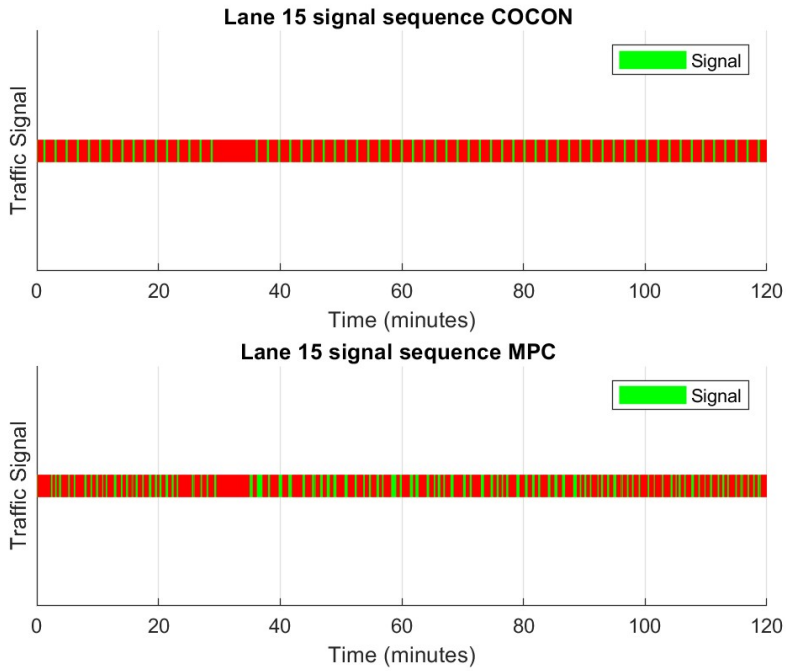

Figure 4-6: Traffic signal sequence comparison for lane 15. Bridge opening occurs at 30 minutes in to the simulation.

Table 4-11: 40 minutes simulation results with bridge opening after 15 minutes. Comparison between MPC-controller that has knowledge of the bridge opening beforehand, and MPC-controller that does not have knowledge of the bridge opening.

Horizon	Delay with knowledge [s]	Delay no knowledge [s]
$H = 6$	80.5	82.3
$H = 8$	74.4	77.1
$H = 10$	71.1	75.6
$H = 12$	65.9	71.6

signal sequence corresponds to a controller that is aware of the bridge opening a minute before it actually occurs.

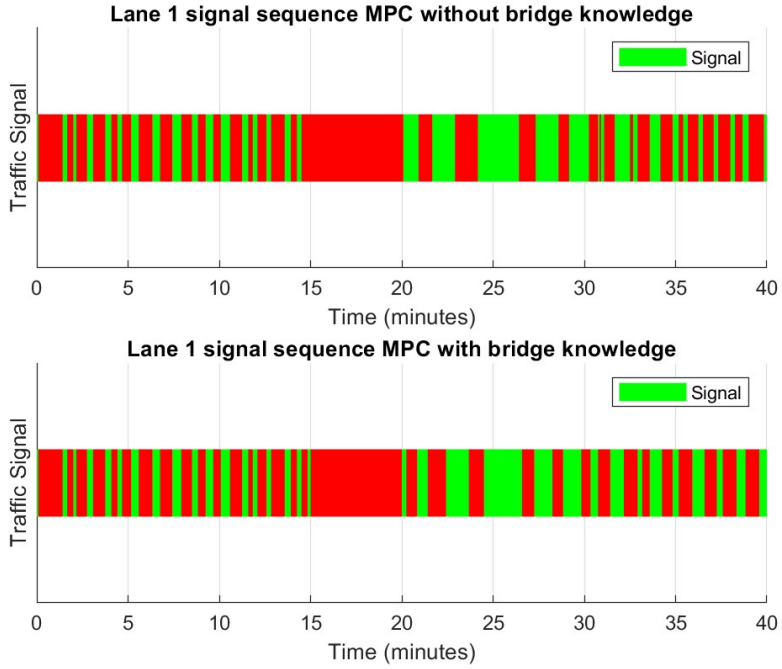

Figure 4-7: Traffic signal sequence comparison regarding bridge knowledge for lane 1

4-3 Upstream traffic flow manipulation

Manipulating traffic flow in surrounding areas of the Lammebrug junction holds the potential to reduce congestion. In Section 3-1-2, an upstream intersection was introduced. The layout of this intersection can be found in Figure 3-4. This upstream intersection is incorporated in this section, to observe what happens when upstream traffic is manipulated. There are two potential strategies for control. A centralized approach and a decentralized approach:

- **Centralized:** The Lammebrug junction and upstream intersection are considered as a single system. They are controlled by one controller.
- **Decentralized:** The Lammebrug junction and upstream intersection are considered two separate systems. Each system is controlled by an individual controller.

If a centralized control approach is used, the system size is 27 states. For a decentralized approach there are 2 problems to solve, one of size 18 and one of size 9. The upstream intersection is just a single intersection which means it does not have midstream lanes. It has 20 conflicting signals, which means if the horizon H is chosen as 6, for the centralized approach, there are 162 decision variables to optimize whilst respecting 1722 linear inequality

constraints. For $H = 12$ this is 324 decision variables to optimize whilst respecting 3390 inequality constraints.

In order to make fair comparisons and a valid result discussion, four scenarios are simulated:

- Entire system is controlled by COCON.
- Lammebrug junction is controlled by MPC, upstream intersection is controlled by COCON.
- Entire network is controlled by MPC, but decentralized (one controller for the upstream intersection, and one controller for the Lammebrug junction).
- The entire network is controlled by one MPC-controller.

Instead of only gathering data on delay times at the Lammebrug junction (like in previous simulations), the delay times at the upstream intersection is now also included in the analysis. This generally results in higher overall delay times since much of the traffic passes through two or even three intersections within the simulation. Including upstream delay times is necessary for a fair comparison, as traffic stalled at an upstream intersection might lead to smooth flow at the Lammebrug junction but simultaneously causes congestion upstream.

For every simulation, the fluctuating traffic demand is used, where the average inflow for the extra lanes is provided in Table 4-12. The simulation time is again 2h with a 5 minute bridge opening after half an hour.

Table 4-12: Traffic inflow to upstream intersection lanes

Signal head	**Inflow [veh/h]**
19	162
20	320
21	Outflow Lammebrug
22	Outflow Lammebrug
23	Outflow Lammebrug
24	16
25	7
26	644
27	281

The average delay for each road user when the entire network is controlled by COCON is $\underline{93.3s}$. The other scenario results can be found in Tables 4-13, 4-14, and 4-15. These represent the scenario where upstream traffic is regulated using COCON and the Lammebrug with MPC, full network decentralized MPC and full network centralized MPC, respectively. Note that the double intersection is regulated using one controller in the decentralized scenario, but the upstream intersection is controlled using a separate controller.

The average computation time for one timestep is shown for both the decentralized and centralized approach in Table 4-16. The simulations are executed on a laptop with the following processor: "AMD Ryzen 7 6800HS Creator Edition, 3.2 GHz base clock speed, 8 cores, 16 logical processors".

Table 4-13: Simulation results with upstream COCON regulation and MPC at the Lammebrug junction.

Horizon	Average delay per road user [s]
$H = 6$	71.4
$H = 8$	66.9
$H = 10$	66.7
$H = 12$	62.9

Table 4-14: Simulation results with decentralized MPC approach.

Horizon	Average delay per road user [s]
$H = 6$	66.9
$H = 8$	63.1
$H = 10$	58.4
$H = 12$	58.1

Table 4-15: Simulation results with centralized MPC approach.

Horizon	Average delay per road user [s]
$H = 6$	63.9
$H = 8$	61.3
$H = 10$	57.9
$H = 12$	54.8

Table 4-16: Average computation time per timestep. Comparison between centralized and decentralized approach

Horizon	Average time decentralized [s]	Average time centralized [s]
$H = 6$	0.03	0.12
$H = 8$	0.09	0.5
$H = 10$	0.36	2.47
$H = 12$	1.35	11.22

4-4 Discussion

The primary aim of this study is to investigate a model predictive control approach for urban traffic signal control, particularly in scenarios involving known disruptive events. This section discusses the findings. Quantifying the results to enable the drawing of meaningful conclusions.

Calibration

The calibration of the model significantly increased its accuracy compared to the initially estimated model from Chapter 3. The Variance Accounted For (VAF) improved from 0.71 to 0.83, as explained in Section 3-2-1. While this still falls short of a perfect VAF of 1, it is important to recognize that traffic dynamics are very stochastic, as discussed multiple times in this report. Factors such as fluctuating vehicle inflow patterns, speed variations, and driver behavior, introduce variability that is impossible to predict with complete accuracy. Although a microscopic model could potentially capture individual vehicle speed variations more precisely, it would also increase the complexity and size of the problem. Enhancing knowledge of vehicle demand by collecting upstream data during simulations could further refine the model parameters in real-time. However, given that a microscopic linear model is being used, the achieved accuracy is sufficiently robust for application in the controller.

Without the constraint on outflow rates, the calibration might have given a value closer to 0 for this parameter on most entries. This would have caused the weight of its corresponding lane to increase during the iterative tuning process, which would have essentially lead to a similar controller performance in the rest of the simulations.

Simulation results

A lot of simulations are done with varying traffic inflow patterns and with/without bridge opening. Without a bridge opening, the MPC-controller with horizon $H = 12$ typically decreases average vehicle delay by approximately 20%. One can observe good performance for shorter horizons like $H = 10$, or $H = 8$ as well. Considering that the computation time only took $1.35s$ for $H = 12$ on a laptop with aforementioned specifications, it would not make sense to go for a lower horizon in practice. Of course, when the system size increases, it is a good thing to keep in mind that also lower horizons provided good results. Looking at the signal comparison, a lot of lessons can be learned. First, too much green time is provided by the baseline. Providing green time to a lane typically reduces its saturation. The traffic outflow from a less saturated lane is smaller compared to that of highly saturated lanes. Therefore it is undesired to provide too much green time consecutively to a single lane. Another insight is the low amount of green time intervals lane 3 gets as shown in Figure 4-3. It only gets green a few times to make small amount vehicles pass through. This is beneficial over green signals every 2 minutes, as this blocks off eight other signals at the intersection that are conflicting.

The next simulations that were done involved a bridge opening, which was the focal point of this study. Promising results were obtained using the same controller that was designed for the regular traffic scenario. Delay times decreased up to 25% compared to the COCON

baseline for horizons of $H = 12$. The green time provided for lanes 1 and 15 after the bridge opening emphasize the importance on the required green time, whilst not providing too much green time consecutively. This finding is supported by the fact that 11 signal switches between signal 1 and 15 occurred in the 10 minutes after the bridge was closed again.

The predictive behaviour of MPC was hypothesized to be beneficial when it comes to having prior knowledge of the bridge opening. Traffic could be provided with green time using the given fact that they might not have this option during the opening. The 40-minute simulation results, that are provided in Section 4-2-2, support the hypothesis, where the MPC-controller with bridge opening knowledge continues to outperform the MPC-controller that does not have this prior information. Increasing the horizon lead to an increased gap between delay times resulted by applying the two different controllers. Looking at the signal comparison, it becomes clear why the controller with knowledge produces better results. Right before the bridge opening, it provides green time to lane 1 which it 'knows' cannot happen after that for at least the next minute (it does not look further in the future than 1 minute).

