Traffic Routing under Dynamic Network Topologies

M. Leeuwenberg

Master of Science Thesis

Delft University of Technology

Delft Center for Systems and Control

Traffic Routing under Dynamic Network Topologies

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

M. Leeuwenberg

December 28, 2021

Faculty of Mechanical, Maritime and Materials Engineering · Delft University of Technology

Abstract

Inefficient road usage in a traffic network context – i.e. over-saturation on one route, where other roads are still available – is an important problem. It appears often and in different types of situations. We concentrate on congestion caused by predicted temporary road blockades, such as open bridges. This research aims to reduce such congestion by focussing on solving a routing problem that accounts for these road blockades.

More specifically, we consider traffic that is to be guided through a network with a number of different routes, where bridges function as temporary blockades when they are opened. A river that is used by freight transport runs through this network, in which road traffic uses bridges to cross the river. These bridges would open to let the freight ships pass. In such a situation, open bridges are a predictable temporary blockade for the vehicles on the road, a disturbance on the traffic flow.

We propose a model predictive controller that routes the vehicles efficiently to their destinations, making a trade-off between waiting in front of bridges and taking a detour. Model predictive control has been selected because it can handle these predicted disturbances that the bridges pose. Furthermore, it can be tuned to make a trade-off between a computationally fast and an accurate solution. A traffic split determines which part of the incoming traffic flow on a road interchange is sent towards which emanating road. These traffic splits represent the actuator variables of the controller. The total time spent by all vehicles in the network is the cost function to be minimised.

We describe a motorway network with METANET, a macroscopic traffic model. We do not use the full model, but a piecewise-affine approximation, in order to simplify the optimisation calculations significantly. We discuss two ways of modelling this approximation: an existing Mixed Logical Dynamical (MLD) formulation and a novel Linear Programming (LP) formulation. An analysis of the novel LP model in an N-step-ahead simulation is performed. We show that this problem cannot be written in a single LP problem, but that it is actually a linear multilevel programming problem (where N LP problems have to be solved consecutively).

As a means to model predicted disturbances, a novel store-and-forward bridge element is added to the nonlinear and the MLD model. A case study is performed to evaluate the effectiveness of this new bridge element. The results of this case study were satisfactory. Moreover, the results obtained with the MLD model provided an acceptable approximation of the results obtained with the nonlinear model.

Table of Contents

Abstract **i**

1 Introduction **1**

1-1 General Introduction 1

1-2 Problem Statement 2

1-3 Structure of the Thesis 4

2 Traffic Network Modelling **5**

2-1 Graph Modelling 5

2-1-1 Vertices and Links 5

2-1-2 Shortest Path Problems 6

2-1-3 Network Flow Problems 7

2-2 Traffic Modelling 8

2-2-1 Motorway Traffic Models 8

2-2-2 METANET 9

2-2-3 Piecewise-Affine and Linear Approximations of Nonlinear Functions in METANET 11

2-2-4 Piecewise-Affine Approximation of METANET Combined with Mixed Integer Linear Programming 13

2-3 Model Predictive Control 15

2-4 Summary 16

3 Improved METANET and Traffic Control **17**

3-1 Considerations about Improvements to the Mixed Logical Dynamical Model . . . 17

3-2 Piecewise-Affine Approximation of METANET Combined with Linear Programming 18

3-2-1 Function Evaluation Using Linear Programming Problems 18

3-2-2 Application to METANET 20

3-3 Accounting for Predicted Disturbances 25

3-3-1 Modified METANET 25
3-3-2 Describing Temporarily Unavailable Segments in METANET 26
3-4 Time-Dependent State Space Representation of the Mixed Logical Dynamical Model and the Linear Programming Model 32
3-5 Control Implementation 33
3-6 Summary 35

4 Simulation Results **37**
4-1 Setup 37
4-2 Analysis of a Single N-Step-Ahead Simulation with the Linear Programming Model 38
4-3 Case Studies on Bridge Elements in METANET 43
4-3-1 Zero-Length Bridges 44
4-3-2 Store-and-Forward Bridges 47
4-4 Summary 49

5 Conclusions and Recommendations **51**
5-1 Project Summary and Discussion 51
5-2 Contributions 53
5-3 Recommendations 53
5-3-1 Recommendations Concerning the Control Implementation 53
5-3-2 Recommendations Concerning Model Improvements 55
5-3-3 Recommendations Concerning Model Validation and Real-Time Implementation 56

A Approximations of Functions in METANET **57**
A-1 Piecewise-Affine Approximation of the Desired Speed Function 57
A-2 Piecewise-Affine Approximation of the Flow Function 58
A-3 Rewriting Equations in the Form of a Mixed Logical Dynamical Model 60
A-3-1 Desired Speed Function 60
A-3-2 Outflow at an Origin 61

B Network Flow Problems **63**
B-1 Minimum-Cost Flow Problem 63
B-2 Multi-Commodity Flow Problem 64
B-3 Linear Programming Problem 64

Glossary **73**
List of Acronyms 73
List of Symbols 74

Index **74**

"Comes a time when you're driftin'
Comes a time when you settle down."

— *Neil Young*

Chapter 1

Introduction

This chapter provides an introduction to the subject of this thesis: traffic routing under dynamic network topologies. It starts with a general introduction in Section 1-1, that emphasises the relevance of the subject. Thereafter, a specific problem statement is given in Section 1-2. Lastly, an overview of the chapter is given in Section 1-3.

1-1 General Introduction

Human driving can be unpredictable, and especially in high-intensity traffic it is likely to cause congestion. Over the past decades, the number of road users has only increased and congestion has therefore been a widely recognised and studied problem for a long time [9, 52, 69, 71]. Under high-intensity traffic, efficient road usage is a very relevant topic for a number of reasons. Reducing traffic congestion will not only lower the travel time of individual users, it can also result in drivers travelling at a more stable speed and therefore in more efficient fuel consumption which is an environmental benefit. Furthermore, congested traffic situations have a negative influence on safety.

Congestion might theoretically be permanently solved by increasing the infrastructure's capacity such that it can handle the worst-case traffic peaks, provided that the roads are filled with predictable and optimally platooning (smart) cars. However, this solution addresses the efficiency of the usage of a single road. We suggest that inefficient road usage across a network is a different problem that has to be tackled as well. Research is conducted to find effective solutions for minimising traffic congestion caused by inefficient road usage across a network [21,42,43,59,70]. In the context of a routing problem, we deem a network inefficiently used when one road is over-saturated, where other roads in the network (that could be used as an alternative route) still have a relatively low traffic density.

Congestion occurs in both urban and motorway traffic. In the literature, traffic control applied to both urban [22,50,56] and motorway models [11,20,60] can be found. In present-day motorway traffic, possible solutions for congestion include variable speed limits and on-ramp control [10, 40], whereas control applications that involve (future) smart cars also try to achieve a reduction of congestion with adaptive cruise control, platooning or by other ways of intercommunicating vehicles [17, 59]. Traffic congestion is also researched in other areas of science; think of road pricing [69], analysing the effects of road widening, and Braess's paradox [9].[1]

In the light of this thesis, inefficient road usage can mean two things. Either it means that on one single road the traffic does not adjust to other traffic properly, causing the flow to be suboptimal, or it means that too many cars take one (temporary blocked) road instead of other roads, causing inefficient usage of the available roads. The first case is a more fundamental problem because it will always exist on roads with too little capacity and unpredictable human drivers. The latter is the subject of this thesis.

The massive congestion on motorways that can occur during holidays or in the weekends is an example of such a problem. Note that especially weekend and holiday traffic is less bounded to the time at which it wants to arrive or the route it takes and in addition, the distances they travel are usually larger than for commuting traffic. Therefore, it is expected that the number of acceptable route alternatives is higher and that the traffic can be spread out over these routes in such a way that the congestion can be significantly reduced. Another example of this subject is the routing of traffic through a network with reduced or zero capacity on certain locations caused by a blocked road due to an accident, road maintenance or an open bridge.

A measure for efficient road usage in a traffic network context is provided by the Wardrop principles [71].[2] In this thesis, we will strive for a system optimal control implementation by minimising the total time spent by all vehicles in the network.

We consider the traffic network as a graph through which vehicles are to be routed. In this graph, each edge has an associated travel time. At specified moments in time we have either unavailable edges, or edges with an increased weight to reflect a temporary blocked road. Furthermore, we assume that we specify one origin and one destination vertex for each vehicle in the graph.

1-2 Problem Statement

A road blockage may have several causes. We focus our research on predicted blockages, i.e. the moment at which they take place is exactly known in advance. In this context, we consider the following two research questions:

> **How can we route traffic through a motorway network, where certain roads are blocked at known time periods, with as objective to minimise the total time spent by all vehicles in the network?**
> **How can this be done in the most computationally efficient way?**

[1]Braess's paradox reads: "Adding extra capacity to a network in which the moving entities chose their routes individually, can in some cases lead to a decrease of the performance of the network as a whole."

[2]Wardrop's first principle is known as the *selfish Wardrop equilibrium*. This is a user optimal equilibrium that is obtained when all road users try to minimise their own travel time. The second principle is an improvement by Wardrop, in which he proposes a new, system optimal equilibrium where road users cooperate. In the *social Wardrop equilibrium*, the average journey time is minimised.

The first research question involves a dynamic traffic routing problem. In order to solve this problem, a macroscopic traffic flow model of a motorway network has to be selected. This model should be able to accurately reflect the event of a road blockage, for example by the deletion or insertion of links. Moreover, when a link is unavailable, the traffic that still arrives at that link has to be queued in some way. In order to achieve a social equilibrium, a queue element should be implemented, in such a way that there is a penalty on letting too many vehicles wait in front of a road blockage. We will focus on solving this problem with predicted disturbances in the motorway network simulation model METANET in a Model Predictive Control (MPC) context. We use the Total Time Spent (TTS) by all vehicles in the network as a measure for efficient road usage. If environmental effects are also to be taken into account, METANET can be extended with the VT-macro model [75] so that the model includes fuel consumption. The controller will route the traffic by actuating the traffic splits at interchanges. The recurring question is: will it be preferable to avoid a blocked road, or to direct the traffic straight to it and wait until the road is unblocked. It is expected that in an optimal solution – when the roads are most efficiently used – a part of a traffic flow has to take a detour around the blocked road and the rest of the traffic has to wait for the road to be unblocked.

We assume that a static origin-destination matrix is provided.[3] Moreover, we assume that the predicted disturbances are caused by bridges that open and close at times that are known beforehand. Store-and-forward links should then allow traffic to queue in front of the bridges, in the case that they are open.

Considering the second research question, we suggest that an MPC approach is promising, as it can take into account the known future disturbances and moreover, it is easy to tune with only two parameters, N_p and N_c, offering a trade-off between computation speed and precision. The computational complexity of an MPC optimisation problem is heavily dependent on the underlying models used. In our case, this model is METANET, a nonlinear nonconvex model that results in high-complexity optimisation problems when used in combination with MPC. Note that METANET and MPC have been used before by others in dynamic routing problems. However, our problem differs from them mainly because we assume predictable unavailability of links. When a controller for motorway traffic routing is to be used in real time, the control optimisations should be calculated faster than a single simulation time step. METANET is a nonconvex and nonlinear model and is deemed too complex for such a cause. Therefore, a proposed simplification of the model by Groot et al. [34, 35] is expected to be a good starting point.

Since we use a mathematical model of a traffic network, we are able to use the traffic splits at interchanges as a control input. When actually implementing a real-time controller, the routes that vehicles take can only be advised, but not controlled. If we would want to achieve a real-time implementation, a Dynamic Route Information Panel (DRIP) could provide advisory information on a motorway traffic network [21, 42, 43].

[3]In an origin-destination matrix, we can find the traffic demand for every origin-destination pair, i.e. the number of vehicles that are leaving from origin o to destination d. Such a matrix is said to be static if it is independent of time.

1-3 Structure of the Thesis

This thesis is organised in the following manner. In Chapter 2, we discuss literature regarding graph modelling, shortest path problems, cost flow problems, and state-of-the-art motorway traffic models. Thereafter, in Chapter 3, we propose two contributions to the literature: an approximation of METANET that can be evaluated with Linear Programming (LP) and the addition of bridge elements to METANET. Chapter 4 provides an analysis of an N-step-ahead simulation with the newly presented LP model as well as the results of a number of case studies with the novel bridge element. Lastly, some conclusions and recommendations are presented in Chapter 5.

Chapter 2

Traffic Network Modelling

In this chapter we explain how to model traffic networks and more specifically we zoom in on METANET, a motorway traffic model. We start with a short general introduction to networks in Section 2-1 to provide some background knowledge. We elaborate on the traffic model METANET in Section 2-2. This model is simplified by means of a Piecewise-Affine (PWA) approximation and we show how a tractable conversion of the model can be embedded in a Model Predictive Control (MPC) framework. In Section 2-3 we further elaborate on this control method. The findings of this chapter are summarised in Section 2-4.