What might have been an interesting experiment is dynamic weight assignments. Prioritizing blocked off lanes right before and right after bridge openings. This essentially should have the same effect as having knowledge on the bridge opening, but might even provide a stronger preference and thus potentially better results. On the other hand, it could prioritize them too much, which potentially leads to congestion in other lanes.

Network expansion

In all simulations, the controller performed well across various demand profiles, and achieved a nearly 25% improvement in restoring flow after a bridge opening compared to COCON. The next step was to expand the network by including another upstream intersection. The MPC-controller was able to restrict signals to turn green for traffic heading towards the bridge, which significantly reduced delay times by 41% in the simulation where the entire network was controlled by MPC. However, the computation time of $11.22s$ per timestep is impractical for implementation. A decentralized approach would be more appropriate in this scenario. In practice, stronger hardware is probably available. This would substantially improve computation times, indicating a preference for the centralized control approach.

Final discussion

In general, the simulation results demonstrate that MPC outperforms COCON. This is in line with existing literature as discussed in the introduction. However, in real-world applications, most traffic signals are controlled by fixed cycles or adaptive systems like SCOOT or SCATS. This preference by road owners for robust and easily understandable solutions over mathematically optimized ones is clear. Nonetheless, this study shows that even a simple linear MPC can surpass traditional approaches.

There is still room for improvement. Enhancing model accuracy, implementing a parameter weight tuning algorithm, and incorporating more upstream intersections into the network could further improve performance.

Finally, for good performance on the road, accurate sensors are needed. They have to be able to identify queue lengths, which requires accurate and robust hardware.

Chapter 5

Conclusion

Regulating traffic in urban areas using an MPC approach is very promising, as proven in the literature. Handling beforehand known disruptive events is a unique field within TSC. This research has demonstrated that MPC is an effective control approach for managing traffic in a network, disrupted by a bridge opening.

The research question stated in the problem formulation is answered by addressing three sub-questions:

Model choice: Several traffic models have been proposed in different papers. The 1-state link model used in this research, as described in Section 2-1, has only one state for each lane and considers a single lane saturation rate. Despite the model's simplicity, computation times increase with network size and horizon. A more accurate model might be feasible since a VAF of 0.83 after calibration, leaves room for improvement. However, nonlinear models might be too complex for efficient optimization. The link transmission model could be a promising option for a more accurate, yet still linear model. Nonetheless, the 1-state link model achieved promising controller performance. Additionally, network expansion was highly beneficial as can be concluded from the results in Section 4-3. This is made possible by the model's efficiency. Therefore, the 1-state link model is concluded to be a very efficient and sufficiently accurate representation of traffic dynamics.

Cost function and constraints: For the cost function, only the queue lengths were considered, resulting in a linear cost. All inequality constraints were also linear, allowing for relatively fast optimization compared to more complex models. This is very beneficial for urban traffic networks with multiple junctions. The cost function could be expanded with additional linear terms, such as penalties for signal value switching. Overall, the cost function and constraints provided good performance and ensured a safe intersection without conflicts. Thus, they are well-constructed for efficient traffic regulation in urban networks. Weights were necessary to allocate green time to lanes with many conflicting areas.

Model validation and controller testing: The model was not sufficiently accurate before calibration. After calibration, the VAF was increased by 17% to 0.83, indicating room for improvement. Still, the results obtained with the calibrated model were promising. The

MPC-controller consistently outperformed COCON in various scenarios. Parameters adjusted included traffic demand, prediction horizon, and bridge openings. These results indicate that the MPC-controller is useful in improving the traffic conditions in terms of delay. The predictive nature of MPC proved beneficial for managing disruptive events, aligning with the assumptions made in the introduction.

With these sub-questions answered, the research question "*How can control strategies mitigate traffic congestion in urban areas experiencing frequent, beforehand known disruptive events?*" can be addressed:

Control strategies involving a prediction model can determine future states. With prior knowledge of disruptive events, the predictive controller can take measures to mitigate negative effects. Upstream traffic manipulation can further enhance traffic flow in urban areas by influencing inflow at pivotal junctions. To achieve this, a computationally efficient prediction model is required.

5-1 Future work and recommendations

This research has demonstrated the effectiveness of using linear MPC for TSC at the Lammebrug junction in Leiden. Future studies could potentially achieve even better results. This section offers suggestions for future work to help realize these improvements.

5-1-1 MPC techniques

In this research, linear MPC was utilized by predicting future states using the 1-state link model as detailed in Section 2-1. However, more advanced models, such as the link transmission model, were not explored. The link transmission model can differentiate between three lane saturation rates, potentially offering greater model accuracy than the 1-state link model. For future work, it would be highly beneficial to compare both models within a simulation environment, evaluating the trade-offs between model accuracy and computation time.

The individual lane weights for queue lengths were manually tuned based on system behavior, as described in Section 4-1. For future research, exploring a hyper-parameter tuning algorithm could potentially yield better weight values. Additionally, assigning dynamic weights could be highly beneficial. After bridge openings, prioritizing certain lanes that require more green time to alleviate congestion could be an effective strategy.

5-1-2 Data collection

Accurate, real-time traffic data collection is essential for effective system control. For the 1-state link model, having precise information on queue lengths, whether in meters or the number of vehicles, is crucial for performing MPC. While this data is always 100% accurate in a simulation environment, real-world implementations present several challenges. Cameras can be obstructed, wind can displace sensors, and the varying sizes and colors of vehicles can cause detection issues. Future research is recommended to explore these challenges and find solutions to overcome them. This would enable the practical application of the control approach described in this research.

5-1-3 Real-World Implementation

The results of this research, obtained in a realistic microscopic simulation environment, were promising. Therefore, it is recommended to implement MPC strategies for TSC in practice. To implement MPC for TSC, the controller must be connected to signal installations. This can be challenging, as the hardware may require specific software for manipulation. Promising future research would be to investigate the practical challenges of traffic signal manipulation. The information gathered from such research would make the implementation of MPC for TSC in practice very realistic.

Appendix A

Linear MPC problem formulation for TSC

This appendix explains how to formulate the optimization problem, in order to solve it using a solver that requires the notation as in (A-1).

$$\min_{\mathbf{u}} f^T\mathbf{u} \text{ subject to } \begin{cases} \mathbf{u} \text{ (intcon) are integers} \\ A \cdot \mathbf{u} \leq b \end{cases} \tag{A-1}$$

In the Integer Linear Programming (ILP) equation (A-1), $f^T\mathbf{u}$ is equal to the cost function in (2-5). The inequality constraints in (2-6) are summarized in the term $A\mathbf{u} \leq b$.

In Section 2-1, the linear model dynamics are introduced and summarized in the form $\mathbf{x}(k+1) = A\mathbf{x}(k) + B\mathbf{u}(k)$. This is called the prediction model. Using the prediction model from Section 2-1, one can predict any state value at any moment in the future within the horizon as shown in Equation (A-2) (assuming there is no model mismatch).

$$\begin{bmatrix} \mathbf{x}(0) \\ \mathbf{x}(1) \\ \vdots \\ \mathbf{x}(H) \end{bmatrix} = \underbrace{\begin{bmatrix} I \\ A \\ \vdots \\ A^H \end{bmatrix}}_{T} \mathbf{x}(0) + \underbrace{\begin{bmatrix} 0 & 0 & \dots & 0 \\ B & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{H-1}B & A^{H-2}B & \dots & 0 \end{bmatrix}}_{S} \begin{bmatrix} \mathbf{u}(0) \\ \mathbf{u}(1) \\ \vdots \\ \mathbf{u}(H-1) \end{bmatrix} \tag{A-2}$$

Matrices T and S in Equation (A-2) are called the prediction matrices. In the optimization problem formulation from Equation (2-4), f^T is equal to S, and T will be used to set up the inequality constraints over a given horizon H. This can be done as in Equation (A-3). The bar above some parameters indicates multiple stacked column vectors. For example $\overline{u}$ indicates the input sequence over the entire horizon stacked as a column vector.

$$\begin{aligned} \overline{x_{\min}} \leq \begin{bmatrix} x(0) \\ \boldsymbol{x}(1) \\ \vdots \\ \boldsymbol{x}(H) \end{bmatrix} \leq \overline{x_{\text{Max}}}, \\ \rightarrow \overline{x_{\min}} \leq Tx_0 + S\bar{u} \leq \overline{x_{\text{Max}}}, \\ \rightarrow \begin{bmatrix} S \\ -S \end{bmatrix} \bar{u} \leq \begin{bmatrix} \overline{x_{\text{Max}}} - Tx_0 \\ Tx_0 - \overline{x_{\min}} \end{bmatrix} \end{aligned} \tag{A-3}$$

The same procedure, but slightly easier, is followed for the input constraints as shown in Equation (A-4).

$$\begin{aligned} \overline{u_{\min}} \leq \begin{bmatrix} \boldsymbol{u}(0) \\ \boldsymbol{u}(1) \\ \vdots \\ \boldsymbol{u}(H) \end{bmatrix} \leq \overline{u_{Max}}, \\ \rightarrow \begin{bmatrix} I_H \\ -I_H \end{bmatrix} \bar{u} \leq \begin{bmatrix} \overline{u_{\text{Max}}} \\ -\overline{u_{\min}} \end{bmatrix} \end{aligned} \tag{A-4}$$

The third set of constraints to be incorporated are the input restrictions for conflicting signals. If lane i and j are conflicting, then $u_i(k) + u_j(k) \leq 1$ at any time k. A matrix is designed in which each row represents a conflicting area. Every entry in a row is zero, except entry i and j. They are 1. All these rows that represent a conflicting area are stacked together in a matrix called U_c which stands for 'u conflicting'. Then the inequality constraint matrix to handle these constraints in the correct form is as in Equation (A-5)

$$\begin{bmatrix} U_c \otimes I_H \end{bmatrix} \bar{u} \leq \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \tag{A-5}$$

Note that the matrix size of U_c is dependent on the problem and system size. It has as many rows as there are conflicting signals at the network, $\#c$. The amount of columns is the same as the amount of decision variables in one timestep, $m+n$, where m is the amount of upstream lanes and n is the amount of midstream lanes. The size of U_c is $U_c \in \mathbb{R}^{(\#c)\times(m+n)}$

The final constraints to be incorporated are the lagged conflicting signals to provide an all-red phase as can be found in (2-6d) and (2-6e). An all-red phase is required to prevent conflicting signals turning green immediately after each other, which could lead to a collision on the intersection. Consider input u_i to be conflicting with input u_c, and there are a total of $m+n$ decision variables for each timestep. Instead of one row per conflicting, now two rows are needed for a single conflicting signal, as it can both switch from u_i to u_c green and vice versa. In the first row, entry i is set to 1 and entry $c+m+n$ is also set to 1. The rest is 0. In the second row, entry c and entry $i+m+n$ are set to 1. So the matrix, that is called U_{arp},

which stands for All Red Phase, has twice as much columns and twice as much rows compared to U_c, indicating $U_{arp} \in \mathbb{R}^{(2\#c)\times(2m+2n)}$. For notation in the standard form for inequality constraints, it is more convenient to split it up in two parts of which one contains the normal signals and one contains the lagged signals: $U_{arp} = [U_{normal}\, U_{lagged}]$. Now it can be written in the standard form, showcased in Equation (A-6), where both $U_{normal} \in \mathbb{R}^{(2\#c)\times(m+n)}$ and $U_{lagged} \in \mathbb{R}^{(2\#c)\times(m+n)}$.

$$\begin{bmatrix} U_{normal} & U_{lagged} & 0 & \dots & \dots & 0 \\ 0 & U_{normal} & U_{lagged} & 0 & \dots & 0 \\ \vdots & 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & \dots & 0 & U_{normal} & U_{lagged} \end{bmatrix} \bar{u} \leq \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \tag{A-6}$$

With this inequality constraints, an all red phase is provided for any future conflicting input. One final, and very important note is that the previous implemented input $u(k-1)$ is not yet taken into account, whilst it has the same restrictions on all-red phase conflicting signals. This is written in standard form as shown in Equation (A-7)

$$\begin{aligned} &\begin{bmatrix} U_{lagged} & 0 \end{bmatrix} \bar{u} + U_{normal} \cdot u(k-1) \leq \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \\ &\rightarrow \begin{bmatrix} U_{lagged} & 0 \end{bmatrix} \bar{u} \leq \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} - U_{normal} \cdot u(k-1) \end{aligned} \tag{A-7}$$

With this inequality constraint, a lagged signal, $u_i(0)$, is only allowed to have value 1 if its conflicting signal $u_c(-1)$ had value 0.

Appendix B

MATLAB code

This is the MATLAB code used. First the main code, which calls the system, as well as the MPC function. These are called 'Lammeplein_system.m', and 'MPC_full.m', respectively.

B-1 Main

This listing shows the MATLAB code that runs VISSIM using the COM-server as shown in Figure 4-1. For future research in which other algorithms are explored, this can be used as a basis. It is divided in multiple sections for clarity.

Listing B.1: Code that runs VISSIM from MATLAB

```
%% Vissim-COM program
clear;
close all
% set up COM server
vis = actxserver('VISSIM.vissim.240');

% Load vissim files
access_path = pwd;
vis.LoadNet([access_path '\Lammeplein.inpx']);
vis.LoadLayout([access_path '\Lammeplein.layx']);
vnet=vis.Net;

%% simulation parameters

sim = vis.Simulation;

% query of all the properties of the given object
% sim.get

%setting simulation time in seconds
period_time = 7200; % 2h simulation
```

```
22
23 %setting step time (how many steps per second)
24 step_time = 5;
25 sim.set('AttValue', 'SimRes', step_time);
26
27
28 %% Signal controllers
29
30 % signal controllers
31 scs=vnet.SignalControllers;
32 sc1 = scs.ItemByKey(1); %(Signal Controller 1)
33 sc2 = scs.ItemByKey(2); %(Signal Controller 2)
34 sc3 = scs.ItemByKey(3); %(Signal Controller 3)
35 sc4 = scs.ItemByKey(4); %(Signal Controller 4)
36 sc5 = scs.ItemByKey(5); %(Signal Controller 5)
37 sc6 = scs.ItemByKey(6); %(Signal Controller 6)
38 sc7 = scs.ItemByKey(7); %(Signal Controller 7)
39 sc8 = scs.ItemByKey(8); %(Signal Controller 8)
40 sc9 = scs.ItemByKey(9); %(Signal Controller 9)
41 sc10 = scs.ItemByKey(10); %(Signal Controller 10)
42 sc11 = scs.ItemByKey(11); %(Signal Controller 11)
43 sc12 = scs.ItemByKey(12); %(Signal Controller 12)
44 sc13 = scs.ItemByKey(13); %(Signal Controller 13)
45 sc14 = scs.ItemByKey(14); %(Signal Controller 14)
46 sc15 = scs.ItemByKey(15); %(Signal Controller 15)
47 sc16 = scs.ItemByKey(16); %(Signal Controller 16)
48 sc17 = scs.ItemByKey(17); %(Signal Controller 17)
49 sc18 = scs.ItemByKey(18); %(Signal Controller 18)
50
51 sc19 = scs.ItemByKey(19); %(Signal Controller 19)
52 sc20 = scs.ItemByKey(20); %(Signal Controller 20)
53 sc21 = scs.ItemByKey(21); %(Signal Controller 21)
54 sc22 = scs.ItemByKey(22); %(Signal Controller 22)
55 sc23 = scs.ItemByKey(23); %(Signal Controller 23)
56 sc24 = scs.ItemByKey(24); %(Signal Controller 24)
57 sc25 = scs.ItemByKey(25); %(Signal Controller 25)
58 sc26 = scs.ItemByKey(26); %(Signal Controller 26)
59 sc27 = scs.ItemByKey(27); %(Signal Controller 27)
60 sc28 = scs.ItemByKey(28); %(Signal Controller 28)
61 sc29 = scs.ItemByKey(29); %(Signal Controller 29)
62 sc30 = scs.ItemByKey(30); %(Signal Controller 30)
63 sc31 = scs.ItemByKey(31); %(Signal Controller 31)
64 sc32 = scs.ItemByKey(32); %(Signal Controller 32)
65 sc33 = scs.ItemByKey(33); %(Signal Controller 33)
66 sc34 = scs.ItemByKey(34); %(Signal Controller 34)
67 sc35 = scs.ItemByKey(35); %(Signal Controller 35)
68 sc36 = scs.ItemByKey(36); %(Signal Controller 36)
69 sc37 = scs.ItemByKey(37); %(Signal Controller 37)
70 sc38 = scs.ItemByKey(38); %(Signal Controller 38)
71
72 %signal groups
73 sgs1 = sc1.SGs;
74 sgs2 = sc2.SGs;
```

```
75  sgs3 = sc3.SGs;
76  sgs4 = sc4.SGs;
77  sgs5 = sc5.SGs;
78  sgs6 = sc6.SGs;
79  sgs7 = sc7.SGs;
80  sgs8 = sc8.SGs;
81  sgs9 = sc9.SGs;
82  sgs10 = sc10.SGs;
83  sgs11 = sc11.SGs;
84  sgs12 = sc12.SGs;
85  sgs13 = sc13.SGs;
86  sgs14 = sc14.SGs;
87  sgs15 = sc15.SGs;
88  sgs16 = sc16.SGs;
89  sgs17 = sc17.SGs;
90  sgs18 = sc18.SGs;
91
92  sgs19 = sc19.SGs;
93  sgs20 = sc20.SGs;
94  sgs21 = sc21.SGs;
95  sgs22 = sc22.SGs;
96  sgs23 = sc23.SGs;
97  sgs24 = sc24.SGs;
98  sgs25 = sc25.SGs;
99  sgs26 = sc26.SGs;
100 sgs27 = sc27.SGs;
101 sgs28 = sc28.SGs;
102 sgs29 = sc29.SGs;
103 sgs30 = sc30.SGs;
104 sgs31 = sc31.SGs;
105 sgs32 = sc32.SGs;
106 sgs33 = sc33.SGs;
107 sgs34 = sc34.SGs;
108 sgs35 = sc35.SGs;
109 sgs36 = sc36.SGs;
110 sgs37 = sc37.SGs;
111 sgs38 = sc38.SGs;
112
113 % naming them conveniently
114 voorschoterweg_1 = sgs1.ItemByKey(1);
115 voorschoterweg_2 = sgs2.ItemByKey(1);
116 parking_3 = sgs3.ItemByKey(1);
117 bus_4 = sgs4.ItemByKey(1);
118 centre_5 = sgs5.ItemByKey(1);
119 centre_6 = sgs6.ItemByKey(1);
120 centre_7 = sgs7.ItemByKey(1);
121 kanaalweg_8 = sgs8.ItemByKey(1);
122 kanaalweg_9 = sgs9.ItemByKey(1);
123 europaweg_10 = sgs10.ItemByKey(1);
124 europaweg_11 = sgs11.ItemByKey(1);
125 europaweg_12 = sgs12.ItemByKey(1);
126 industry_13 = sgs13.ItemByKey(1);
127 middle_14 = sgs14.ItemByKey(1);
```

```
128 middle_15 = sgs15.ItemByKey(1);
129 middle_16 = sgs16.ItemByKey(1);
130 middle_17 = sgs17.ItemByKey(1);
131 bridge_18 = sgs18.ItemByKey(1);
132
133 cycle_1 = sgs19.ItemByKey(1);
134 cycle_2 = sgs20.ItemByKey(1);
135 cycle_3 = sgs21.ItemByKey(1);
136 cycle_4 = sgs22.ItemByKey(1);
137 cycle_5 = sgs23.ItemByKey(1);
138 cycle_6 = sgs24.ItemByKey(1);
139 cycle_7 = sgs25.ItemByKey(1);
140 cycle_8 = sgs26.ItemByKey(1);
141 cycle_9 = sgs27.ItemByKey(1);
142 cycle_10 = sgs28.ItemByKey(1);
143 cycle_11 = sgs29.ItemByKey(1);
144 cycle_12 = sgs30.ItemByKey(1);
145 cycle_13 = sgs31.ItemByKey(1);
146 cycle_14 = sgs32.ItemByKey(1);
147 cycle_15 = sgs33.ItemByKey(1);
148 cycle_16 = sgs34.ItemByKey(1);
149 cycle_17 = sgs35.ItemByKey(1);
150 cycle_18 = sgs36.ItemByKey(1);
151
152 voorschoterweg_upstream = sgs37.ItemByKey(1);
153 europaweg_upstream = sgs38.ItemByKey(1);
154 %% Simulation setup
155
156 Ts = 5;
157 queueCounter(1) = vnet.QueueCounters.ItemByKey(1);
158 queueCounter(2) = vnet.QueueCounters.ItemByKey(2);
159 queueCounter(3) = vnet.QueueCounters.ItemByKey(3);
160 queueCounter(4) = vnet.QueueCounters.ItemByKey(4);
161 queueCounter(5) = vnet.QueueCounters.ItemByKey(5);
162 queueCounter(6) = vnet.QueueCounters.ItemByKey(6);
163 queueCounter(7) = vnet.QueueCounters.ItemByKey(7);
164 queueCounter(8) = vnet.QueueCounters.ItemByKey(8);
165 queueCounter(9) = vnet.QueueCounters.ItemByKey(9);
166 queueCounter(10) = vnet.QueueCounters.ItemByKey(10);
167 queueCounter(11) = vnet.QueueCounters.ItemByKey(11);
168 queueCounter(12) = vnet.QueueCounters.ItemByKey(12);
169 queueCounter(13) = vnet.QueueCounters.ItemByKey(13);
170 queueCounter(14) = vnet.QueueCounters.ItemByKey(14);
171 queueCounter(15) = vnet.QueueCounters.ItemByKey(15);
172 queueCounter(16) = vnet.QueueCounters.ItemByKey(16);
173 queueCounter(17) = vnet.QueueCounters.ItemByKey(17);
174 queueCounter(18) = vnet.QueueCounters.ItemByKey(18);
175
176 time = zeros(1,period_time/Ts);
177 Q1 = zeros(1,period_time/Ts);
178 Q2 = zeros(1,period_time/Ts);
179 Q3 = zeros(1,period_time/Ts);
180 Q4 = zeros(1,period_time/Ts);
```

```
181 Q5 = zeros(1,period_time/Ts);
182 Q6 = zeros(1,period_time/Ts);
183 Q7 = zeros(1,period_time/Ts);
184 Q8 = zeros(1,period_time/Ts);
185 Q9 = zeros(1,period_time/Ts);
186 Q10 = zeros(1,period_time/Ts);
187 Q11= zeros(1,period_time/Ts);
188 Q12 = zeros(1,period_time/Ts);
189 Q13 = zeros(1,period_time/Ts);
190 Q14 = zeros(1,period_time/Ts);
191 Q15 = zeros(1,period_time/Ts);
192 Q16 = zeros(1,period_time/Ts);
193 Q17 = zeros(1,period_time/Ts);
194 Q18 = zeros(1,period_time/Ts);
195
196 SimRun = 1;
197
198 x0 = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]';
199 u0 = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]';
200
201 simtime = period_time/60;   % simtime in minutes
202 Tmax = simtime*60/Ts;
203 % bridge opening at time t*60/Ts (in minutes)
204 t_bo = 30*60/Ts;
205 bridgetime = 5*60/Ts; % 5 minute bridge opening
206
207 run('Lammeplein_system.m');  % run file that sets up the system
208
209 %Initialize simulation
210 x_t = zeros(length(A),Tmax+1);
211 x_t(:,1) = x0;
212 x_model = zeros(length(A),Tmax+1);
213 x_model(:,1) = x0;
214 u_t = zeros(length(B),Tmax);
215 queue_sum = zeros(1,Tmax);
216 queue_sum(1) = sum(x0);
217
218 [x,u] = MPC(A, B, x0, u0, S, T, Uc, Uc_arp, x_min, x_max, u_min, u_max, H
      );
219
220 x_t(:,2) = x;
221 u_t(:,1) = u;
222
223 %% simulation
224
225 for i=1:(period_time * step_time)
226  sim.RunSingleStep;
227
228  if (mod(i,step_time*Ts)) == 0
229     sim.get('AttValue', 'SimSec');
230     time(i/(step_time*Ts)) = i/step_time;
231     % Obtaining queue lengths
```

```
232     Q1(i/(step_time*Ts)) = 1*queueCounter(1).AttValue(['QLenMax(' num2str
            (SimRun) ',' num2str(time(i/(step_time*Ts))) ')']);
233     Q2(i/(step_time*Ts)) = 1*queueCounter(2).AttValue(['QLenMax(' num2str
            (SimRun) ',' num2str(time(i/(step_time*Ts))) ')']);
234     Q3(i/(step_time*Ts)) = 1*queueCounter(3).AttValue(['QLenMax(' num2str
            (SimRun) ',' num2str(time(i/(step_time*Ts))) ')']);
235     Q4(i/(step_time*Ts)) = 1*queueCounter(4).AttValue(['QLenMax(' num2str
            (SimRun) ',' num2str(time(i/(step_time*Ts))) ')']);
236     Q5(i/(step_time*Ts)) = 1*queueCounter(5).AttValue(['QLenMax(' num2str
            (SimRun) ',' num2str(time(i/(step_time*Ts))) ')']);
237     Q6(i/(step_time*Ts)) = 1*queueCounter(6).AttValue(['QLenMax(' num2str
            (SimRun) ',' num2str(time(i/(step_time*Ts))) ')']);
238     Q7(i/(step_time*Ts)) = 1*queueCounter(7).AttValue(['QLenMax(' num2str
            (SimRun) ',' num2str(time(i/(step_time*Ts))) ')']);
239     Q8(i/(step_time*Ts)) = 1*queueCounter(8).AttValue(['QLenMax(' num2str
            (SimRun) ',' num2str(time(i/(step_time*Ts))) ')']);
240     Q9(i/(step_time*Ts)) = 1*queueCounter(9).AttValue(['QLenMax(' num2str
            (SimRun) ',' num2str(time(i/(step_time*Ts))) ')']);
241     Q10(i/(step_time*Ts)) = 1*queueCounter(10).AttValue(['QLenMax('
            num2str(SimRun) ',' num2str(time(i/(step_time*Ts))) ')']);
242     Q11(i/(step_time*Ts)) = 1*queueCounter(11).AttValue(['QLenMax('
            num2str(SimRun) ',' num2str(time(i/(step_time*Ts))) ')']);
243     Q12(i/(step_time*Ts)) = 1*queueCounter(12).AttValue(['QLenMax('
            num2str(SimRun) ',' num2str(time(i/(step_time*Ts))) ')']);
244     Q13(i/(step_time*Ts)) = 1*queueCounter(13).AttValue(['QLenMax('
            num2str(SimRun) ',' num2str(time(i/(step_time*Ts))) ')']);
245     Q14(i/(step_time*Ts)) = 1*queueCounter(14).AttValue(['QLenMax('
            num2str(SimRun) ',' num2str(time(i/(step_time*Ts))) ')']);
246     Q15(i/(step_time*Ts)) = 1*queueCounter(15).AttValue(['QLenMax('
            num2str(SimRun) ',' num2str(time(i/(step_time*Ts))) ')']);
247     Q16(i/(step_time*Ts)) = 1*queueCounter(16).AttValue(['QLenMax('
            num2str(SimRun) ',' num2str(time(i/(step_time*Ts))) ')']);
248     Q17(i/(step_time*Ts)) = 1*queueCounter(17).AttValue(['QLenMax('
            num2str(SimRun) ',' num2str(time(i/(step_time*Ts))) ')']);
249     Q18(i/(step_time*Ts)) = 1*queueCounter(18).AttValue(['QLenMax('
            num2str(SimRun) ',' num2str(time(i/(step_time*Ts))) ')']);
250
251     % Finding optimal policy
252     x_t(:,i/(step_time*Ts) + 1) = [Q1(i/(step_time*Ts)) Q2(i/(step_time*
            Ts)) Q3(i/(step_time*Ts)) Q4(i/(step_time*Ts)) Q5(i/(step_time*Ts)
            ) Q6(i/(step_time*Ts)) Q7(i/(step_time*Ts)) Q8(i/(step_time*Ts))
            Q9(i/(step_time*Ts)) Q10(i/(step_time*Ts)) Q11(i/(step_time*Ts))
            Q12(i/(step_time*Ts)) Q13(i/(step_time*Ts)) Q14(i/(step_time*Ts))
            Q15(i/(step_time*Ts)) Q16(i/(step_time*Ts)) Q17(i/(step_time*Ts))
            Q18(i/(step_time*Ts))]';
253     u_t(:,i/(step_time*Ts)) = u;
254     [x,u] = MPC_full(A, B, x_t(:,i/(step_time*Ts)), u, S, T, Uc, Uc_arp,
            x_min, x_max, u_min, u_max, u_max_b, H, t_bo, bridgetime, i/(Ts*
            step_time));
255     x_model(:,i/(step_time*Ts) + 1) = x;