2-1 Graph Modelling

We discuss some graph theory in this section. This is by no means intended to be a complete overview, but rather to serve as a bridge towards the much more specific modelling of a traffic network. Next to the graph theory, we discuss some other operations research problems. This field is interesting because it gives a different view to a similar problem; (re-)routing traffic in case of a blocked link. The content of this section is largely based on [16, 23] covering both basic graph modelling and basic algorithms that can be used on graphs. For operations research literature related to routing problems under dynamic network topologies, the interested reader is kindly referred to [13,26,32,37,61].

2-1-1 Vertices and Links

A graph $G(M, V)$ is a data structure that can be used to model networks. Here, V is a finite set of *vertices* (in the literature also called *nodes*). In a graph, a vertex $\nu \in V$ can e.g. represent an intersection between roads on which vehicles can travel, a railway station, or a dead end. Moreover, $M \subset V \times V$ is a finite set of *edges*; they can represent a road or a railway between two intersections or stations. An edge $m = (\nu_1, \nu_2) \in M$ corresponds to a vertex pair. In an *undirected* graph we refer to such pairs as edges, whereas in case of a *directed* graph, we call them *links* (or *arcs*). A graph is said to be directed if its links (ν_1, ν_2) can only be crossed from ν_1 to ν_2, and not the other way around. This is contrary to an undirected graph where an edge (ν_1, ν_2) can also be crossed from ν_2 to ν_1. When an (undirected) edge exists between vertices ν_1 and ν_2, they are said to be *adjacent* to each other; $\text{adj}(\nu)$ is the set of all vertices adjacent to ν. In case of a directed graph, we define $\text{adj}_\text{i}(\nu)$ as the set of incoming links at vertex ν and $\text{adj}_\text{o}(\nu)$ as the set of emanating links at vertex ν.

In a weighted graph $G(M, V, w)$, we assign weight $w(\nu_1, \nu_2)$ to link (ν_1, ν_2). The weight can e.g. represent the length of the link or the time it takes to cross it. We consider a *static* network, when these weights are constant over time. If on the other hand, a *dynamic* model is desired, the values of w can vary with time t and are then written as $w(\nu_1, \nu_2, t)$. In the case of a *spatial* network we also assign a geographical location to every vertex.

In Figure 2-1, a static directed weighted graph $G(M, V, w, c)$ is considered. The links have a capacity c, which is the amount of traffic that can traverse the link in a certain amount of time, expressed for example in (veh/h).

In the context of this thesis, we consider a directed dynamic network topology. The link weights $w(x, y, t)$ can vary over time, or specific links can be unavailable at certain times. Therefore, depending on how we model this, the set M does not necessarily contain a fixed number of n links. This can be either:

n **not fixed** When a link is temporarily unavailable, it is also deleted in the model, giving the set M size $n - 1$.

n **fixed** The approach that is for example used in [15]. In the case of an unavailable link, it still exists in the model, but is given a relatively high value.

Weight updates and adding or removing links are closely related to each other [32]. It is therefore not expected that preferring one above the other will cause implementation issues.

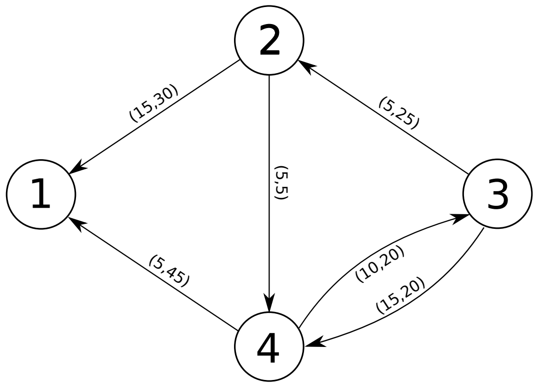


Figure 2-1: Simple network with four vertices and six directed links. The numbers between brackets are respectively the weight w and the capacity c.

2-1-2 Shortest Path Problems

A *path* is a sequence of adjacent vertices $p(\nu_1, \nu_2, ..., \nu_n)$ that form a route through a network. The (time dependent) length of a path is calculated as

$$L(p) = w(\nu_1, \nu_2, t_1) + w(\nu_2, \nu_3, t_2) + ... + w(\nu_{n-1}, \nu_n, t_{n-1}). \tag{2-1}$$

The *single-pair shortest path*, given as $\text{SP}(s, d)$, is the path between a source vertex s and a destination vertex d on which the parameter $L(p)$ is minimal. The shortest path from vertex ν_x to ν_y can be formulated as

$$\min_{p \in \mathcal{P}(\nu_x, \nu_y)} \quad L(p), \tag{2-2}$$

where $\mathcal{P}(\nu_x, \nu_y)$ is the set of paths with source vertex ν_x and destination vertex ν_y. Note that the shortest path from ν_x to ν_y can, but does not have to differ from the shortest path from ν_y to ν_x. Also note that it is possible that multiple shortest paths are found between two vertices.

It is a well known and widely studied problem to find the shortest path between two vertices, and moreover to find it as fast as possible [2,3,16,24,25,28,30,33,36]. The *shortest path distance* from vertex ν_x to ν_y and is defined as $\delta(\nu_x, \nu_y)$. It is the optimal value of $L(p)$ that follows from the minimisation in (2-2). The shortest path itself is defined as $\text{SP}(\nu_x, \nu_y) = \text{argmin}_{p \in \mathcal{P}(\nu_x, \nu_y)} L(p)$.

2-1-3 Network Flow Problems

We assume a graph that is defined by a *node-node adjacency matrix* G_{ij}, with i and j the matrix indices.[1] Furthermore, on link (i, j), we have the cost (or weight) W_{ij}, flow F_{ij}, and capacity C_{ij}. When applied to a traffic network, the link cost is the travel time in (h), the flow is the amount of traffic on a link in (veh/h), and the capacity is the maximum flow allowed on a link in (veh/h).

In the appendix we have defined two traffic routing problems: the *minimum-cost flow problem* in (B-1) and the *multi-commodity flow problem* in (B-2). When solved, they provide an optimal traffic flow distribution in a network with link weights and capacities.

The multi-commodity flow problem can be written in the form of a Linear Programming (LP) problem, which results in relatively low computation times (in Appendix B-3 we derive how to do this). We could run this problem in a discrete-time environment, so that we are able to route the traffic in a dynamic network. Then, in case of an unavailable link at a certain time step, we could update the graph matrix G_{ij} in between time steps and also update matrices W_{ij} and C_{ij} accordingly.

In the case we have 1 origin-destination pair and $a \ll C_{ij} \; \forall \; (i, j) \in M$, the solution to the multi-commodity flow problem would coincide with the shortest path that Dijkstra's algorithm [24] would find. We could say that this solution is the *selfish Wardrop equilibrium* (cf. footnote 2 in Chapter 1 on page 2). In the case of multiple origin-destination pairs and when the demand a exceeds the link capacity C_{ij} for one or more links, the minimisation results a *social Wardrop equilibrium*, without exceeding the network capacity.

[1] A node-node adjacency matrix that describes a network with n vertices can be created by starting with an $n \times n$ matrix filled with zeros and setting a 1 on every entry where a link exists. So, if $G_{\nu_x \nu_y} = 1$, a link exists from vertex ν_x to ν_y. Row i of G_{ij} represents the links emanating from vertex i. Similarly, column j represents the incoming links at vertex j.

The multi-commodity flow problem provides a partial answer to our research questions. However, it has some drawbacks too that we will address now.

The first problem is the accuracy of the model. Although many real-life traffic situations could be represented by a graph G_{ij} with weights and capacities, this traffic representation is a relatively low-detailed description. It does not include the traffic speed or density at a link. This has the implication that, regardless of the amount of traffic flow, the travel time on an edge is the same (that is, as long as the flow is not larger than the capacity). Furthermore, when the required flow would become higher than the capacity, the traffic is put in an origin queue until the network has the capacity again to dissolve the queue. An alternative would be to use a more realistic motorway model that is also capable of describing a jam under blocked road conditions in case of an unavailable link.

Secondly, the multi-commodity flow problem does not account for topology changes in advance. Notice that it is a *static* problem. That is, each instance is run on a specific situation at a set time and the optimisation does not take future disturbances into account. Suppose a bridge opens and closes again before the traffic arrives at that bridge. The multi-commodity flow problem (and other algorithms from the operations research literature [26, 32, 63–65]) would re-route the traffic twice, which is undesired. Solutions that account for *dynamic* network topologies are also found in the operations research literature [13, 37, 62], but they are computationally intensive because they consider the network at all time instants. Another solution can be provided by a motorway traffic model combined with MPC, which is a control approach capable of handling a topology change in advance. MPC provides a trade-off between speed and accuracy because the amount of time it predicts ahead can be tuned.

2-2 Traffic Modelling

We give a high-level overview of motorway traffic models in Section 2-2-1. The METANET model is selected and presented in Section 2-2-2, its nonlinearities are discussed in Section 2-2-3 and a PWA approximation that is known from the literature is given in Section 2-2-4, as well as the embedding of that PWA approximated model in an MPC optimisation that can be solved with Mixed Integer Linear Programming (MILP).

2-2-1 Motorway Traffic Models

Traffic models are usually divided in three types: microscopic models that are highly detailed and describe each vehicle separately, macroscopic models that have a more high-level approach and describe the average traffic behaviour in low detail, while the third category, mesoscopic models, uses characteristics from both microscopic and macroscopic models by describing the traffic in medium detail, mostly in a probabilistic manner. In [44] an extensive overview of these different types of models is provided. Our focus lies on fast state prediction, so that we can quickly route traffic with a priori knowledge of road blockades. We chose to use METANET, since this macroscopic model provides relatively good accuracy compared to its computation time. The traffic is represented by a flow, rather than by individual vehicles, giving a more high-level, general idea of the traffic distribution instead of a highly detailed description. Therefore, METANET is expected to be a good starting point.

2-2-2 METANET

METANET [53, 60] is used to model motorway networks.[2] In a graph representation of a such a network, vertices mark the changes in motorway situations, like on-ramps or a change in the number of lanes. Links are pieces of road with a constant number of lanes and speed limits (we use a speed limit of v_{free} on all links) connecting the vertices to each other. All links consist of a predetermined number of segments on which traffic progresses at each time step, so METANET is a deterministic, discrete-time, discrete-space model. It keeps track of the traffic flow $q_{m,i}$ in (veh/h), density $\rho_{m,i}$ in (veh/km/lane), and speed $v_{m,i}$ in (km/h) on each segment i of link m.

The flow at time step k is calculated by

$$q_{m,i}(k) = \lambda_m \rho_{m,i}(k) v_{m,i}(k), \tag{2-3}$$

where λ_m is the number of lanes.

The update equations for ρ and v are

$$\rho_{m,i}(k+1) = \rho_{m,i}(k) + \frac{T_{\text{s}}}{L_m \lambda_m}[q_{m,i-1}(k) - q_{m,i}(k)] \tag{2-4}$$

and

$$\begin{aligned} v_{m,i}(k+1) = v_{m,i}(k) &+ \frac{T_{\text{s}}}{\tau}[V[\rho_{m,i}(k)] - v_{m,i}(k)] \\ &+ \frac{T_{\text{s}} v_{m,i}(k)[v_{m,i-1}(k) - v_{m,i}(k)]}{L_m} \\ &- \frac{T_{\text{s}}\eta[\rho_{m,i+1}(k) - \rho_{m,i}(k)]}{\tau L_m(\rho_{m,i}(k) + \kappa)}, \end{aligned} \tag{2-5}$$

with

$$V[\rho_{m,i}(k)] = v_{\text{free},m} \exp\left[-\frac{1}{a_m}\left(\frac{\rho_{m,i}(k)}{\rho_{\text{cr},m}}\right)^{a_m}\right]. \tag{2-6}$$

In order of appearance, T_{s} (h) is the sample time, L_m (km) is the length of the segments of link m and τ (h), η (km^2/h), and κ (veh/km/lane) are model parameters.

In order to use METANET, the traffic network links are to be cut into road segments. The traffic advances from one segment into its adjacent segment at each time step, that is as long as the Courant-Friedrichs-Lewy (CFL) condition $L_m < T_s v_{\text{free}} \ \forall \ m \in M$ is preserved [18].