256     if i == period_time * step_time
257         break
258     end
```

```
259     if u(1) == 1
260         voorschoterweg_1.set('AttValue', 'State', 3);    % green
261     elseif (u(1) == 0) && (u_t(1,(i/(step_time*Ts))) == 1)
262         voorschoterweg_1.set('AttValue', 'State', 4);    % amber
263     else
264         voorschoterweg_1.set('AttValue', 'State', 1);    % red
265     end
266
267     if u(2) == 1
268         voorschoterweg_2.set('AttValue', 'State', 3);
269     elseif (u(2) == 0) && (u_t(2,(i/(step_time*Ts))) == 1)
270         voorschoterweg_2.set('AttValue', 'State', 4);
271     else
272         voorschoterweg_2.set('AttValue', 'State', 1);
273     end
274
275     if u(3) == 1
276         parking_3.set('AttValue', 'State', 3);
277     elseif (u(3) == 0) && (u_t(3,(i/(step_time*Ts))) == 1)
278         parking_3.set('AttValue', 'State', 4);
279     else
280         parking_3.set('AttValue', 'State', 1);
281     end
282
283     if u(4) == 1
284         bus_4.set('AttValue', 'State', 3);
285     elseif (u(4) == 0) && (u_t(4,(i/(step_time*Ts))) == 1)
286         bus_4.set('AttValue', 'State', 4);
287     else
288         bus_4.set('AttValue', 'State', 1);
289     end
290
291     if u(5) == 1
292         centre_5.set('AttValue', 'State', 3);
293     elseif (u(5) == 0) && (u_t(5,(i/(step_time*Ts))) == 1)
294         centre_5.set('AttValue', 'State', 4);
295     else
296         centre_5.set('AttValue', 'State', 1);
297     end
298
299     if u(6) == 1
300         centre_6.set('AttValue', 'State', 3);
301     elseif (u(6) == 0) && (u_t(6,(i/(step_time*Ts))) == 1)
302         centre_6.set('AttValue', 'State', 4);
303     else
304         centre_6.set('AttValue', 'State', 1);
305     end
306
307     if u(7) == 1
308         centre_7.set('AttValue', 'State', 3);
309     elseif (u(7) == 0) && (u_t(7,(i/(step_time*Ts))) == 1)
310         centre_7.set('AttValue', 'State', 4);
311     else
```

```
312         centre_7.set('AttValue', 'State', 1);
313     end
314
315     if u(8) == 1
316         kanaalweg_8.set('AttValue', 'State', 3);
317     elseif (u(8) == 0) && (u_t(8,(i/(step_time*Ts))) == 1)
318         kanaalweg_8.set('AttValue', 'State', 4);
319     else
320         kanaalweg_8.set('AttValue', 'State', 1);
321     end
322
323     if u(9) == 1
324         kanaalweg_9.set('AttValue', 'State', 3);
325     elseif (u(9) == 0) && (u_t(9,(i/(step_time*Ts))) == 1)
326         kanaalweg_9.set('AttValue', 'State', 4);
327     else
328         kanaalweg_9.set('AttValue', 'State', 1);
329     end
330
331     if u(10) == 1
332         europaweg_10.set('AttValue', 'State', 3);
333     elseif (u(10) == 0) && (u_t(10,(i/(step_time*Ts))) == 1)
334         europaweg_10.set('AttValue', 'State', 4);
335     else
336         europaweg_10.set('AttValue', 'State', 1);
337     end
338
339     if u(11) == 1
340         europaweg_11.set('AttValue', 'State', 3);
341     elseif (u(11) == 0) && (u_t(11,(i/(step_time*Ts))) == 1)
342         europaweg_11.set('AttValue', 'State', 4);
343     else
344         europaweg_11.set('AttValue', 'State', 1);
345     end
346
347     if u(12) == 1
348         europaweg_12.set('AttValue', 'State', 3);
349     elseif (u(12) == 0) && (u_t(12,(i/(step_time*Ts))) == 1)
350         europaweg_12.set('AttValue', 'State', 4);
351     else
352         europaweg_12.set('AttValue', 'State', 1);
353     end
354
355
356     if u(13) == 1
357         industry_13.set('AttValue', 'State', 3);
358     elseif (u(13) == 0) && (u_t(13,(i/(step_time*Ts))) == 1)
359         industry_13.set('AttValue', 'State', 4);
360     else
361         industry_13.set('AttValue', 'State', 1);
362     end
363
364     if u(14) == 1
```

```
365         middle_14.set('AttValue', 'State', 3);
366     elseif (u(14) == 0) && (u_t(14,(i/(step_time*Ts))) == 1)
367         middle_14.set('AttValue', 'State', 4);
368     else
369         middle_14.set('AttValue', 'State', 1);
370     end
371
372     if u(15) == 1
373         middle_15.set('AttValue', 'State', 3);
374     elseif (u(15) == 0) && (u_t(15,(i/(step_time*Ts))) == 1)
375         middle_15.set('AttValue', 'State', 4);
376     else
377         middle_15.set('AttValue', 'State', 1);
378     end
379
380     if u(16) == 1
381         middle_16.set('AttValue', 'State', 3);
382     elseif (u(16) == 0) && (u_t(16,(i/(step_time*Ts))) == 1)
383         middle_16.set('AttValue', 'State', 4);
384     else
385         middle_16.set('AttValue', 'State', 1);
386     end
387
388     if u(17) == 1
389         middle_17.set('AttValue', 'State', 3);
390     elseif (u(17) == 0) && (u_t(17,(i/(step_time*Ts))) == 1)
391         middle_17.set('AttValue', 'State', 4);
392     else
393         middle_17.set('AttValue', 'State', 1);
394     end
395
396     if u(18) == 1
397         bridge_18.set('AttValue', 'State', 3);
398     elseif (u(18) == 0) && (u_t(18,(i/(step_time*Ts))) == 1)
399         bridge_18.set('AttValue', 'State', 4);
400     else
401         bridge_18.set('AttValue', 'State', 1);
402     end
403
404
405     %%% cycle controller
406     if u_t_c(1,(i/(step_time*Ts))+1) == 1
407         cycle_1.set('AttValue', 'State', 3);   % green
408     elseif (u_t_c(1,(i/(step_time*Ts))+1) == 0) && (u_t_c(1,(i/(step_time
        *Ts))) == 1)
409         cycle_1.set('AttValue', 'State', 4);   % amber
410     else
411         cycle_1.set('AttValue', 'State', 1);   % red
412     end
413
414     if u_t_c(2,(i/(step_time*Ts))+1) == 1
415         cycle_2.set('AttValue', 'State', 3);   % green
```

```
416     elseif (u_t_c(2,(i/(step_time*Ts))+1) == 0) && (u_t_c(2,(i/(step_time
            *Ts))) == 1)
417         cycle_2.set('AttValue', 'State', 4);   % amber
418     else
419         cycle_2.set('AttValue', 'State', 1);   % red
420     end
421
422     if u_t_c(3,(i/(step_time*Ts))+1) == 1
423         cycle_3.set('AttValue', 'State', 3);   % green
424     elseif (u_t_c(3,(i/(step_time*Ts))+1) == 0) && (u_t_c(3,(i/(step_time
            *Ts))) == 1)
425         cycle_3.set('AttValue', 'State', 4);   % amber
426     else
427         cycle_3.set('AttValue', 'State', 1);   % red
428     end
429
430     if u_t_c(4,(i/(step_time*Ts))+1) == 1
431         cycle_4.set('AttValue', 'State', 3);   % green
432     elseif (u_t_c(4,(i/(step_time*Ts))+1) == 0) && (u_t_c(4,(i/(step_time
            *Ts))) == 1)
433         cycle_4.set('AttValue', 'State', 4);   % amber
434     else
435         cycle_4.set('AttValue', 'State', 1);   % red
436     end
437
438     if u_t_c(5,(i/(step_time*Ts))+1) == 1
439         cycle_5.set('AttValue', 'State', 3);   % green
440     elseif (u_t_c(5,(i/(step_time*Ts))+1) == 0) && (u_t_c(5,(i/(step_time
            *Ts))) == 1)
441         cycle_5.set('AttValue', 'State', 4);   % amber
442     else
443         cycle_5.set('AttValue', 'State', 1);   % red
444     end
445
446     if u_t_c(6,(i/(step_time*Ts))+1) == 1
447         cycle_6.set('AttValue', 'State', 3);   % green
448     elseif (u_t_c(6,(i/(step_time*Ts))+1) == 0) && (u_t_c(6,(i/(step_time
            *Ts))) == 1)
449         cycle_6.set('AttValue', 'State', 4);   % amber
450     else
451         cycle_6.set('AttValue', 'State', 1);   % red
452     end
453
454     if u_t_c(7,(i/(step_time*Ts))+1) == 1
455         cycle_7.set('AttValue', 'State', 3);   % green
456     elseif (u_t_c(7,(i/(step_time*Ts))+1) == 0) && (u_t_c(7,(i/(step_time
            *Ts))) == 1)
457         cycle_7.set('AttValue', 'State', 4);   % amber
458     else
459         cycle_7.set('AttValue', 'State', 1);   % red
460     end
461
462     if u_t_c(8,(i/(step_time*Ts))+1) == 1
```

```
463         cycle_8.set('AttValue', 'State', 3);    % green
464     elseif (u_t_c(8,(i/(step_time*Ts))+1) == 0) && (u_t_c(8,(i/(step_time
            *Ts))) == 1)
465         cycle_8.set('AttValue', 'State', 4);    % amber
466     else
467         cycle_8.set('AttValue', 'State', 1);    % red
468     end
469
470     if u_t_c(9,(i/(step_time*Ts))+1) == 1
471         cycle_9.set('AttValue', 'State', 3);    % green
472     elseif (u_t_c(9,(i/(step_time*Ts))+1) == 0) && (u_t_c(9,(i/(step_time
            *Ts))) == 1)
473         cycle_9.set('AttValue', 'State', 4);    % amber
474     else
475         cycle_9.set('AttValue', 'State', 1);    % red
476     end
477
478     if u_t_c(10,(i/(step_time*Ts))+1) == 1
479         cycle_10.set('AttValue', 'State', 3);    % green
480     elseif (u_t_c(10,(i/(step_time*Ts))+1) == 0) && (u_t_c(10,(i/(
            step_time*Ts))) == 1)
481         cycle_10.set('AttValue', 'State', 4);    % amber
482     else
483         cycle_10.set('AttValue', 'State', 1);    % red
484     end
485
486     if u_t_c(11,(i/(step_time*Ts))+1) == 1
487         cycle_11.set('AttValue', 'State', 3);    % green
488     elseif (u_t_c(11,(i/(step_time*Ts))+1) == 0) && (u_t_c(11,(i/(
            step_time*Ts))) == 1)
489         cycle_11.set('AttValue', 'State', 4);    % amber
490     else
491         cycle_11.set('AttValue', 'State', 1);    % red
492     end
493
494     if u_t_c(12,(i/(step_time*Ts))+1) == 1
495         cycle_12.set('AttValue', 'State', 3);    % green
496     elseif (u_t_c(12,(i/(step_time*Ts))+1) == 0) && (u_t_c(12,(i/(
            step_time*Ts))) == 1)
497         cycle_12.set('AttValue', 'State', 4);    % amber
498     else
499         cycle_12.set('AttValue', 'State', 1);    % red
500     end
501
502     if u_t_c(13,(i/(step_time*Ts))+1) == 1
503         cycle_13.set('AttValue', 'State', 3);    % green
504     elseif (u_t_c(13,(i/(step_time*Ts))+1) == 0) && (u_t_c(13,(i/(
            step_time*Ts))) == 1)
505         cycle_13.set('AttValue', 'State', 4);    % amber
506     else
507         cycle_13.set('AttValue', 'State', 1);    % red
508     end
509
```

```
510     if u_t_c(14,(i/(step_time*Ts))+1) == 1
511         cycle_14.set('AttValue', 'State', 3);   % green
512     elseif (u_t_c(14,(i/(step_time*Ts))+1) == 0) && (u_t_c(14,(i/(
            step_time*Ts))) == 1)
513         cycle_14.set('AttValue', 'State', 4);   % amber
514     else
515         cycle_14.set('AttValue', 'State', 1);   % red
516     end
517
518     if u_t_c(15,(i/(step_time*Ts))+1) == 1
519         cycle_15.set('AttValue', 'State', 3);   % green
520     elseif (u_t_c(15,(i/(step_time*Ts))+1) == 0) && (u_t_c(15,(i/(
            step_time*Ts))) == 1)
521         cycle_15.set('AttValue', 'State', 4);   % amber
522     else
523         cycle_15.set('AttValue', 'State', 1);   % red
524     end
525
526     if u_t_c(16,(i/(step_time*Ts))+1) == 1
527         cycle_16.set('AttValue', 'State', 3);   % green
528     elseif (u_t_c(16,(i/(step_time*Ts))+1) == 0) && (u_t_c(16,(i/(
            step_time*Ts))) == 1)
529         cycle_16.set('AttValue', 'State', 4);   % amber
530     else
531         cycle_16.set('AttValue', 'State', 1);   % red
532     end
533
534     if u_t_c(17,(i/(step_time*Ts))+1) == 1
535         cycle_17.set('AttValue', 'State', 3);   % green
536     elseif (u_t_c(17,(i/(step_time*Ts))+1) == 0) && (u_t_c(17,(i/(
            step_time*Ts))) == 1)
537         cycle_17.set('AttValue', 'State', 4);   % amber
538     else
539         cycle_17.set('AttValue', 'State', 1);   % red
540     end
541
542     if u_t_c(18,(i/(step_time*Ts))+1) == 1
543         cycle_18.set('AttValue', 'State', 3);   % green
544     elseif (u_t_c(18,(i/(step_time*Ts))+1) == 0) && (u_t_c(18,(i/(
            step_time*Ts))) == 1)
545         cycle_18.set('AttValue', 'State', 4);   % amber
546     else
547         cycle_18.set('AttValue', 'State', 1);   % red
548     end
549
550
551  end
552
553 end
```

B-2 Lammeplein_system

This code, provided in the listing, sets up the system model and calibrates it. The model is used for state predictions in the MPC function.

Listing B.2: Lammeplein model

```
%% setting up the system

% departure rate 'a' per lane (initial guess)
l = 6;  % average car length
a1 = -2.5*l;
a2 = -2.5*l;
a3 = -2.5*l;
a4 = -2.5*l;
a5 = -2.5*l;
a6 = -2.5*l;
a7 = -2.5*l;
a8 = -2.5*l;
a9 = -2.5*l;
a10 = -2.5*l;
a11 = -2.5*l;
a12 = -2.5*l;
a13 = -2.5*l;
a14 = -2.5*l;
a15 = -2.5*l;
a16 = -2.5*l;
a17 = -2.5*l;
a18 = -2.5*l;

% Inflow 'd' per lane based on 2h simulation data
d1 = 1604/1440*l;
d2 = 313/1440*l;
d3 = 19/1440*l;
d4 = 33/1440*l;
d5 = 236/1440*l;
d6 = 643/1440*l;
d7 = 11/1440*l;
d8 = 26/1440*l;
d9 = 472/1440*l;
d10 = 0*l;
d11 = 0*l;
d12 = 0*l;
d13 = 4/1440*l;
d14 = 0*l;
d15 = 0*l;
d16 = 0*l;
d17 = 0*l;
d18 = 2326/1440*l;

% Turn fractions 'p' of connected lanes
p3_14 = 0.1;
p5_14 = 0.85;
```

```
47 p9_14 = 0.5;
48 p3_15 = 0.85;
49 p9_15 = 0.5;
50 p2_16 = 0.66;
51 p10_16 = 0.66;
52 p13_16 = 0.17;
53 p2_17 = 0.34;
54 p10_17 = 0.34;
55 p13_17 = 0.08;
56 p18_10 = 0.237;
57 p18_11 = 0.76;
58 p18_12 = 0.003;
59
60 % system matrices
61 A = eye(18);
62 B = [diag([a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14,a15,a16,a17,a18
       ]) [d1 d2 d3 d4 d5 d6 d7 d8 d9 d10 d11 d12 d13 d14 d15 d16 d17 d18]'];
63
64 % Inclusion of connected links traffic flow in B matrix (for midstream
       lanes)
65 B(14,3) = -p3_14*a3;
66 B(14,4) = -a4;
67 B(14,5) = -p5_14*a5;
68 B(14,9) = -p9_14*a9;
69
70 B(15,3) = -p3_15*a3;
71 B(15,6) = -a6;
72 B(15,9) = -p9_15*a9;
73
74 B(16,2) = -p2_16*a2;
75 B(16,10) = -p10_16*a10;
76 B(16,13) = -p13_16*a13;
77
78 B(17,2) = -p2_17*a2;
79 B(17,10) = -p10_17*a10;
80 B(17,13) = -p13_17*a13;
81
82 B(10,18) = -p18_10*a18;
83 B(11,18) = -p18_11*a18;
84 B(12,18) = -p18_12*a18;
85
86 %% Constraints
87
88 % state constraints
89 x_min = [-5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -5 -100]' ;
90 x_max = 1*[999 999 15 999 999 999 999 999 999 999 999 999 15 15 15 15 15
       999]' ;
91
92 % input constraints
93 u_min = [-0.1*ones(18,1); 0.9];
94 u_max = 1.1*ones(19,1);
95
96 % open bridge input restricictions
```

```
97  u_max_b = u_max;
98  u_max_b(1) = 0.1;
99  u_max_b(13) = 0.1;
100 u_max_b(15) = 0.1;
101 u_max_b(6) = 0.1;
102 u_max_b(9) = 0.1;
103 u_max_b(18) = 0.1;
104
105 % conflicting areas (Uc stands for u conflicting)
106 Uc = zeros(36,length(u_max));
107 Uc(1,1) = 1;
108 Uc(1,12) = 1;
109 Uc(2,1) = 1;
110 Uc(2,13) = 1;
111 Uc(3,1) = 1;
112 Uc(3,15) = 1;
113 Uc(4,2) = 1;
114 Uc(4,10) = 1;
115 Uc(5,2) = 1;
116 Uc(5,11) = 1;
117 Uc(6,2) = 1;
118 Uc(6,13) = 1;
119 Uc(7,2) = 1;
120 Uc(7,15) = 1;
121 Uc(8,3) = 1;
122 Uc(8,4) = 1;
123 Uc(9,3) = 1;
124 Uc(9,5) = 1;
125 Uc(10,3) = 1;
126 Uc(10,6) = 1;
127 Uc(11,3) = 1;
128 Uc(11,7) = 1;
129 Uc(12,3) = 1;
130 Uc(12,8) = 1;
131 Uc(13,3) = 1;
132 Uc(13,9) = 1;
133 Uc(14,3) = 1;
134 Uc(14,16) = 1;
135 Uc(15,3) = 1;
136 Uc(15,17) = 1;
137 Uc(16,4) = 1;
138 Uc(16,5) = 1;
139 % 4 and 6 also conflicting (was added later)
140 Uc(36,4) = 1;
141 Uc(36,6) = 1;
142
143 Uc(17,4) = 1;
144 Uc(17,9) = 1;
145 Uc(18,4) = 1;
146 Uc(18,16) = 1;
147 Uc(19,5) = 1;
148 Uc(19,9) = 1;
149 Uc(20,5) = 1;
```

```
150 Uc(20,16) = 1;
151 Uc(21,6) = 1;
152 Uc(21,9) = 1;
153 Uc(22,6) = 1;
154 Uc(22,16) = 1;
155 Uc(23,7) = 1;
156 Uc(23,9) = 1;
157 Uc(24,7) = 1;
158 Uc(24,16) = 1;
159 Uc(25,7) = 1;
160 Uc(25,17) = 1;
161 Uc(26,8) = 1;
162 Uc(26,16) = 1;
163 Uc(27,9) = 1;
164 Uc(27,16) = 1;
165 Uc(28,10) = 1;
166 Uc(28,13) = 1;
167 Uc(29,11) = 1;
168 Uc(29,13) = 1;
169 Uc(30,11) = 1;
170 Uc(30,14) = 1;
171 Uc(31,11) = 1;
172 Uc(31,15) = 1;
173 Uc(32,12) = 1;
174 Uc(32,13) = 1;
175 Uc(33,12) = 1;
176 Uc(33,15) = 1;
177 Uc(34,13) = 1;
178 Uc(34,14) = 1;
179 Uc(35,13) = 1;
180 Uc(35,15) = 1;
181
182 % Introduction short all red phase (arp) of conflicting signals
183 Uc_arp = zeros(72, 2*length(u_max));
184
185 Uc_arp(1,1) = 1;
186 Uc_arp(1,12+length(B)) = 1;
187 Uc_arp(2,1+length(B)) = 1;
188 Uc_arp(2,12) = 1;
189
190 Uc_arp(3,1) = 1;
191 Uc_arp(3,13+length(B)) = 1;
192 Uc_arp(4,1+length(B)) = 1;
193 Uc_arp(4,13) = 1;
194
195 Uc_arp(5,1) = 1;
196 Uc_arp(5,15+length(B)) = 1;
197 Uc_arp(6,1+length(B)) = 1;
198 Uc_arp(6,15) = 1;
199
200 Uc_arp(7,2) = 1;
201 Uc_arp(7,10+length(B)) = 1;
202 Uc_arp(8,2+length(B)) = 1;
```

```
203 Uc_arp(8,10) = 1;
204
205 Uc_arp(9,2) = 1;
206 Uc_arp(9,11+length(B)) = 1;
207 Uc_arp(10,2+length(B)) = 1;
208 Uc_arp(10,11) = 1;
209
210 Uc_arp(11,2) = 1;
211 Uc_arp(11,13+length(B)) = 1;
212 Uc_arp(12,2+length(B)) = 1;
213 Uc_arp(12,13) = 1;
214
215 Uc_arp(13,2) = 1;
216 Uc_arp(13,15+length(B)) = 1;
217 Uc_arp(14,2+length(B)) = 1;
218 Uc_arp(14,15) = 1;
219
220 Uc_arp(15,3) = 1;
221 Uc_arp(15,4+length(B)) = 1;
222 Uc_arp(16,3+length(B)) = 1;
223 Uc_arp(16,4) = 1;
224
225 Uc_arp(17,3) = 1;
226 Uc_arp(17,5+length(B)) = 1;
227 Uc_arp(18,3+length(B)) = 1;
228 Uc_arp(18,5) = 1;
229
230 Uc_arp(19,3) = 1;
231 Uc_arp(19,6+length(B)) = 1;
232 Uc_arp(20,3+length(B)) = 1;
233 Uc_arp(20,6) = 1;
234
235 Uc_arp(21,3) = 1;
236 Uc_arp(21,7+length(B)) = 1;
237 Uc_arp(22,3+length(B)) = 1;
238 Uc_arp(22,7) = 1;
239
240 Uc_arp(23,3) = 1;
241 Uc_arp(23,8+length(B)) = 1;
242 Uc_arp(24,3+length(B)) = 1;
243 Uc_arp(24,8) = 1;
244
245 Uc_arp(25,3) = 1;
246 Uc_arp(25,9+length(B)) = 1;
247 Uc_arp(26,3+length(B)) = 1;
248 Uc_arp(26,9) = 1;
249
250 Uc_arp(27,3) = 1;
251 Uc_arp(27,16+length(B)) = 1;
252 Uc_arp(28,3+length(B)) = 1;
253 Uc_arp(28,16) = 1;
254
255 Uc_arp(29,3) = 1;
```

```
256 Uc_arp(29,17+length(B)) = 1;
257 Uc_arp(30,3+length(B)) = 1;
258 Uc_arp(30,17) = 1;
259
260 Uc_arp(31,4) = 1;
261 Uc_arp(31,5+length(B)) = 1;
262 Uc_arp(32,4+length(B)) = 1;
263 Uc_arp(32,5) = 1;
264
265 % set bus and lane 6 also on conflict
266 Uc_arp(71,4) = 0;
267 Uc_arp(71,6+length(B)) = 0;
268 Uc_arp(72,4+length(B)) = 0;
269 Uc_arp(72,6) = 0;
270
271 Uc_arp(33,4) = 1;