In (2-6), V is the desired speed, $v_{\text{free},m}$ is the free-flow speed, $\rho_{\text{cr},m}$ the critical density on link m, and a_m is a model parameter. Unless stated otherwise, we use the following values:

$$\begin{aligned} T_{\text{s}} &= 10 \text{ s} & \rho_{\text{jam}} &= 180 \text{ veh/km/lane} \\ L_m &= 0.5 \text{ km} & \rho_{\text{cr}} &= 33.5 \text{ veh/km/lane} \\ v_{\text{free}} &= 102 \text{ km/h} & a_m &= 1.867 \\ \tau &= 18 \text{ s} & \eta &= 60 \text{ km}^2\text{/h} \\ \kappa &= 40 \text{ veh/km/lane} & C_o &= 2000 \text{ veh/h}, \end{aligned}$$

[2]METANET stands for Modèle d'Ecoulement du Trafic Autoroutier: NETwork

where C_o is the capacity of origins (discussed below). This is a standard parameter set that is also used in [39, 54].

The desired speed function $V[\cdot]$ in (2-6) is plotted in Figure 2-2.

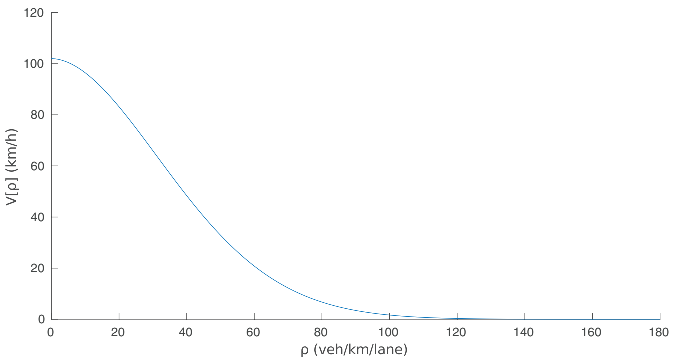


Figure 2-2: Fundamental diagram in which the desired speed V is plotted against the density $\rho_{m,i}$.

Origins In METANET, origins are modelled as a queue. Here, w_o represents the number of vehicles in the queue at origin o. Its update equation is given by

$$w_o(k+1) = w_o(k) + T_\mathrm{s}(d_o(k) - q_o(k)). \tag{2-7}$$

In this equation, d_o (veh/h) is the traffic demand and q_o (veh/h) is the outflow at the origin vertex o, given by

$$q_o(k) = \min\left[d_o(k) + \frac{w_o(k)}{T_\mathrm{s}},\ r_o(k)C_o,\ C_o\left(\frac{\rho_{\mathrm{jam},m} - \rho_{m,1}(k)}{\rho_{\mathrm{jam},m} - \rho_{\mathrm{cr},m}}\right)\right], \tag{2-8}$$

where C_o (veh/h) is the capacity of the origin, $\rho_{\mathrm{jam},m}$ (veh/km/lane) is the maximum density. Lastly, $r_o(k) \in [1, 0]$ is a ramp metering rate, which can be used on metered on-ramps.

Interchanges Considering a link m, we need a few extra equations describing the variables with subscript $i-1$ on a segment at the beginning of the link ($i = 1$) and similarly the variables with subscript $i+1$ on the last segment of the link ($i = i_{\mathrm{l},m}$). If two or more roads merge, the outflow is the sum of the inflows. It follows that the flows on the last segments of links entering a vertex ν can be summed to form the outflow

$$Q_\nu(k) = \sum_{\mu \in I_\nu} q_{\mu,i_{\mathrm{l},\mu}}(k), \tag{2-9}$$

which is distributed over the outgoing links. Here, I_ν is the set of incoming links and $i_{\mathrm{l},\mu}$ is the last segment of such an incoming link μ. When multiple links leave vertex ν, the flow on emanating link m is a fraction, $\beta_{\nu,m} \in [0, 1]$, of the incoming flow. The flow on link m emanating from ν, given by $q_{m,0}(k)$ is

$$q_{m,0}(k) = \beta_{\nu,m}(k) Q_\nu(k). \tag{2-10}$$

In (2-5), the *upstream velocity* $v_{m,i-1}$ and *downstream density* $\rho_{m,i+1}$ on the first and last segment of a link respectively are calculated as follows. In the update equation for the speed, ρ_{m,i_1+1} is written as

$$\rho_{m,i_1+1}(k) = \frac{\sum_{\mu \in O_\nu} \rho^2_{\mu,1}(k)}{\sum_{\mu \in O_\nu} \rho_{\mu,1}(k)}, \tag{2-11}$$

where O_ν denotes the set of all outgoing links at vertex ν.

For links in the set I_ν incoming at vertex ν, $v_{m,0}(k)$ is given by

$$v_{m,0}(k) = \frac{\sum_{\mu \in I_\nu} v_{\mu,i_{1,m}}(k) q_{\mu,i_{1,m}}(k)}{\sum_{\mu \in I_\nu} q_{\mu,i_{1,m}}(k)}. \tag{2-12}$$

The full nonlinear behaviour in METANET is not deemed necessary, because for traffic routing we assume that only a global idea of the network behaviour is needed, i.e. we accept an approximation error. Therefore, we propose a simplification of METANET, largely based on [34, 35]. These papers implement a PWA approximation and an MILP algorithm to use in an MPC environment. When making this approximation of the model, the calculation times become a lot smaller. The main ideas behind the work in these papers are discussed next.

2-2-3 Piecewise-Affine and Linear Approximations of Nonlinear Functions in METANET

In this section, we point out the nonlinearities that are present in METANET, as well as methods to linearise them or to make a PWA approximation of them. The results of the PWA approximations are graphically depicted in Figure 2-3. We sum up the adaptations made to the model below:

The Desired Speed Function We have the nonlinear function $V[\rho_{m,i}]$, a single-variate equation that can be efficiently approximated by a PWA function with a least-squares optimisation. The minimisation function is the squared error between the original function and its piecewise approximation. This is mathematically written as

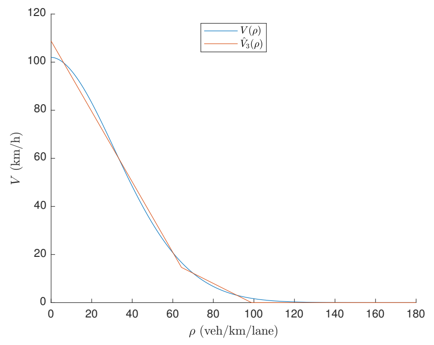
$$\min_{\alpha_1..\alpha_{n-1},\beta_1..\beta_n} \int_{x_{\min}}^{x_{\max}} (f_{\text{PWA},n}(x) - f(x))^2 dx, \tag{2-13}$$

in which the PWA function $f_{\text{PWA},n}(x)$ in n regions is defined on the interval $[x_{\min}, x_{\max}]$ by the pattern

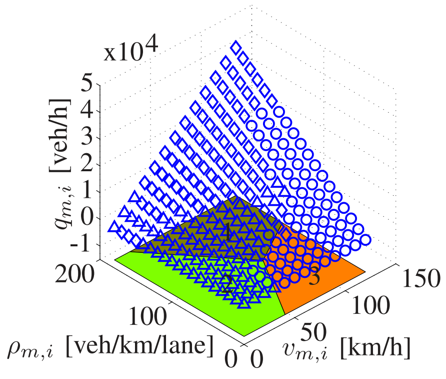
$$f_{\text{PWA},n}(x) = \begin{cases} \beta_0 + \frac{x - x_{\min}}{\alpha_1 - x_{\min}}(\beta_1 - \beta_0), & \text{for } x_{\min} \le x < \alpha_1 \\ \beta_1 + \frac{x - \alpha_1}{\alpha_2 - \alpha_1}(\beta_2 - \beta_1), & \text{for } \alpha_1 \le x < \alpha_2 \\ \vdots & \\ \beta_{n-1} + \frac{x - \alpha_{n-1}}{x_{\max} - \alpha_{n-1}}(\beta_n - \beta_{n-1}), & \text{for } \alpha_{n-1} \le x \le x_{\max}. \end{cases} \tag{2-14}$$

The n regions of which this continuous function consists are defined by parameters α and β. These parameters respectively define the x and y coordinates of the intersections of the affine functions, i.e. $f_{\text{PWA}}(\alpha_j) = \beta_j$ for $j = \{1, 2, ..., n-1\}$.

In the specific case of the desired speed function V, we add two constraints to the least-squares optimisation: $V_{\text{PWA}}(\alpha_{n-1}) = 0$ and $V_{\text{PWA}}(x_{\max}) = 0$, so that the last affine function, with domain $[\alpha_{n-1}, x_{\max}]$, has zero slope. When we do not impose these constraints, in an extreme case when $\rho > \rho_{\text{jam}}$, we could have $V_{\text{PWA}}(\rho) < 0$ and thus a negative speed and flow (traffic driving in reverse). Note that $\beta_{n-1} = 0$ and $\beta_n = 0$ are not added to the constraints, but rather excluded from the optimisation. To solve the unconstrained least-squares problem, we use a multi-start Levenberg-Marquardt algorithm. The result for a PWA function with three regions is plotted in Figure 2-3a. More details on the PWA approximation (in three but also in two regions) are given in Appendix A-1.



(a) PWA approximation of the fundamental diagram in 3 regions. This approximation is very similar to the one used in [35].



(b) PWA approximation in 3 regions of the flow function. This figure is taken from [34].

Figure 2-3: The two nonlinear functions in METANET that are approximated by PWA functions. On the left, we have the desired speed V from (2-6) and on the right the flow q from (2-3).

The Flow Function The second nonlinear function is $q = \rho v$ in (2-6). This bivariate equation is harder to capture in a PWA approximation. The accuracy of this approximation is dependent on e.g. the number of regions in the PWA functions and the method used for making the approximation. We describe two different methods in this thesis. The first is a three-dimensional PWA identification. In the second approach, the PWA identification is performed in two dimensions, after a *separation* of the bivariate equation.

The three-dimensional PWA identification used by Groot et al. [34, 35] consists of three methods that are all available in the Hybrid Identification Toolbox (HIT) [27], which is a platform embedded within the Multi-Parametric Toolbox (MPT) for Matlab [41]. An approximation is created with the aid of this toolbox using three PWA functions, the result of which is illustrated in Figure 2-3b.

The second method is the PWA approximation after a separation of the flow function. This method is proposed by [73]. A bivariate function $x_1 x_2$ can be separated by writing the function as

$$x_1x_2 = \frac{1}{4}\left[(x_1+x_2)^2 - (x_1-x_2)^2\right],$$

so that at this point, the two quadratic functions can be approximated by a two-dimensional PWA identification. This method will be more elaborately discussed in Section 3-2-2.

The Update Equation The update equation for $v_{m,i}$ in (2-5) contains two nonlinear parts. Firstly, a multiplication by the speed $v_{m,i}(k)$ in $v_{m,i}(k)[v_{m,i-1}(k) - v_{m,i}(k)]$ and secondly a division by the density $\rho_{m,i}(k)$ in $(\rho_{m,i+1}(k) - \rho_{m,i}(k))/(\rho_{m,i}(k)+\kappa)$. Both are linearised by keeping terms constant, i.e. the first term in the first equation, $v_{m,i}(k)$, and the denominator $\rho_{m,i}(k)$ in the second equation. This can be done by either keeping them constant according to historical data or, since the model is eventually controlled by MPC, by keeping them constant over the prediction horizon.

This is justified by the fact that the multiplication factor (T_s/L_m) in the first equation is relatively small and the multiplication factor $(T_\text{s}\eta/\tau L_m)$ as well as κ in the second equation are relatively large. These three factors all positively contribute to keeping the error caused by the substitution minimal.

The downstream density on the last segment of a link in (2-12) and the upstream velocity at the first segment of a link in (2-11) are linearised in a similar way by keeping the density $\rho_{\mu,1}$ and velocity v_{μ,i_1} constant.

A PWA system can be rewritten to a mixed logical dynamical model. This too has been done by Groot et al. [34, 35]. This conversion is described in the next section.

2-2-4 Piecewise-Affine Approximation of METANET Combined with Mixed Integer Linear Programming

Using the aforementioned methods to remove the nonlinearities from METANET, we arrive at a PWA system [68] of the form

$$\begin{aligned} x(k+1) &= A_i x(k) + B_i u(k) + f_i \\ y(k) &= C_i x(k) + D_i u(k) + g_i \end{aligned} \quad \text{for} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} \in \Omega_i \text{ and for } i \in \{1, ..., N\} \tag{2-15}$$

in which $\Omega_1, ..., \Omega_N$ are convex polyhedra. Furthermore, $x(k)$, $y(k)$ and $u(k)$ denote the state, output and input respectively.