272 Uc_arp(33,9+length(B)) = 1;
273 Uc_arp(34,4+length(B)) = 1;
274 Uc_arp(34,9) = 1;
275
276 Uc_arp(35,4) = 1;
277 Uc_arp(35,16+length(B)) = 1;
278 Uc_arp(36,4+length(B)) = 1;
279 Uc_arp(36,16) = 1;
280
281 Uc_arp(37,5) = 1;
282 Uc_arp(37,9+length(B)) = 1;
283 Uc_arp(38,5+length(B)) = 1;
284 Uc_arp(38,9) = 1;
285
286 Uc_arp(39,5) = 1;
287 Uc_arp(39,16+length(B)) = 1;
288 Uc_arp(40,5+length(B)) = 1;
289 Uc_arp(40,16) = 1;
290
291 Uc_arp(41,6) = 1;
292 Uc_arp(41,9+length(B)) = 1;
293 Uc_arp(42,6+length(B)) = 1;
294 Uc_arp(42,9) = 1;
295
296 Uc_arp(43,6) = 1;
297 Uc_arp(43,16+length(B)) = 1;
298 Uc_arp(44,6+length(B)) = 1;
299 Uc_arp(44,16) = 1;
300
301 Uc_arp(45,7) = 1;
302 Uc_arp(45,9+length(B)) = 1;
303 Uc_arp(46,7+length(B)) = 1;
304 Uc_arp(46,9) = 1;
305
306 Uc_arp(47,7) = 1;
307 Uc_arp(47,16+length(B)) = 1;
308 Uc_arp(48,7+length(B)) = 1;
```

```
309 Uc_arp(48,16) = 1;
310
311 Uc_arp(49,7) = 1;
312 Uc_arp(49,17+length(B)) = 1;
313 Uc_arp(50,7+length(B)) = 1;
314 Uc_arp(50,17) = 1;
315
316 Uc_arp(51,8) = 1;
317 Uc_arp(51,16+length(B)) = 1;
318 Uc_arp(52,8+length(B)) = 1;
319 Uc_arp(52,16) = 1;
320
321 Uc_arp(53,9) = 1;
322 Uc_arp(53,16+length(B)) = 1;
323 Uc_arp(54,9+length(B)) = 1;
324 Uc_arp(54,16) = 1;
325
326 Uc_arp(55,10) = 1;
327 Uc_arp(55,13+length(B)) = 1;
328 Uc_arp(56,10+length(B)) = 1;
329 Uc_arp(56,13) = 1;
330
331 Uc_arp(57,11) = 1;
332 Uc_arp(57,13+length(B)) = 1;
333 Uc_arp(58,11+length(B)) = 1;
334 Uc_arp(58,13) = 1;
335
336 Uc_arp(59,11) = 1;
337 Uc_arp(59,14+length(B)) = 1;
338 Uc_arp(60,11+length(B)) = 1;
339 Uc_arp(60,14) = 1;
340
341 Uc_arp(61,11) = 1;
342 Uc_arp(61,15+length(B)) = 1;
343 Uc_arp(62,11+length(B)) = 1;
344 Uc_arp(62,15) = 1;
345
346 Uc_arp(63,12) = 1;
347 Uc_arp(63,13+length(B)) = 1;
348 Uc_arp(64,12+length(B)) = 1;
349 Uc_arp(64,13) = 1;
350
351 Uc_arp(65,12) = 1;
352 Uc_arp(65,15+length(B)) = 1;
353 Uc_arp(66,12+length(B)) = 1;
354 Uc_arp(66,15) = 1;
355
356 Uc_arp(67,13) = 1;
357 Uc_arp(67,14+length(B)) = 1;
358 Uc_arp(68,13+length(B)) = 1;
359 Uc_arp(68,14) = 1;
360
361 Uc_arp(69,13) = 1;
```

```
362 Uc_arp(69,15+length(B)) = 1;
363 Uc_arp(70,13+length(B)) = 1;
364 Uc_arp(70,15) = 1;
365
366
367
368 %% Calibrate system
369
370 load('20h_sim_run_results.mat')
371 ut = u_t(:,141:end);
372 xt = x_t(:,142:end);              % y
373 xm = A*x_t(:,141:end-1) + B*ut;  % x
374
375 U = zeros(size(B,1),numel(B),size(ut,2));
376 Hc = zeros(size(U,2),size(U,2));
377 fc = zeros(1,numel(B))';
378
379 for i = 1:size(ut,2)
380     U(:,:,i) = kron(eye(size(B,1)),ut(:,i)');
381     Hc = Hc + U(:,:,i)'*U(:,:,i);
382     fc = fc + (xm(:,i)'*A'*U(:,:,i)-xt(:,i)'*U(:,:,i))';
383 end
384
385 Aineq1 = zeros(18,numel(B));
386 Aineq2 = zeros(13,numel(B));
387 Aineq3 = zeros(13,numel(B));
388 Aineq4 = zeros(18,numel(B));
389 bineq1 = -10*ones(18,1);
390 bineq2 = -2*ones(13,1);
391 bineq3 = zeros(13,1);
392 bineq4 = 20*ones(18,1);
393
394 Aineq1(1,1) = 1;
395 Aineq1(2,21) = 1;
396 Aineq1(3, 41) = 1;
397 Aineq1(4, 61) = 1;
398 Aineq1(5, 81) = 1;
399 Aineq1(6, 101) = 1;
400 Aineq1(7, 121) = 1;
401 Aineq1(8, 141) = 1;
402 Aineq1(9, 161) = 1;
403 Aineq1(10, 181) = 1;
404 Aineq1(11, 201) = 1;
405 Aineq1(12, 221) = 1;
406 Aineq1(13, 241) = 1;
407 Aineq1(14, 261) = 1;
408 Aineq1(15, 281) = 1;
409 Aineq1(16, 301) = 1;
410 Aineq1(17, 321) = 1;
411 Aineq1(18, 341) = 1;
412
413 Aineq2(1,41) = 1;
414 Aineq2(1,250) = 1;
```

```
Aineq2(2,61) = 1;
Aineq2(2,251) = 1;
Aineq2(3,81) = 1;
Aineq2(3,252) = 1;
Aineq2(4,161) = 1;
Aineq2(4,256) = 1;
Aineq2(5,41) = 1;
Aineq2(5,269) = 1;
Aineq2(6,101) = 1;
Aineq2(6,272) = 1;
Aineq2(7,161) = 1;
Aineq2(7,255) = 1;
Aineq2(8,21) = 1;
Aineq2(8,287) = 1;
Aineq2(9,181) = 1;
Aineq2(9,295) = 1;
Aineq2(10,241) = 1;
Aineq2(10,298) = 1;
Aineq2(11,21) = 1;
Aineq2(11,306) = 1;
Aineq2(12,181) = 1;
Aineq2(12,314) = 1;
Aineq2(13,241) = 1;
Aineq2(13,317) = 1;

Aineq3(1,250) = -1;
Aineq3(2,251) = -1;
Aineq3(3,252) = -1;
Aineq3(4,256) = -1;
Aineq3(5,269) = -1;
Aineq3(6,272) = -1;
Aineq3(7,255) = -1;
Aineq3(8,287) = -1;
Aineq3(9,295) = -1;
Aineq3(10,298) = -1;
Aineq3(11,306) = -1;
Aineq3(12,314) = -1;
Aineq3(13,317) = -1;

Aineq4(1,1) = -1;
Aineq4(2,21) = -1;
Aineq4(3, 41) = -1;
Aineq4(4, 61) = -1;
Aineq4(5, 81) = -1;
Aineq4(6, 101) = -1;
Aineq4(7, 121) = -1;
Aineq4(8, 141) = -1;
Aineq4(9, 161) = -1;
Aineq4(10, 181) = -1;
Aineq4(11, 201) = -1;
Aineq4(12, 221) = -1;
Aineq4(13, 241) = -1;
Aineq4(14, 261) = -1;
```

```
Aineq4(15, 281) = -1;
Aineq4(16, 301) = -1;
Aineq4(17, 321) = -1;
Aineq4(18, 341) = -1;

Aineq = [Aineq1; Aineq2; Aineq3; Aineq4];
bineq = [bineq1; bineq2; bineq3; bineq4];

Aeq = eye(numel(B)); % remove 1, 21, 38, 41, 57, 61, 76, 81, 95, 101,
    114, 121, 133, 141, 152, 161, 171, 181, 189, 201, 208, 221, 227, 241,
    247, 250, 251, 252, 256, 261, 266, 269, 272, 275, 281, 287, 295, 298,
    301, 306, 314, 317, 321, 341, 342
% Define the indices of rows to remove
indices_to_remove = [1, 21, 38, 41, 57, 61, 76, 81, 95, 101, 114, 121,
    133, 141, 152, 161, 171, 181, 189, 201, 208, 221, 227, 241, 247, 250,
    251, 252, 256, 261, 266, 269, 272, 275, 281, 287, 295, 298, 301, 306,
    314, 317, 321, 341, 342];
% Create a logical index to keep rows that are not in indices_to_remove
logical_index = true(numel(B), 1);
logical_index(indices_to_remove) = false;

% Remove specified rows from Aeq
Aeq = Aeq(logical_index, :);
beq = zeros(size(Aeq,1),1);

% new B
B_entries = quadprog(Hc,fc,Aineq,bineq,Aeq,beq);
B_new = reshape(B_entries, size(B,2), size(B,1))';

%% Controller settings

% Calibrated  B
B = B_new;

% Sampling time and horizon
Ts = 5;
H = 12;

% prediction matrices
T = zeros(length(A)*(H+1),length(A));
for i = 1:H+1
    T(length(A)*(i-1)+1:length(A)*i,:) = A^(i-1);
end

S = zeros(length(A)*(H+1),length(B)*(H));

S1 = [zeros(size(B)); T(1:end-length(A),:)*B];

for i = 1:H
    S(length(A)*(i-1)+1:end,length(B)*(i-1)+1:length(B)*i) = S1(1:end-
        length(A)*(i-1),:);
end

```

```
514 %% COCON controller
515
516 % closed bridge:
517 uc1 = [ones(1,60/Ts) zeros(1,50/Ts)];
518 uc2 = [zeros(1,45/Ts) ones(1,15/Ts) zeros(1,50/Ts)];
519 uc3 = [zeros(1,20/Ts) ones(1,5/Ts) zeros(1,85/Ts)];
520 uc4 = [zeros(1,30/Ts) ones(1,30/Ts) zeros(1,50/Ts)];
521 uc5 = [zeros(1,65/Ts) ones(1,15/Ts) zeros(1,30/Ts)];
522 uc6 = [zeros(1,65/Ts) ones(1,15/Ts) zeros(1,30/Ts)];
523 uc7 = [zeros(1,65/Ts) ones(1,15/Ts) zeros(1,30/Ts)];
524 uc8 = [ones(1,15/Ts) zeros(1,40/Ts) ones(1,55/Ts)];
525 uc9 = [ones(1,15/Ts) zeros(1,70/Ts) ones(1,25/Ts)];
526 uc10 = [ones(1,40/Ts) zeros(1,25/Ts) ones(1,30/Ts) zeros(1,15/Ts)];
527 uc11 = [ones(1,40/Ts) zeros(1,70/Ts)];
528 uc12 = [zeros(1,90/Ts) ones(1,5/Ts) zeros(1,15/Ts)];
529 uc13 = [zeros(1,100/Ts) ones(1,5/Ts) zeros(1,5/Ts)];
530 uc14 = [zeros(1,50/Ts) ones(1,45/Ts) zeros(1,15/Ts)];
531 uc15 = [zeros(1,65/Ts) ones(1,20/Ts) zeros(1,25/Ts)];
532 uc16 = [zeros(1,35/Ts) ones(1,25/Ts) zeros(1,50/Ts)];
533 uc17 = [ones(1,20/Ts) zeros(1,15/Ts) ones(1,25/Ts) zeros(1,25/Ts) ones
        (1,25/Ts)];
534 uc18 = [ones(1,110/Ts)];
535 uc19 = [ones(1,110/Ts)];
536
537 UC = [uc1; uc2; uc3; uc4; uc5; uc6; uc7; uc8; uc9; uc10; uc11; uc12; uc13
        ; uc14; uc15; uc16; uc17; uc18; uc19];
538 cycles = ceil(Tmax/length(uc1));
539
540 % 1 cycle of traffic signals during a normal situation (closed bridge)
541 full_cb = [kron(ones(1,cycles), UC)];
542
543 % open bridge
544 ub1 = [0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 1]';
545 ub2 = [0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1]';
546 ub3 = [0 0 0 1 0 0 1 1 0 0 0 0 0 1 0 0 0 0 1]';
547 ub4 = [0 0 0 0 1 0 1 1 0 0 0 0 0 1 0 0 0 0 1]';
548 ub_arp = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]';
549
550 % 1 cycle of traffic signals during a abnormal situation (open bridge)
551 cycle_ob = [kron(ones(1,12),ub1) ub_arp kron(ones(1,2),ub3) ub_arp kron(
        ones(1,3),ub4) ub_arp kron(ones(1,1),ub2) ub_arp];
552 full_ob = kron(ones(1,ceil(Tmax/length(cycle_ob(1,:)))),cycle_ob);
553
554 x_t_c = zeros(length(A),Tmax);
555 x_t_c(:,1) = x0;
556 u_t_c = zeros(length(B),Tmax);
557
558 for i = 1:Tmax-1
559     if (t_bo <= i) && (i <= t_bo + bridgetime)
560         u_t_c(:,i) = full_ob(:,i-t_bo+1);
561     elseif t_bo > i
562         u_t_c(:,i) = full_cb(:,i);
563     else
```

```
564         u_t_c(:,i) = full_cb(:,i-(t_bo + bridgetime)+1);
565     end
566 end
```

B-3 MPC_full

The following listing provides the function that is the actual controller. The bridge knowledge is incorporated by the lines 10-20.