Using this PWA model with MPC is possible, but becomes computationally intractable very fast. A last conversion is done, in order to make the MPC problem tractable for larger networks. The PWA model is rewritten in the form of a Mixed Logical Dynamical (MLD) model [4–7, 73]. In such a model, binary decision variables indicate in which region of the PWA functions the state variables are. The model description is given by

$$\begin{aligned} x(k+1) &= Ax(k) + B_1u(k) + B_2\delta(k) + B_3z(k) + f \\ y(k) &= Cx(k) + D_1u(k) + D_2\delta(k) + D_3z(k) + g \\ E_1x(k) + E_2u(k) &+ E_3\delta(k) + E_4z(k) \leq h, \end{aligned} \tag{2-16}$$

in which $\delta(k) \in \{0,1\}^{n_\mathrm{b}}$ is the binary decision variable and $z(k)$ an auxiliary variable that arises from the procedure explained next.

For the conversion of the PWA functions into a tractable MPC model, the next three statements (adapted from [6, 73]) are used:

$$f(x) \leq 0 \Leftrightarrow [\delta = 1] \text{ is equivalent to: } \begin{cases} f(x) \leq M(1-\delta) \\ f(x) \geq \epsilon + (m-\epsilon)\delta, \end{cases} \tag{2-17}$$

$$\delta = \delta_1\delta_2 \quad \Leftrightarrow \begin{cases} -\delta_1 + \delta \leq 0 \\ -\delta_2 + \delta \leq 0 \\ \delta_1 + \delta_2 - \delta \leq 1, \end{cases} \tag{2-18}$$

$$z = \delta f(x) \Leftrightarrow \begin{cases} z \leq M\delta \\ z \geq m\delta \\ z \leq f(x) - m(1-\delta) \\ z \geq f(x) - M(1-\delta), \end{cases} \tag{2-19}$$

in which $\delta_1, \delta_2 \in \{0,1\}$ are binary variables. We use m and M as a lower and upper bound respectively of the function $f(\cdot)$ and ϵ is a small tolerance variable. Now, if $f(\cdot)$ is an affine function defined on a bounded set X, then m and M are finite and the right-hand sides of the equivalences in (2-17)-(2-19) are (mixed integer) linear inequalities.

A single MLD-MPC iteration can now be written as an MILP problem in which the integer variables all belong to the set $\{0,1\}$. The approximations of the desired speed function V_PWA and the flow function q_PWA can now be rewritten in the from of an MLD model. An example of how to do this is provided in Appendix A-3-1. A few extra considerations to take into account when implementing the MLD model are mentioned next.

Suppose we have a PWA function with n regions and thus n functions $a_i\rho + b_i \;\; i \in \{1,2,...,n\}$. We observe that we do not need an explicit binary decision variable for the first region, $a_1\rho + b_1$. In other words, the system is always on the first region of the PWA function unless a decision variable prevents that. Therefore, if all binary variables are equal to 0, the system is on the first region $a_1\rho + b_1$. Therefore, a PWA function with n regions results in $n-1$ binary decision variables.

With m the number of decision variables, we could theoretically use 2^m combinations of binary values to describe $2^m + 1$ different settings of a PWA model. Put differently, we could say that in a PWA function with n regions, in principle a number of

$$m = \lceil \log_2(n-1) \rceil,$$

decision variables, where $\lceil \cdot \rceil$ is the ceiling function, would be sufficient to define all affine regions. Suppose we have a PWA function with 5 regions, then we could, using a different conversion to an MLD system, theoretically use only $\lceil \log_2(n-1) \rceil = 2$ binary variables (resulting in four unique configurations $[0,0], [1,0], [0,1]$, and $[1,1]$, and the first affine function). In our current model description, we use $n-1=4$ decision variables. When the number of binary variables increases, the resulting MILP problem is expected to be computationally more intensive. That is, in the specific case of using more regions and thus binary variables for the description of the desired speed function or the flow in Figure 2-3, we expect the calculations to be slower and the results more accurate. Therefore, using less binary variables is thought to speed up the calculation times. However, using a description with $\lceil \log_2(n-1) \rceil$ binary variables results in a more complex MLD system, and thus it is difficult to predict which of the two descriptions can be most efficiently solved (cf. [4]).

We add $\delta_i \geq \delta_{i+1} \;\; \forall \; i \in \{1, 2, ..., n-1\}$ as extra constraints to the ones we already mentioned. The decision variable δ_{i+1} can only be equal to 1 if all decision variables before it, δ_1 to δ_i, are equal to 1. We use these constraints, because a situation where $\delta_{i+1} = 1 \wedge \delta_i = 0$, does not describe an affine function that we want to use. We expect this to speed up the optimisation.

Apart from the functions for the desired speed and the traffic flow, the function for the outflow at origin q_o, given in (2-8), needs to be rewritten as well. The derivation of the conversion of this minimisation of three parameters to MLD form is provided in Appendix A-3-2.

2-3 Model Predictive Control

We use both the MLD and the yet to describe LP model as prediction models in an MPC context. This control method is expected to provide accurate predictions, which is desired because we want to efficiently route traffic through a network that is subject to predicted disturbances, meaning that we know beforehand when and where the disturbance is going to take place.

Model Predictive Control started to be used since Cutler and Ramaker [19] and Richalet et al. [66] pioneered it simultaneously. Nowadays it is a widely studied model-based control approach and it has been used many times in traffic control (e.g. [12,46,70]). In MPC a cost function J is minimized over a prediction horizon N_p. The prediction horizon is an integer number of time steps (T_s) over which the controller predicts the states by means of a given model. The minimisation of the cost function J is done at each time step, resulting in a set of control actions for the entire prediction horizon. Of this set, only the control input for the next time step is used. At the next time step, the optimisation problem is run again. Because the prediction horizon is shifted forward in each time step we also speak of *receding horizon control*. Sometimes, a control horizon N_c is added to the MPC problem. In that case, the control actions are determined over the first N_c time steps and kept constant over the next $N_\mathrm{p} - N_\mathrm{c}$ time steps. This procedure makes the calculations computationally less intensive and, in case of LP, it is sometimes used for robustness.

We expect that using MPC will yield favourable results because it is an optimisation problem that takes the dynamics of the system into account. The performance of MPC is for a large part determined by the cost function. We could implement a cost function that has a selfish goal, but MPC has the potential of performing better than this. If, contrary to the individual journey times, the average journey time of all users is minimised we might achieve system-optimality. We want to investigate if there will be less congestion when implementing such a social goal. This would come close to the social Wardrop equilibrium contrasting to the selfish Wardrop equilibrium (cf. footnote 2 in Chapter 1 on page 2).

The cost function used for the MPC problem is the Total Time Spent (TTS) over the prediction horizon, which is given by

$$J_{\mathrm{TTS}}^{\mathrm{MPC}}(k) = T_\mathrm{s} \sum_{j=1}^{N_\mathrm{p}} \left(\sum_{(m,i) \in I_\mathrm{all}} L_m \lambda_m \rho_{m,i}(k+j) + \sum_{o \in O_\mathrm{all}} w_o(k+j) \right) . \tag{2-20}$$

2-4 Summary

We have started with a few basics on graph modelling, after which we have discussed how to model a (time-dependent) graph and how to find a shortest path in one. This graph theory served as a starting point for explaining a traffic network model. We have showed how to model a traffic network by means of METANET. This is a macroscopic nonlinear and nonconvex model in which the traffic is represented with a flow q, a speed v, and a density ρ on each road segment. We have discussed a method from the literature that reduces the nonlinearities in METANET by creating a PWA approximation that can be captured in an MLD model description. This model is used in combination with MPC, a widely used predictive control strategy, and the resulting optimisation problem can be solved with MILP. The cost function that is used for the MPC problem is the TTS.

Chapter 3

Improved METANET and Traffic Control

In this chapter we describe two contributions to the literature. In Section 3-1, we briefly discuss where and how the current state of the art could be improved. Firstly we propose a possible alternative for the Mixed Logical Dynamical (MLD) model that was explained in the previous chapter. More specifically, in Section 3-2, we provide a method that makes the control optimisation problem suitable to be solved with Linear Programming (LP) instead of Mixed Integer Linear Programming (MILP), removing the computationally heavy binary decision variables that occur in the latter. This step to LP is possible under the condition that we use convex Piecewise-Affine (PWA) functions to approximate the nonlinear METANET variables and that we use the LP model as simulation model – when it is used as prediction model, we need to solve a linear multilevel programming problem. The second contribution, accounting for predicted disturbances in a traffic control context, is elaborated on in Section 3-3. We give a more in-depth model description and describe how the time-dependent network topology changes can be incorporated in the model; in Section 3-4, we define the time-dependent state space equations for both the LP and the MLD model. In Section 3-5, we discuss the different control methods in which the model will be embedded. The chapter will be summarised in Section 3-6.

3-1 Considerations about Improvements to the Mixed Logical Dynamical Model

From MILP to Multilevel LP In the previous chapter, we have discussed the main findings from [34, 35]. The approximation discussed in that chapter has the disadvantage that the MILP problem is expected to become harder to solve when increasing the accuracy by using more PWA regions. In a case study in [35], the error percentages are given for METANET with some levels of approximation w.r.t. to the original nonlinear model, when the Total Time Spent (TTS) was compared. The relative errors presented next result from using (a combination of) the methods for removing the nonlinearities in METANET, proposed in Section 2-2-3. It resulted in a 3.3% relative error in the TTS w.r.t. the original model when only using the approximation of keeping both $v(k)$ and $\rho(k)$ constant in the update equation (2-5). When only using the approximation of a PWA function in 3 regions of the desired speed function, this resulted in a 3.4% error w.r.t. the original model and when using 2 regions it resulted in an 8.8% error. Finally, when only approximating the flow function with a PWA function with 4 regions, this resulted in a relative error of 15%.

Concluding, we expect that, especially when using the PWA approximation in three regions for the desired speed function, this method can be mainly improved by adding more regions to the PWA approximation for the flow function q. However, the calculation times are expected to increase exponentially when doing so using MILP. We propose a new method, that provides the embedding in the eventual Model Predictive Control (MPC) optimisation to be a linear multilevel programming problem.

Predicted Disturbances Since we aim to use the models in a routing context under blocked road conditions, the insertion and deletion of segments is added to METANET in the form of a novel bridge element. The new model can still be used in combination with MPC, since the moments of opening and closing of the bridges are assumed to be known in advance. Furthermore, the traffic split $F_{\nu,m}$, defined later in this chapter, is used as the control input, such that the controller determines the routes of the traffic flows while minimising the TTS.

3-2 Piecewise-Affine Approximation of METANET Combined with Linear Programming

In this section, we propose a new method that might improve the calculation speed of the MLD model in an MPC context. We describe a linear minimization problem that can determine the value of a convex PWA function. This is described in Section 3-2-1. The LP method is incorporated into METANET, congruent to the original MILP approximation described in the previous chapter, so that a meaningful comparison can be made between the two. A detailed description of how this theory is implemented is given in Section 3-2-2.

3-2-1 Function Evaluation Using Linear Programming Problems

Two types of approximations of nonlinear equations in METANET are addressed here. The PWA approximations and the minimum of several parameters. The first covers the desired speed function V and the flow function q, the second covers the traffic outflow at an origin q_o.

A linear minimisation with convex piecewise-affine constraints

Given a *convex* PWA function $f_{\text{PWA},K}(x) : \mathbb{R}^n \to \mathbb{R}^m$ with K regions

$$f_i(x) = a_i^T x + b_i,$$

where $x \in N_i$ with N_i the active region and $i \in \{1, ..., K\}$. The function value can be evaluated at x in the following manner:

$$f_{\text{PWA}}(x) = \max_i (a_i^T x + b_i). \tag{3-1}$$

This method applies to convex PWA approximations only. The affine function that holds in its own region is always the one with the largest values for f_i. It is most easily seen that this is true if the affine functions are extended beyond the regions in which they hold. This is illustrated in Figure 3-1.

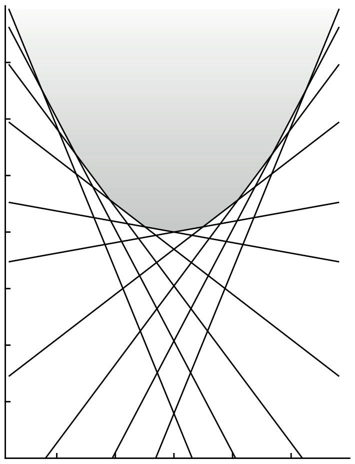


Figure 3-1: Example of a convex PWA function where the affine functions are extended beyond their bounds. The feasible region of the minimisation problem in (3-2) is indicated in grey.

Equation (3-1) is equivalent with

$$\begin{aligned} &\min \hat{f} \\ &\text{s.t. } \hat{f} \geq a_i^T x + b_i \;\forall\; i \in \{1, ..., K\}, \end{aligned} \tag{3-2}$$

which is an LP problem.

This concept is applied to two functions in METANET: the flow q in (2-3) and the desired speed V (2-5). The equations are worked out in Section 3-2-2.

The minimum of several parameters

The traffic outflow q_o at an origin in (2-8), is calculated by taking the minimum of several parameters. Consider a function f:

$$f = \min\{f_1, f_2, ..., f_n\}.$$

This function can be incorporated in an LP problem as well. We search the minimum of f, so f is smaller than or equal to each of the individual variables $f_1, ..., f_n$. These inequalities are the constraints to the following linear programming problem, where we find the maximum of f:

$$f = \min\{f_1, f_2, ..., f_n\} \Leftrightarrow \begin{aligned} \max\ & \hat{f} \\ \text{s.t.}\ & \hat{f} \leq f_1 \\ & \hat{f} \leq f_2 \\ & : \\ & \hat{f} \leq f_n. \end{aligned} \tag{3-3}$$

In Section 2-2-4, we provided an MLD method from the literature to find the minimum of several parameters, the notation given here provides an alternative to that minimisation. We will compare the performance of both methods in the next chapter.

3-2-2 Application to METANET

We now apply the methods from the previous section to the following functions in METANET: the outflow at origins q_o in (2-8), the desired speed V in (2-5) and the flow q in (2-3).

Outflow at Origins

In METANET, origins are modelled by (2-8), which is repeated here and generalised to the minimisation of three time dependent functions as:

$$\begin{aligned} q_o(k) &= \min\left[d_o(k) + \frac{w_o(k)}{T_\text{s}},\ r_o(k)C_o,\ C_o\left(\frac{\rho_{\text{jam},m} - \rho_{m,1}(k)}{\rho_{\text{jam},m} - \rho_{\text{cr},m}}\right)\right] \\ q_o(k) &= \min[f_1(k), f_2(k), f_3(k)]. \end{aligned} \tag{3-4}$$

To find the minimum of the three parameters $f_1(k)$, $f_2(k)$, and $f_3(k)$, we use the modification described in the previous section, so that we are able to incorporate it in an LP minimisation problem as

$$\begin{aligned} \min\ & -\hat{q}_o(k) \\ \hat{q}_o(k) &\leq f_1(k) \\ \hat{q}_o(k) &\leq f_2(k) \\ \hat{q}_o(k) &\leq f_3(k), \end{aligned} \tag{3-5}$$

where $\hat{q}_o(k)$ is the approximation of $q_o(k)$.

Desired Speed Function

This nonlinear and nonconvex function needs to be altered so that it is solvable with a linear programming problem. We approximate

$$V(\rho_{m,i}(k)) = v_{\text{free},m} \exp\left[-\frac{1}{a_m}\left(\frac{\rho_{m,i}(k)}{\rho_{\text{cr},m}}\right)^{a_m}\right]$$

by a PWA function with three regions as

$$\begin{aligned} &\min\ \hat{V}(\rho) \\ &\text{s.t.}\ -\hat{V}(\rho) \leq -a_i^T\rho - b_i \quad \forall\, i \in \{1,2,3\}, \end{aligned} \tag{3-6}$$

where $\hat{V}(\rho)$ is the LP evaluation in ρ using the PWA function $V_{\text{PWA}}(\rho)$. Here, $V_{\text{PWA}}(\rho)$ is the PWA approximation of the function $V(\rho)$.

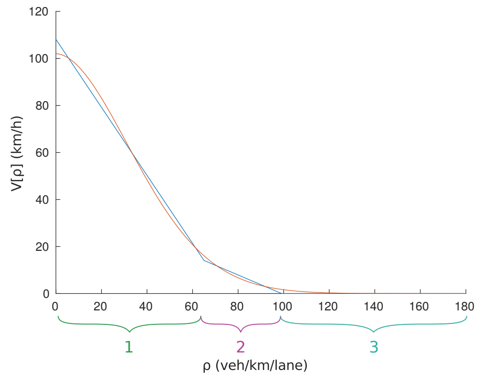


Figure 3-2: PWA approximation of the fundamental diagram in 3 regions

Figure 3-2 shows the PWA approximation of the desired speed function $V(\rho)$ in 3 regions. As we have seen in the previous chapter, this approximation with 3 regions results in an error of 3.4%. This is five times smaller than the error resulting from the approximation of the flow equation $q = \rho v$, as performed in Figure 2-3b. For this reason, we will concentrate on improving the approximation for q and keep the approximation for V, as described in this paragraph, restricted to these three segments.

Moreover, this approximation is convex without explicitly constraining it when using three (or less) regions. When using more regions, the least-squares optimization results in flattening out the affine functions close to $\rho = 0$. In that case, the constraint $a_1 < a_2 < ... < a_i$ would have to be added at the cost of accuracy.

For a given ρ, the linear minimisation in (3-6) gives the solution that lies on the piecewise approximation. The PWA functions have been determined with a least-squares optimisation as described in Section 2-2-3 and in more detail in Appendix A-1.

Flow Function

We also want to approximate the nonlinear and nonconvex function for the traffic flow with the use of PWA functions. We have the equation

$$q = \rho v,$$

which is plotted in Figure 3-3. This is the most important of the three approximations discussed in this section because it has the largest contribution to the error. In the following, we work out a separation of the flow function q, after which we perform the PWA approximation in two dimensions.

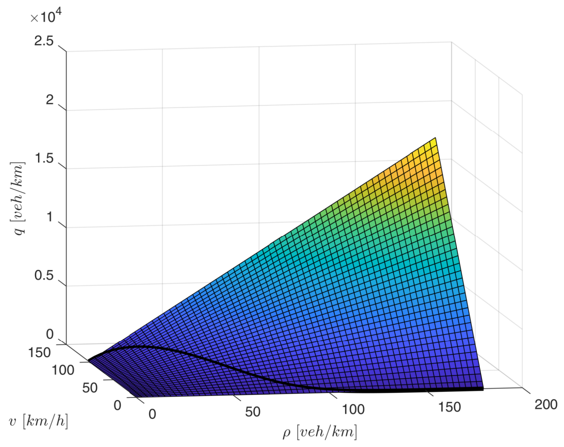


Figure 3-3: Function $q = \rho v$. The fundamental line, with equation $q_e = \rho V(\rho)$ is also shown.

Separation of the Flow Function We will now work out a method that previously has been mentioned, and is proposed by for example [73]. The nonconvex flow function will be separated into two convex functions, q^+ and q^-. We separate the function

$$q = \rho v$$

by writing

$$\rho v = \frac{1}{4}(\rho + v)^2 - \frac{1}{4}(\rho - v)^2. \tag{3-7}$$

With this separation we can write the equation as

$$q = \frac{1}{4}x_1^2 - \frac{1}{4}x_2^2,$$

with $x_1 = \rho + v$ and $x_2 = \rho - v$. At this point we define

$$q^+(x_1) = \frac{1}{4}x_1^2 \quad \text{and} \quad q^-(x_2) = \frac{1}{4}x_2^2,$$

and so we have

$$q = q^+(x_1) - q^-(x_2).$$

This separation has the important implication that we can approximate both the functions q^+ and q^- with a PWA function using two-dimensional identification. This results in an approximation of the form

$$q_{\text{PWA}} = q^{+}_{\text{PWA}}(x_1) - q^{-}_{\text{PWA}}(x_2). \tag{3-8}$$

Note that we could have used the same PWA function as an approximation for both q^+ and q^-. However, when we define the two functions separately, as we have done here, we reduce the number of regions in the PWA approximations and hence the number of decision variables in the MLD system as well as the number of inequality constraints in the LP system, cf. Figure 3-4.

Constraints on PWA Identification Problem Suppose we have a situation in which $\rho = \rho_{\text{jam}}$ and $v = 0$. The traffic is jammed and we expect $q = \rho_{\text{jam}} \cdot 0 = 0$. We fill in (3-8) with $x_1 = \rho_{\text{jam}} + 0$ and $x_2 = \rho_{\text{jam}} - 0$ and obtain $q_{\text{PWA}} = q^{+}_{\text{PWA}}(\rho_{\text{jam}}) - q^{-}_{\text{PWA}}(\rho_{\text{jam}})$. Since the functions q^{+}_{PWA} and q^{-}_{PWA} have a different domain on which they hold, when a least-squares optimisation would be run on them separately, the PWA approximations would (slightly) differ from each other. This is expected to become problematic when $x_1 = \pm x_2$, because in these situations the approximated flow q_{PWA} is expected to be equal to 0. However, when q^{-}_{PWA} is larger than q^{+}_{PWA}, which could occur when these approximations differ from each other, this would result in q_{PWA} being negative, i.e. the traffic flows backwards. To prevent this from happening we impose that $q^{+}_{\text{PWA}}(\rho_{\text{jam}}) = q^{-}_{\text{PWA}}(\rho_{\text{jam}})$.

The other extreme scenario is when $\rho = 0$ and $v = v_{\text{free}}$. Similarly to the above, the flow could potentially become negative when the PWA approximations do not meet the constraint $q^{+}_{\text{PWA}}(v_{\text{free}}) = q^{-}_{\text{PWA}}(-v_{\text{free}})$.

To implement these constraints, we adapt the PWA identification to perform only one least-squares optimisation on the negative half of the horizontal axis ($x_1, x_2 < 0$) after which we mirror the solution over the vertical axis, satisfying the constraint $q^{+}_{\text{PWA}}(v_{\text{free}}) = q^{-}_{\text{PWA}}(-v_{\text{free}})$, and we use the solution for both q^{+}_{PWA} and q^{-}_{PWA} simultaneously, which will satisfy the constraint $q^{+}_{\text{PWA}}(\rho_{\text{jam}}) = q^{-}_{\text{PWA}}(\rho_{\text{jam}})$. These measures might result in the approximation becoming less accurate than it potentially could be. It is preferred however over an approximation that potentially results in negative flows. We define

$$q^{\pm}_{\text{PWA}}(z) = \frac{1}{4}z^2$$

as the mirrored approximation of both q^{+}_{PWA} and q^{-}_{PWA}.

Bounds on PWA Identification Problem We note that an approximation on the entire domain $-(\rho_{\text{jam}} + v_{\text{free}}) \leq z \leq \rho_{\text{jam}} + v_{\text{free}}$ is not necessary because not all theoretically possible values for q are equally likely to be encountered (for example $q = \rho_{\text{jam}} v_{\text{free}}$ will not actually occur in METANET). To get an idea of what values for q are typical, we evaluate the expected flow with $v = V(\rho)$ as

$$q_{\text{e}} = \rho V(\rho). \tag{3-9}$$

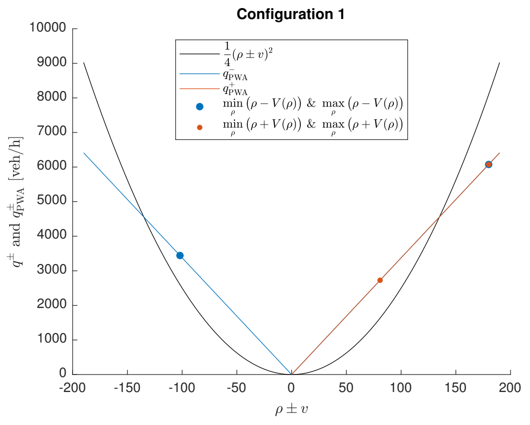
This line is plotted in Figure 3-3. These are not the only feasible values that q can have, i.e. the update equation for $v_{m,i}(k)$ in (2-5) is not equal to $V(\rho)$ but it will mostly be in the vicinity of this line, since the corrections on $v_{m,i-1}(k)$ and $\rho_{m,i+1}(k)$ are relatively small and the other terms steer the function towards $V(\rho)$ and thus towards this line.