Listing B.3: MPC Full Function

```
1
2  function [x,u] = MPC_full(A, B, x0, u0, S, T, Uc, Uc_arp, x_min, x_max,
       u_min, u_max, u_max_b, H, t_bo, bridgetime, i)
3
4  % create constraint on all red phase (arp)
5  ARP = zeros(length(Uc_arp)*(H-1), length(S(1,:)));
6  for j = 1:H-1
7      ARP(length(Uc_arp)*(j-1)+1:length(Uc_arp)*j,length(B)*(j-1)+1:length(
           B)+length(B)*j) = Uc_arp;
8  end
9
10 if i <= t_bo - H
11     max_inputs = kron(ones(H,1),u_max);
12 elseif (t_bo - H < i) && (i <= t_bo)
13     max_inputs = [kron(ones(t_bo-i,1),u_max); kron(ones(H-(t_bo-i),1),
           u_max_b)];
14 elseif (t_bo < i) && (i <= t_bo + bridgetime - H)
15     max_inputs = kron(ones(H,1),u_max_b);
16 elseif (t_bo + bridgetime - H < i) && (i <= t_bo + bridgetime)
17     max_inputs = [kron(ones(t_bo+bridgetime-i,1),u_max_b); kron(ones(H-(
           t_bo+bridgetime-i),1),u_max)];
18 else
19     max_inputs = kron(ones(H,1),u_max);
20 end
21
22 A_c = [S;                  % state
23     -S;                    % state
24     eye(length(B)*H);           % input
25     -eye(length(B)*H);          % input
26     kron(eye(H),Uc);       % conflicting area
27     ARP;                   % all red phase for conflict areas
28     Uc_arp(:,length(u_min)+1:2*length(B)) zeros(length(Uc_arp),(H-1)*
           length(B))];   % all red with previous input incorporated
29
30 b_c = [kron(ones(H+1,1),x_max)-T*x0;
31     T*x0 - kron(ones(H+1,1),x_min);
32     max_inputs;
33     kron(ones(H,1),-u_min);
34     ones(H*length(Uc(:,1)),1);
35     ones(length(Uc_arp)*(H-1),1);
```

```
36      ones(length(Uc_arp),1) - Uc_arp(:,1:length(B))*u0];
37
38  intcon = 1:H*length(u_min); % define integer inputs (all)
39
40  w = diag([1 2 2 5 1.5 1.5 0.5 0.5 1.5 0.5 2 0.5 2 0.5 2.5 1.5 0.5 5.01
        2.5 1.5 1 2 1 4 1 1.5 2]); % weight matrix
41
42  Sw = kron(eye(H+1),w)*S;
43
44  ft = sum(Sw, 1);
45  options = optimoptions('intlinprog', 'MaxNodes', 1500000);
46  u_t = intlinprog(ft',intcon,A_c,b_c,[],[],[],[],options);
47  u = u_t(1:length(B));
48  x = A*x0 + B*u;
49
50  end
```

Appendix C

VISSIM model

The VISSIM model is provided in Figure C-1. In the later stages of the research an upstream intersection was added. The full model, including upstream intersection, can be found in Figure C-2. In both figures, some colours can be identified, these are the following:

- **dark black**: At the beginning of lanes, there are vehicle inputs, which are thick black lines.
- **Purple:** These thick purple/pink lines, located between vehicle inputs and the intersection, represent vehicle routing decisions. Vehicles have a destination indicated in a thick blue line.
- **Dark red:** Traffic signal installations are indicated in dark red. Right next to one, it is often possible to find a queue counter, indicated in pink.
- The quadrilateral around the junction is called a node. It collects data, like average delay per road user.

Figure C-1: VISSIM model of the Lammebrug junction.

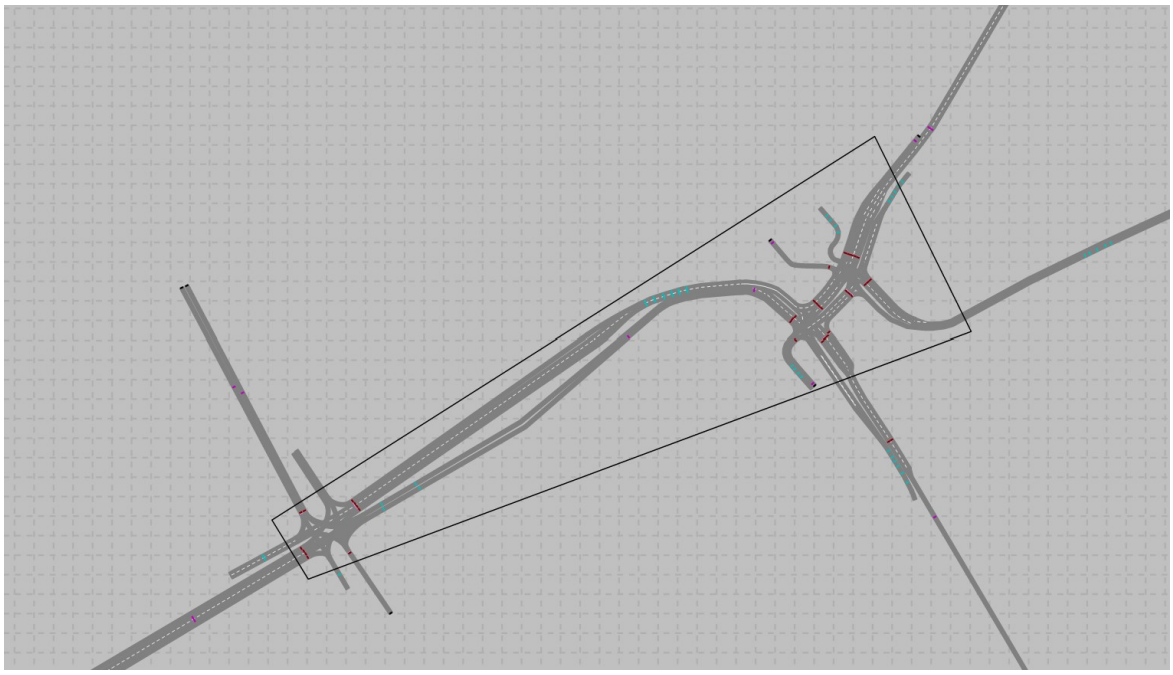

Figure C-2: VISSIM model of the Lammebrug junction with upstream intersection.

Bibliography

[1] Francesco Abbracciavento, Francesco Zinnari, Simone Formentin, Andrea G Bianchessi, and Sergio M Savaresi. Multi-intersection traffic signal control: A decentralized mpc-based approach. *IFAC Journal of Systems and Control*, 23:100214, 2023.

[2] Tobias Achterberg, Timo Berthold, Thorsten Koch, and Kati Wolter. Constraint integer programming: A new approach to integrate CP and MIP. In *International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming*, pages 6–20. Springer, 2008.

[3] Alessandro Alessio and Alberto Bemporad. A survey on explicit model predictive control. *Nonlinear Model Predictive Control: Towards New Challenging Applications*, pages 345–369, 2009.

[4] Nicola Bellomo and Christian Dogbe. On the modeling of traffic and crowds: A survey of models, speculations, and perspectives. *SIAM review*, 53(3):409–463, 2011.

[5] You-Ren Chen, Keng-Pin Chen, and Pao-Ann Hsiungy. Dynamic traffic light optimization and control system using model-predictive control method. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 2366–2371. IEEE, 2016.

[6] Lucas Barcelos De Oliveira and Eduardo Camponogara. Multi-agent model predictive control of signaling split in urban traffic networks. *Transportation Research Part C: Emerging Technologies*, 18(1):120–139, 2010.

[7] Nathan H Gartner. *OPAC: A demand-responsive strategy for traffic signal control*. TRID, 1983.

[8] Google Maps. Google maps. https://www.google.com/maps/@52.155737,4.4702879,11.75z?entry=ttu. Accessed: May 8, 2024.

[9] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2021.

[10] Jean-Jacques Henry, Jean Loup Farges, and Jean Tuffal. The PRODYN real time traffic algorithm. In *Control in transportation systems*, pages 305–310. Elsevier, 1984.

[11] PB Hunt, DI Robertson, RD Bretherton, and M Cr Royle. The SCOOT on-line traffic signal optimisation technique. *Traffic Engineering & Control*, 23(4), 1982.

[12] Huo Jie, Irfan Khan, Majed Alharthi, Muhammad Wasif Zafar, and Asif Saeed. Sustainable energy policy, socio-economic development, and ecological footprint: The economic significance of natural resources, population growth, and industrial development. *Utilities Policy*, 81:101490, 2023.

[13] MAS Kamal, J-i Imura, A Ohata, T Hayakawa, and K Aihara. Control of traffic signals in a model predictive control framework. *IFAC Proceedings Volumes*, 45(24):221–226, 2012.

[14] MAS Kamal, J-i Imura, A Ohata, T Hayakawa, and K Aihara. Control of traffic signals in a model predictive control framework. *IFAC Proceedings Volumes*, 45(24):221–226, 2012.

[15] Amal Kumarage. Urban traffic congestion: The problem & solutions. *The Asian economic review*, 01 2004.

[16] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. Microscopic traffic simulation using SUMO. In *2018 21st international conference on intelligent transportation systems (ITSC)*, pages 2575–2582. IEEE, 2018.

[17] Reinhard Madlener and Yasin Sunak. Impacts of urbanization on urban structures and energy demand: What can we learn for urban energy planning and urbanization management? *Sustainable Cities and Society*, 1(1):45–53, 2011.

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

[19] ApS Mosek. The MOSEK optimization toolbox for MATLAB manual, 2015.

[20] Anna Nagurney. Congested urban transportation networks and emission paradoxes. *Transportation Research Part D: Transport and Environment*, 5(2):145–151, 2000.

[21] Hiroaki Nakanishi and Toru Namerikawa. Optimal traffic signal control for alleviation of congestion based on traffic density prediction by model predictive control. In *2016 55th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, pages 1273–1278. IEEE, 2016.

[22] Michael Nikolaou. Model predictive controllers: A critical synthesis of theory and industrial needs. *Sciencedirect*, 2001.

[23] Bob Pishue. Global traffic scorecard. *Inrix*, Jan 2023.

[24] Provincie Zuid-Holland. N206. https://www.zuid-holland.nl/onderwerpen/verkeer-vervoer/wegverkeer/alle-n-wegen/wegen/n206/. Accessed: September, 2023.

[25] PTV Group. PTV VISSIM, 1992. Developed by PTV Group, Karlsruhe, Germany.

[26] Angus Eugene Retallack and Bertram Ostendorf. Current understanding of the effects of congestion on traffic accidents. *International journal of environmental research and public health*, 16(18):3400, 2019.

[27] Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[28] Arthur G Sims and Kenneth W Dobinson. The sydney coordinated adaptive traffic (SCAT) system philosophy and benefits. *IEEE Transactions on vehicular technology*, 29(2):130–137, 1980.

[29] Inc. The MathWorks. *MATLAB Optimization Toolbox: Mixed-Integer Linear Programming Solver*. Natick, Massachusetts, 2024. Version R2024a.

[30] Goof Sterk van de Weg, Mehdi Keyvan-Ekbatani, Andreas Hegyi, and Serge Paul Hoogendoorn. Linear MPC-based urban traffic control using the link transmission model. *IEEE Transactions on Intelligent Transportation Systems*, 21(10):4133–4148, 2019.

[31] Andreas Warberg, Jesper Larsen, and Rene Munk Jørgesen. *Green wave traffic optimization-a survey*. Informatics and Mathematical Modeling, Technical University of Denmark, 2008.

[32] Fo Vo Webster. Traffic signal settings. Technical report, Transportation research board, 1958.

[33] A Wilson and F Middelham. COCON: A connected packet of programmes for the design of traffic control. conference papers on the working days on traffic engineering 1989. *Crow Publikatie 23*, 1989.

[34] Bao-Lin Ye, Weimin Wu, Huimin Gao, Yixia Lu, Qianqian Cao, and Lijun Zhu. Stochastic model predictive control for urban traffic networks. *Applied Sciences*, 7(6):588, 2017.