For an indication of the bounds on the PWA identification problem, we calculate the expected values for q^+ and q^- in the same manner. We obtain the functions

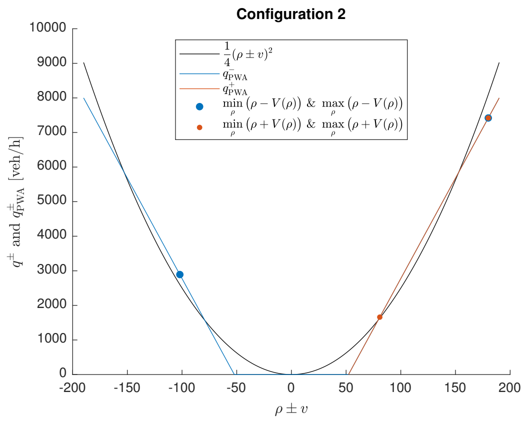
$$q_{\mathrm{e}}^+(\rho) = \frac{1}{4}(\rho + V(\rho))^2 \quad \text{and} \quad q_{\mathrm{e}}^-(\rho) = \frac{1}{4}(\rho - V(\rho))^2. \tag{3-10}$$

Here, the domains of q_{e}^+ and q_{e}^- are respectively given by

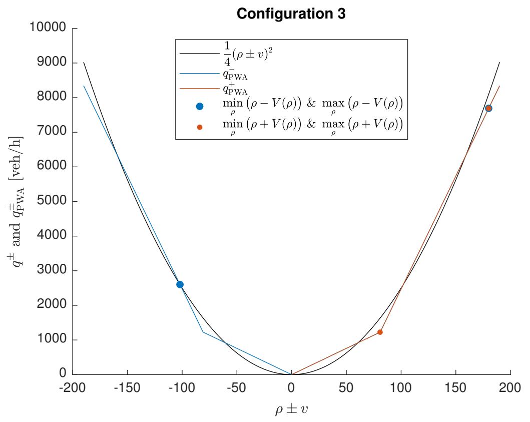
$$\min_\rho(\rho + V(\rho)) \le \rho \le \max_\rho(\rho + V(\rho)) \quad \text{and} \quad \min_\rho(\rho - V(\rho)) \le \rho \le \max_\rho(\rho - V(\rho)).$$



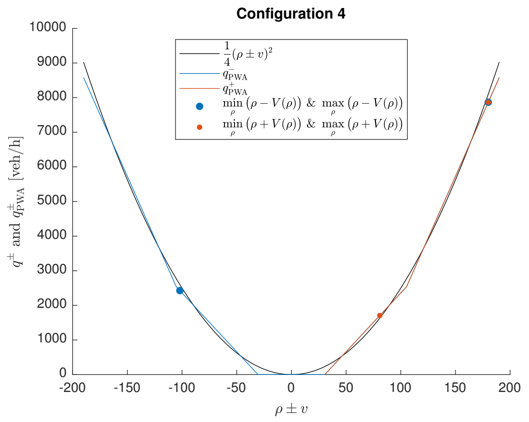
(a) PWA approximation in 2 regions. Here, q^+_{PWA} is always in the second region and q^-_{PWA} is in both regions, so we have 1 decision variable in this configuration.



(b) PWA approximation in 3 regions. Here, q^+_{PWA} is always in the third region and q^-_{PWA} is in all three, so we have 2 decision variables in this configuration.



(c) PWA approximation in 4 regions. Here, q^-_{PWA} is in all four regions. The minimum of q^+_{PWA} is only just in the fourth region, so we have chosen a safety region and divided q^+_{PWA} in 2 regions. We have 4 decision variables in this configuration.



(d) PWA approximation in 5 regions. Here, q^+_{PWA} is in the last two regions and the minimum of q^-_{PWA} is only just in the second region, so we have chosen a safety region and divided q^-_{PWA} in 5 regions. We have 5 decision variables in this configuration.

Figure 3-4: The four configurations of $q^\pm_{\mathrm{PWA}}$, with increasing numbers of decision variables, that we have chosen to run simulations on. We show the original function $\frac{1}{4}(\rho \pm v)^2$ and the approximations $q^\pm_{\mathrm{PWA}}$ of this function. The minimum and maximum of $(\rho \pm V)$ are also shown.

An estimate of the domain on which $q^{\pm}(z)$ holds, is then acquired by calculating

$$z_{\min} = \min_{\rho} \ (\rho + V(\rho),\ \rho - V(\rho)) \text{ and}$$
$$z_{\max} = \max_{\rho} \ (\rho + V(\rho),\ \rho - V(\rho)).$$

We note that $q^{\pm}$ is symmetrical over the vertical axis. Then,

$$\zeta = \max\left(|z_{\min}|, z_{\max}\right) = 180$$

provides an estimate for both the upper and lower bound, so $\{q^{\pm}(z) | -\zeta \leq z \leq \zeta\}$.

We use these bounds in the optimisation problem to determine $q^{\pm}_{\text{PWA}}$. We perform this unconstrained least-squares optimisation with a multi-start Levenberg-Marquardt approach. The required symmetry is embedded in the cost function. Therefore, we were still able to use the Levenberg-Marquardt algorithm on the minimisation problem. The exact functions that resulted from these optimisations are given in (A-3), (A-4), (A-5), and (A-6) in Appendix A.

Improving the Results Let us now analyse the effect of reducing the number of regions of the PWA functions q^{+}_{PWA} and q^{-}_{PWA} using the estimate of the bounds that we previously calculated in (3-10). Fewer PWA regions result in fewer decision variables, which can speed up the calculation times. The result of this simplification is captured in Figure 3-4, where four configurations of PWA approximations are shown, with 1, 2, 4, and 5 decision variables respectively. In configuration 3, we have used an extra PWA function for q^{+}_{PWA} as a safety region. The same has been done for q^{-}_{PWA} in configuration 4. A more elaborate discussion about the PWA identification of this flow function can be found in Appendix A-2. The configurations in the final form of Figure 3-4 have been used in the case studies that are discussed in Chapter 4.

3-3 Accounting for Predicted Disturbances

In this section, we propose a method that incorporates bridges in METANET. We interpret the opening or closing of these bridges as a predicted disturbance and we model these events as an alteration in the network topology. Therefore at least one of the system matrices in the MLD and LP (prediction) models has to be time dependent. We encounter time-dependent behaviour, both in the state space equations and in the linear (in)equalities that apply on the states and control actions. Furthermore, we describe the implementation of a controller that accounts for these predicted disturbances, caused by the opening and closing of bridges.

We start with a description of the modifications to METANET. More specifically, in Section 3-3-1 we zoom in on road interchanges and turning fractions. Here we define control input of the models. In Section 3-3-2 we describe several ways of modelling a bridge that opens and closes at predetermined moments in time.

3-3-1 Modified METANET

We analyse METANET again, to investigate which variables are suitable to be used as control input u.

We propose a modification of the equations that govern the traffic progression in and around a METANET vertex, defined earlier in (2-9) and (2-10). In METANET, β specifies the fractions by which the summed incoming flow at a vertex ν is split and divided over the emanating links. The flow towards link m was previously given as

$$q_{m,0}(k) = \beta_{\nu,m}(k) Q_\nu(k).$$

We propose to omit this variable $q_{m,0}(k)$, and instead use the traffic split $F_{\nu,m}(k)$. Here, $F_{\nu,m}(k)$ is the traffic flow (veh/h) at vertex ν towards link m at time step k. This traffic split is used as the only control input, so

$$u(k) = F_{\nu,m}(k). \tag{3-11}$$

Now, at the first segment of a link, we use the update equation

$$\rho_{m,1}(k+1) = \rho_{m,1}(k) + \frac{T_\mathrm{s}}{L_m \lambda_m}[F_{\nu,m}(k) - q_{m,1}(k)], \tag{3-12}$$

instead of the update equation in (2-4) used otherwise.

This implies that we do not use on-ramp metering or variable speed limits as control inputs. We have chosen to use the traffic split over the turning fractions, because the system would have to be quadratic in order to multiply $\beta_{\nu,m}(k)$ with $Q_\nu(k)$. This means that we work with actual flows $F_{\nu,m}(k)$, and not with the percentages $\beta_{\nu,m}(k)$. When the turning fractions are needed, which is the case when we use the nonlinear model as simulation model, they can be easily calculated back from (3-11) as

$$\beta_{\nu,m}(k) = F_{\nu,m}(k)/Q_\nu(k). \tag{3-13}$$

The equality conditions and bounds that hold for $F_{\nu,m}$ are given next. The sum of the turning fractions at each vertex should be equal to Q_ν and each $F_{\nu,m}$ in ν has a value between 0 and Q_ν, thus we have

$$\sum_{\mu \in I_\nu} F_{\nu,\mu}(k) = Q_\nu(k) \quad \forall\ \nu \in N \tag{3-14}$$

and

$$0 \leq F_{\nu,m}(k) \quad \forall\ \nu \in N, \forall\ m \in I_\nu. \tag{3-15}$$

3-3-2 Describing Temporarily Unavailable Segments in METANET

We speak of a topology change when a network structure alters. In this section we propose a means of handling topology changes in METANET. More specifically, we describe a couple of possible methods for implementing a bridge element in the model.

A bridge consists of two parts. Firstly, we have the queue area (on two sides of the bridge) which is typically but not necessarily a short area where vehicles wait in front of a boom barrier when the bridge is open. Secondly, we have the part of the bridge that actually opens to let water traffic pass. This is also typically a short segment compared to a METANET segment (which measures 500 metres). If this bridge is open, the traffic should be stopped and unable to flow to the downstream segment.

Bridges that can actually open are fairly uncommon on motorways, but they do exist. However, we will not run a case study on an real traffic situation, so the model does not yet have to exactly simulate an existing bridge.

METANET Segment

Both the queue area and the bridge itself are typically short elements compared to the 500 metres of a regular METANET segment. A straightforward method would be to let both the queue areas and the bridge itself be normal METANET segments. This implies that at least two segments are needed (with a combined length of at least 1000 metres), where an entire bridge might not even be as long as 500 metres. Creating a METANET segment that is shorter than the original 500 metres is possible, but this could result in the Courant-Friedrichs-Lewy (CFL) condition [18] being violated. The CFL condition is given as

$$T_\mathrm{s} \leq \min_{m \in I_\mathrm{link}} \frac{L_m}{v_{\mathrm{free},m}}, \tag{3-16}$$

where I_link is the set of all the links in a network. This condition ensures that traffic does not traverse more than one segment in a single time step. When we would implement a segment as short as the movable part of a bridge, the CFL condition is very likely to be violated.

Zero-Length Bridges

Another way to model a topology change is to place a bridge on the node in between two segments. This generalisation consists of two adjacent METANET segments somewhere in a link that can be disconnected from each other when the bridge opens. This bridge has zero length and only one segment upstream from the bridge node is used as queue area. The length of the bridge is included in the downstream segment, because the bridge itself is not a part of the queue area.

This is implemented in METANET as follows. We call the segment upstream from the bridge node i_u and the downstream segment i_d. When the bridge opens, the upstream traffic should be stopped and thus we set $q_{i_\mathrm{u}} = 0$. So we evaluate the flow $q_{i_\mathrm{u},b}$ at a segment upstream from a bridge b as

$$q_{i_\mathrm{u},b}(k) = \begin{cases} q_{i_\mathrm{u}} & \text{if } \delta_b = 0 \\ 0 & \text{if } \delta_b = 1, \end{cases} \tag{3-17}$$

where $\delta_b(k)$ is a binary value indicating whether a bridge b is open (1) or closed (0) at time step k. Furthermore, we have $q_{i_\mathrm{u}}(k) = \rho_{i_\mathrm{u}}(k) v_{i_\mathrm{u}}(k)$. When $\delta_b = 1$, the traffic is prevented from flowing out of i_u and so it fills up that segment, whilst maintaining the total amount of traffic.

The upstream velocity correction at segment i_d needs to be addressed, because when a bridge opens we have $v_{i_\text{d}-1} \neq v_{i_\text{u}}$. We set $v_{i_\text{d}-1} = v_{i_\text{d}}$, as if the bridge node was an origin node.

The downstream velocity correction at the upstream side of the bridge $\rho_{i_\text{u}+1}$ also needs to be addressed, because when a bridge opens we have $\rho_{i_\text{u}+1} \neq \rho_{i_\text{d}}$. We set $\rho_{i_\text{u}+1} = \rho_{i_\text{u}}$, as if the bridge node was a destination node.

Using this method can easily result in unsatisfactory METANET behaviour. To illustrate what happens we work out a potential case. Suppose a bridge has been open for a while and segment i_u is filled to its maximum capacity, so $\rho_{i_\text{u}} = \rho_\text{jam}$ and $v_{i_\text{u}} = 0$. Then we must have $q_{i_\text{u}-1} = 0$, otherwise segment i_u would still be filled with traffic and when that happens ρ_{i_u} crosses the maximum capacity and the model bounds are violated. However, segment $i_{\text{u}-1}$ might not yet be full so $\rho_{i_\text{u}-1} \neq \rho_\text{jam}$. Therefore, $V_{i_\text{u}-1}(\rho) \neq 0$ and this would cause $v_{i_\text{u}-1}$ to be larger than 0. This is a contradiction because we specifically required $q_{i_\text{u}-1} = 0$ in order to stop the traffic flow to segment i_u. We conclude that we use a single segment as queue area and that we have to make sure that the densities do not increase to values close to ρ_jam.

The main problem with this method is that METANET cannot handle an abrupt blockage very well. Even in a less extreme scenario ($\rho_{i_\text{u}} < \rho_\text{jam}$), the model is expected to cross the maximum capacity relatively easily. The update equations are based on traffic flows and thus setting one of these flows to 0 will very likely result in abnormal behaviour. Upstream traffic keeps filling the segment in front of the bridge, even when the maximum capacity ρ_jam of that segment is reached. To prevent this from happening we take two measures. Firstly, we put speed restrictions in the area upstream from the bridge, by using a minimum speed of $v_\text{min} = 4$ km/h on all segments, to regulate the inflow of traffic. Secondly, we simply limit the time the bridge is open.

Store-and-Forward Bridges

In explaining the previous method, we observed that the maximum capacity ρ_jam in the queue area can be exceeded relatively easily. To address this and other issues, we now propose a *vehicle store-and-forward* method that is more elaborate and therefore expected to be more robust the zero-length bridges method. Vehicles that are waiting in front of a bridge can be stacked in a queue similar to one used for an origin. Whenever the queue is empty and the bridge is closed, the queue segment is unused, but when the bridge opens, the queue segment starts storing traffic in a queue w_b with a maximum queue length of $M_{w,b}$ (and a minimum of $m_{w,b} = 0$).

Nonlinear Store-and-Forward Model The update equation for the bridge queue length w_b is given by

$$w_b(k+1) = w_b(k) + T_\text{s}\left(q_{u,b}(k) - q_{w,b}(k)\right), \tag{3-18}$$

where $q_{u,b}(k)$ is the inflow of traffic from segment i_u to the bridge queue and $q_{w,b}(k)$ is the traffic outflow from the bridge queue towards segment i_d. The outflow $q_{w,b}$ of the queue of bridge b is equal to 0 when the bridge is open, but also when the bridge is closed and the queue is empty. This variable is described in more detail further on. The traffic inflow to the bridge queue is given by

$$q_{u,b}(k) = \begin{cases} 0 & \text{if } \delta_b(k) = 0 \ \wedge \ w_b(k) = 0 \\ q_{\text{in},b}(k) & \text{otherwise.} \end{cases} \tag{3-19}$$

Furthermore, we define the actual value of the flow $q_{\text{in},b}$ from segment i_u to the bridge queue by

$$q_{\text{in},b}(k) = \min\left[q_{i_\text{u}}(k),\ \frac{M_{w,b} - w_b(k)}{T_\text{s}}\right], \tag{3-20}$$

with

$$q_{i_\text{u}}(k) = \rho_{i_\text{u}}(k) v_{i_\text{u}}(k). \tag{3-21}$$

Only when the queue segment is empty again after a bridge has closed, we have $q_{u,b} = 0$. This makes sense, because if the queue in front of the bridge has not dissolved yet, newly arriving traffic has to queue first before crossing the bridge. As with the zero-length bridges method, the bridge can be located in the middle of a link, in which case (3-20) holds. When however the bridge is located in between a vertex ν and the first segment i of link μ, we set $q_{\text{in},b}(k) = \min[F_{\nu,\mu}(k), (M_{w,b} - w_b(k))/T_s]$. For simplicity, in the following we will assume that the bridge is located in the middle of an edge.

The maximum queue length $M_{w,b}$ can be reached if bridge b is open for a long enough time period. When $M_{w,b}$ is reached, the queue cannot hold more traffic, and from that moment on, the traffic is stored in the upstream segment i_u in the same way as in the zero-length bridges method.

Suppose we are at time step k_m, the bridge is open, and the queue length will reach $M_{w,b}$ at the next time step. We have, from (3-20), the inflow at that time step

$$q_{\text{in},b}(k_m) = \frac{M_{w,b} - w_b(k_m)}{T_\text{s}}.$$

Because the bridge is still open we have $\delta_b(k_m) = 1$ and thus $q_{u,b}(k_m) = q_{\text{in},b}(k_m)$ and $q_{w,b}(k_m) = 0$ so

$$w_b(k_m + 1) = w_b(k_m) + T_\text{s} q_{\text{in},b}(k_m) = M_{w,b}.$$

It is important to make sure that the queue length at time step $k_m + 1$ is exactly $M_{w,b}$, otherwise we lose traffic, i.e. the density of segment i_u is not updated correctly.

When the bridge closes, the traffic flow starts up again, beginning with emptying the queue w_b into segment i_d, followed by the regular progression of the traffic in the network. Remember that, as long as $w_b \neq 0$, the queue is being filled at the same time. So the queue is emptied like a regular origin while (3-18) is still valid. The outflow $q_{w,b}$ is based on the outflow at an origin given in (2-8):

$$q_{w,b}(k) = \begin{cases} q_{\text{out},b}(k) & \text{if } \delta_b(k) = 0 \\ 0 & \text{if } \delta_b(k) = 1, \end{cases} \tag{3-22}$$

with

$$q_{\mathrm{out},b}(k) = \min\left[q_{u,b}(k) + \frac{w_b(k)}{T_\mathrm{s}},\ C_b\left(\frac{\rho_{\mathrm{jam},m} - \rho_{m,i_\mathrm{d}}(k)}{\rho_{\mathrm{jam},m} - \rho_{\mathrm{cr},m}}\right),\ C_b\right]. \tag{3-23}$$

Here, C_b (veh/h) is the capacity of the bridge. Unless otherwise specified, we will use

$$C_b = 2000 \text{ veh/h}.$$

The regular METANET equations are only used when both the bridge queue is empty and the bridge is closed. Similarly, the store-and-forward element is only used if either the bridge is open or the bridge queue is not empty. Therefore, the outflow $q_{i_\mathrm{u},b}$ at segment i_u and the inflow $q_{i_\mathrm{d}-1,b}$ at segment i_d are defined by

$$q_{i_\mathrm{u},b}(k) = \begin{cases} q_{i_\mathrm{u}}(k) & \text{if } \delta_b(k) = 0 \ \wedge \ w_b(k) = 0 \\ q_{\mathrm{in},b}(k) & \text{otherwise,} \end{cases} \tag{3-24}$$

and

$$q_{i_\mathrm{d}-1,b}(k) = \begin{cases} q_{i_\mathrm{u}}(k) & \text{if } \delta_b(k) = 0 \ \wedge \ w_b(k) = 0 \\ q_{w,b}(k) & \text{otherwise.} \end{cases} \tag{3-25}$$

At the time step at which we can completely empty a traffic queue, we make sure that this is done correctly. Suppose that at time step k_w we have

$$q_{\mathrm{out},b}(k_w) = q_{u,b}(k_w) + \frac{w_b(k_w)}{T_\mathrm{s}},$$

which implies that at this time step, the queue can be emptied in the adjacent segment. Then the last bit of traffic in the queue $w_b(k_w)$ is summed with the incoming traffic $q_{u,b}(k_w) = q_{i_\mathrm{u}}(k_w)$ at that time step and directly forwarded to the outgoing traffic via

$$q_{w,b}(k_w) = q_{\mathrm{out},b}(k_w) = q_{u,b}(k_w) + \frac{w_b(k_w)}{T_\mathrm{s}}.$$

This results in

$$w_b(k+1) = w_b(k) + T_\mathrm{s}\left(q_{u,b}(k) - q_{w,b}(k)\right) = 0,$$

so the bridge queue is empty and $q_{w,b} = q_{u,b} = 0$ from that point on and the queue segment stays empty until the bridge opens again. Moreover, there is no traffic lost or excess traffic left in the queue.

Furthermore, we note that the queue element is always emptied at the maximum rate. At time step k_w+1, the segments i_u and i_d are reconnected and the traffic the regular METANET model equations are used again for the state evolution.

Remark 3.1. *A zero-length bridge is essentially a special case of a store-and-forward bridge with* $M_{w,b} = 0$.

When a store-and-forward bridge queue is full, i.e. when at some time step $w_b(k) = M_{w,b}$*, the METANET segment upstream from this bridge has to hold the excess traffic. This is realised by forcing* $q_{i_\mathrm{u},b} = 0$*, so that the outflow at that upstream segment is blocked. Similarly, when using a zero-length bridge, the outflow is forced to* 0 *to block the outflow, c.f.* (3-17) *and* (3-24)*. Therefore, when we set* $M_{w,b} = 0$ *in the store-and-forward model, we observe the same behaviour as with the zero-length bridges.*

MLD Store-and-Forward Model The new equations introduced above can be added to the nonlinear model, but have to be rewritten for usage in the MLD model. This can be achieved using the methods described in [6, 73].

For the flow $q_{\mathrm{in},b}$ to the bridge we have

$$\hat{q}_{\mathrm{in},b}(k) = \delta_{q_{\mathrm{in},b}}(k)\hat{q}_{i_\mathrm{u}}(k) - [1 - \delta_{q_{\mathrm{in},b}}(k)]\frac{M_{w,b} - w_b(k)}{T_\mathrm{s}}, \tag{3-26}$$

where $\hat{q}_{i_\mathrm{u}}(k) = \hat{q}^+_{i_\mathrm{u}}(k) - \hat{q}^-_{i_\mathrm{u}}(k)$ and

$$\hat{q}_{i_\mathrm{u}}(k) \leq \frac{M_{w,b} - w_b(k)}{T_\mathrm{s}} \Leftrightarrow [\delta_{q_{\mathrm{in},b}}(k) = 1]. \tag{3-27}$$

For the conversion of $q_{u,b}$ to MLD form, we first define

$$\delta_{bw_b}(k) = \delta_b(k)\delta_{w_b}(k). \tag{3-28}$$

Now we have

$$\hat{q}_{u,b}(k) = [\delta_b(k) + \delta_{w_b}(k) - \delta_{bw_b}(k)]\, \hat{q}_{\mathrm{in},b}(k), \tag{3-29}$$

which is in MLD form if we add the three constraints given in (2-18) on $\delta_{bw_b}(k)$. Furthermore we have

$$\delta_{w_b}(k) = \begin{cases} 1 & \text{if } w_b(k) > 0 \\ 0 & \text{if } w_b(k) = 0, \end{cases}$$

which translates to

$$\begin{aligned} w_b(k) &\leq M_{w,b}\delta_{w_b} \\ w_b(k) &\geq \epsilon\delta_{w_b}, \end{aligned} \tag{3-30}$$

where ϵ is the machine precision.

The MLD form of $q_{w,b}(k)$ is derived next. We start with rewriting (3-23) as

$$q_{\mathrm{out},b}(k) = \min\left[q_{u,b}(k) + \frac{w_b(k)}{T_\mathrm{s}},\ C_b\left(\frac{\rho_{\mathrm{jam},m} - \rho_{m,i_\mathrm{d}}(k)}{\rho_{\mathrm{jam},m} - \rho_{\mathrm{cr},m}}\right),\ C_b\right]$$

$$q_{\mathrm{out},b}(k) = \min\left[q^{\mathrm{out}}_{b,1}(k),\ q^{\mathrm{out}}_{b,2}(k),\ C_b\right], \tag{3-31}$$

which is similar to the equation for the outflow at an origin; cf. Appendix A-3-2. In a similar manner, we can rewrite (3-31) in MLD form as

$$\hat{q}_{\mathrm{out},b}(k) = q_{b,1}^{\mathrm{out}}(k)\delta_{b,1}^{\mathrm{out}}(k) + q_{b,2}^{\mathrm{out}}(k)\delta_{b,2}^{\mathrm{out}}(k) + C_b\delta_{b,3}^{\mathrm{out}}(k), \tag{3-32}$$

where

$$\begin{aligned} [\delta_{b,1}^{\mathrm{out}}(k) = 1] \Leftrightarrow \quad & [q_{b,1}^{\mathrm{out}}(k) \leq q_{b,2}^{\mathrm{out}}(k) \text{ and } q_{b,1}^{\mathrm{out}}(k) \leq C_b] \\ [\delta_{b,2}^{\mathrm{out}}(k) = 1] \Leftrightarrow \quad & [q_{b,2}^{\mathrm{out}}(k) \leq q_{b,1}^{\mathrm{out}}(k) \text{ and } q_{b,2}^{\mathrm{out}}(k) \leq C_b] \\ [\delta_{b,3}^{\mathrm{out}}(k) = 1] \Leftrightarrow \quad & [C_b \leq q_{b,1}^{\mathrm{out}}(k) \text{ and } C_b \leq q_{b,2}^{\mathrm{out}}(k)]. \end{aligned} \tag{3-33}$$

Now we can write the MLD equivalent of $q_{w,b}$ as

$$\hat{q}_{w,b}(k) = [1 - \delta_b(k)]\hat{q}_{\mathrm{out},b}(k). \tag{3-34}$$

The last two variables we rewrite to MLD form are $q_{i_\mathrm{u},b}(k)$ and $q_{i_\mathrm{d}-1,b}(k)$. We have

$$\hat{q}_{b,i_\mathrm{u}}(k) = \hat{q}_{i_\mathrm{u}}(k) + \big(\delta_b(k) + \delta_{w_b}(k) - \delta_{bw_b}(k)\big)\big(\hat{q}_{\mathrm{in},b}(k) - \hat{q}_{i_\mathrm{u}}(k)\big), \tag{3-35}$$

and

$$\hat{q}_{b,i_\mathrm{d}-1}(k) = \hat{q}_{i_\mathrm{u}}(k) + \big(\delta_b(k) + \delta_{w_b}(k) - \delta_{bw_b}(k)\big)\big(\hat{q}_{w,b}(k) - \hat{q}_{i_\mathrm{u}}(k)\big). \tag{3-36}$$

Again, we removed the multiplications of the binary decision variables $\delta_b(k)\delta_{w_b}(k)$ in (3-35) and (3-36) by replacing them with the already introduced binary variable δ_{bw_b} and the three constraints in (2-18).

Topology Changes

Because we approached the opening of a bridge in METANET as a topology change in the underlying network, we can generalise these methods to be applicable to a variety of traffic situations, i.e. the road can be obstructed for more reasons than an open bridge, for example an accident that blocks the motorway, red traffic lights, or road maintenance. The methods for describing temporarily unavailable connections in METANET could be used in these cases as well.

3-4 Time-Dependent State Space Representation of the Mixed Logical Dynamical Model and the Linear Programming Model

At this point, as all the different parts of the models have been discussed separately, we put them together to arrive at a generalised form of a state space system that fits both the MLD and LP model as

$$\begin{aligned} x(k+1) &= A(k)x(k) + B(k)u(k) + \tilde{B}(k)\tilde{v}(k) + f(k) \\ y(k) &= C(k)x(k) + D(k)u(k) + \tilde{D}(k)\tilde{v}(k) + g(k) \\ E_1(k)x(k) &+ E_2(k)u(k) + \tilde{E}(k)\tilde{v}(k) \leq h(k), \end{aligned} \tag{3-37}$$

where $x(k)$ are the states, $u(k)$ the control inputs, the optimisation variables $\tilde{v}(k)$, and $y(k)$ the system's outputs.

The MLD and LP model have a time-dependent nature because they can account for predicted disturbances. A change in the topology of the network (i.e. when a bridge opens or closes) results in an update of the system matrices.

The state x is given as $x(k) = [w_o(k), w_b(k), \rho(k), v(k)]^T$ in the MLD model, where we use the store-and-forward method for the bridges. For the LP model it is defined by $x(k) = [w_o(k), \rho(k), v(k)]^T$, because we use the zero-length bridges method here.

The control input is given by the controlled traffic split $u(k) = F_{\nu,m}(k)$. This is the only real control input, i.e. only by controlling $F_{\nu,m}(k)$ we can steer the system.

The optimisation variables for the MLD model are given as a stacked vector with all the auxiliary variables, $\tilde{v}(k) = [\delta^T(k), z^T(k)]^T$. In the LP model, we use a vector containing all the dummy variables $\tilde{v}(k) = [\hat{q}_o(k), \hat{V}(k), \hat{q}^+(k), \hat{q}^-(k)]$.

The output $y(k)$ is a linear combination of the input and the state. It is a (not always directly measurable) system parameter that can say something about the performance of a controller or the system in general.

In this thesis, we use the TTS to evaluate the overall performance of the different models and controllers. This variable was already given in (2-20) and is a little more elaborately discussed in the next section. In the scope of this thesis, the system does not necessarily require any more outputs. However, since the outputs are a linear function of the states and inputs, we can use them to get insight in the inputs. This is especially useful for summing the (sometimes elaborate) functions that $\hat{q}(k)$ consists of, for $\hat{q_o}(k)$, and $\hat{V}(k)$. Thus we use $y(k) = [J^y_{\text{TTS}}(k), \hat{V}(k), \hat{q_o}(k), \hat{q}(k)]^T$, where $J^y_{\text{TTS}}(k)$ is the TTS at an instant of time. In the MLD model, where we use store-and-forward bridges, we add the most important variables of this element to the output; we have $y(k) = [J^y_{\text{TTS}}(k), \hat{V}(k), \hat{q_o}(k), \hat{q}(k), \hat{q}_{u,b}(k), \hat{q}_{w,b}(k)]^T$.

3-5 Control Implementation

In this section, we discuss the control implementation that we use in combination with the traffic models. The only available actuators are the traffic splits. Many control methods are suitable for obtaining an input vector $u(k) = F_{\nu,m}(k)$ at each time step. We propose to apply MPC, which is used in combination with both the MLD and the LP model described throughout this thesis.

Control Framework

When implementing a controller, we close the control loop and link the outputs back to the inputs. In the context of this thesis, the aim of the controllers is to minimise the TTS.

In the case we use the store-and-forward method for bridges, we add the bridge queue w_b to the function for the TTS, which is then given by

$$J_{\text{TTS}} = T_{\text{s}} \sum_{k=1}^{N_{\text{sim}}} \left(\sum_{(m,i)\in I_{\text{all}}} L_m \lambda_m \rho_{m,i}(k) + \sum_{o\in O_{\text{all}}} w_o(k) + \sum_{b\in B_{\text{all}}} w_b(k) \right), \tag{3-38}$$

where N_{sim} is the number of time steps in the simulation, I_{all} are all possible (m,i) pairs, O_{all} is the set of all origins, and B_{all} the set of all bridges in the network.

A major motivation for computationally light controllers is real-time implementation. When using a controller in real time, we require the control inputs before moving to the next time step. That means that the computation time T_{c} always has to be less than the time step, i.e. $T_{\text{c}} < T_{\text{s}}$.

Model Predictive Control Applied to the Modified Models

We use MPC in combination with the generalised time-dependent system described in (3-37) for both the MLD and LP model. This is possible under the condition that the disturbances on the system are known beforehand. We can for example anticipate on the opening of a bridge easily because ships travel at a relatively constant speed. We can use the models in combination with MPC because, although the system matrices are time dependent, we update them at each time step in which a bridge opens. Sudden topology changes are not accounted for by this controller, so we leave disturbance rejection outside the scope of this thesis.

The cost function used for the MPC problem is TTS over the prediction horizon, is given by

$$J_{\text{TTS}}^{\text{MPC}}(k) = T_{\text{s}} \sum_{j=1}^{N_{\text{p}}} \left(\sum_{(m,i)\in I_{\text{all}}} L_m \lambda_m \rho_{m,i}(k+j) + \sum_{o\in O_{\text{all}}} w_o(k+j) + \sum_{b\in B_{\text{all}}} w_b(k+j) \right). \tag{3-39}$$

Remark 3.2. *An implementation issue arises when using a control horizon N_{c}. In the time steps from N_{c} to N_{p}, the input $u(k) = F_{\nu,m}(k)$ is kept constant, and the auxiliary variables in $\tilde{v}(k)$ are still updated at each time step. Therefore, we encounter an inaccuracy. Since the values in $u(k)$ are traffic flows and not the turning fractions, they need to be constantly updated in order to obtain an accurate prediction.*

If we would choose $N_{\text{c}} < N_{\text{p}}$, we would have the constant traffic splits $F_{\nu,m}(k)$, next to the variable flows $\hat{q}(k)$ in the constraints on the inputs given in (3-14) *and* (3-15)*. This would cause the prediction values to be inaccurate. The model is infeasible in that case, unless we disable the constraints on the inputs. Therefore, the usage of a control horizon N_c is up to this point undesired and we will use $N_{\text{c}} = N_{\text{p}}$ in the following.*

An N-step-ahead simulation with the LP model cannot be accurately solved using a single LP optimisation, but only with a sequence of LP problems. We provide a theoretical motivation for this statement in Section 4-2. We propose to use time-dependent weighing factors as a potential way of solving the N-step-ahead simulation with a single LP problem. In the LP-MPC model, we use

$$
\begin{aligned}
J_{\mathrm{LP}}^{\mathrm{MPC}}(k) = \frac{w_{\mathrm{TTS}}}{J_{\mathrm{nom,TTS}}} J_{\mathrm{TTS}}^{\mathrm{MPC}}(k) + \sum_{j=0}^{N_{\mathrm{p}}-1} \Bigg(& \frac{w_{\hat{V}}(k)}{J_{\mathrm{nom},\hat{V}}} \hat{V}(k+j) + \frac{w_{\hat{q}^+}(k)}{J_{\mathrm{nom},\hat{q}^+}} \hat{q}^+(k+j) \\
& + \frac{w_{\hat{q}^-}(k)}{J_{\mathrm{nom},\hat{q}^-}} \hat{q}^-(k+j) - \frac{w_{\hat{q}_o}(k)}{J_{\mathrm{nom},\hat{q}_o}} \hat{q}_o(k+j) \Bigg),
\end{aligned} \tag{3-40}
$$

as the objective function, where w_{TTS}, $w_{\hat{V}}(k)$, $w_{\hat{q}^\pm}(k)$ and $w_{\hat{q}_o}(k)$ are weighting factors. In Section 5-3, we briefly discuss a potential way of solving the N-step-ahead simulation in a single LP problem with the use of these time-dependent weighing factors.

3-6 Summary

In this chapter, we have made two contributions to the literature. We have expanded the already existing method that approximates METANET with an MLD model that can be evaluated with MILP. This resulted in a model that can be evaluated using LP, which is expected to improve the trade-off between speed and accuracy significantly. Furthermore, using MPC, we are able to account for predicted topology changes. MPC can be used in combination with both the MLD and LP model. The computation times of MPC problems, where either the MLD or LP model is used as prediction model, need to be compared to each other.

The second contribution to the literature is an extension to METANET that allows for topology changes in the network. More specifically we have proposed a method that incorporates store-and-forward bridges in the nonlinear and in the MLD model.

Chapter 4

Simulation Results

In this chapter, we run a number of simulations using the newly proposed Linear Programming (LP) model, and using the newly added bridge elements in the Mixed Logical Dynamical (MLD) model. In Section 4-1, we describe the general setup of the simulations: the software we have used and the tuning of the Model Predictive Control (MPC) problems. Thereafter, in Section 4-2, we describe an N-step-ahead simulation with the LP model in detail and we show that, to solve this simulation correctly, it should be written as a linear multilevel programming problem. Since we expected the resulting problem to be a single LP problem, we do not run further simulations with the LP model. Instead, to provide a proof of concept of the newly proposed bridge elements, we run a number of simulations on both the nonlinear and MLD model in Section 4-3.

4-1 Setup

In this section, we describe the setup used for the simulations performed in this chapter.

Software For the simulations that have been run in this chapter we have used Gurobi 8.1.0 to evaluate the LP and Mixed Integer Linear Programming (MILP) problems. We have used the MATLAB Application Programming Interface (API) for MATLAB R2019b on both Ubuntu and Windows.

Tuning the Model Predictive Controller In MPC, we can tune the performance using the parameters N_p and N_c. In Remark 3.2 we have already discussed that we need to have $N_\mathrm{p} = N_\mathrm{c}$ in order to satisfy the equality constraints on $F_{\nu,m}$. Furthermore, we choose their value according to a rule of thumb that is closely related to the one proposed by Hegyi [38,49], according to whom N_p should be larger, but not much larger than the typical travel time in the network. That rule of thumb is based on a network with a single stretch of road with variable speed limits. However, we consider a motorway network with multiple routes of different lengths and therefore we will use a different rule of thumb.

We choose

$$N_\mathrm{p} \geq |\mathcal{I}_\mathrm{l}| T_\mathrm{s},$$

where $\mathcal{I}_\mathrm{l}$ is the set of segments that comprise the longest route in the network and $|\mathcal{I}_\mathrm{l}|$ is the number of elements in that set. If there is some traffic in a segment, some of it progresses to the downstream segment at each time step, unless a bridge is open. Therefore, we can say that in $|\mathcal{I}_\mathrm{l}|$ time steps, the prediction horizon spans the entire network size (if we do not consider bridges), which is the minimum requirement for the MPC to work properly. This is most easily understood when filling an empty network with two routes of unequal length. When the prediction horizon is shorter than the time it takes to fill the shortest route with traffic, the Total Time Spent (TTS) has the same value no matter the values of $F_{\nu,m}$, whether we route all traffic over the longest or the shortest route, because for either choice the amount of traffic in the network is exactly the same. From the moment the traffic disappears at a destination, a difference can be found.

If we compare the LP model with the MLD model, when used as prediction model in an MPC context, we expect them to give the same predictions when the same Piecewise-Affine (PWA) configurations are used and the prediction and control horizon have the same length. When used for routing on a large scale network, we allow for some deviation from the MLD model, as long as the prediction with the LP model is reasonably accurate. However, when used in a simple single-lane network where no routing is required, we expect the predictions to be exactly the same. We will show that this is not the case when solving an N-step-ahead problem in a single LP problem, and that it should be written as a linear multilevel programming problem instead.

4-2 Analysis of a Single N-Step-Ahead Simulation with the Linear Programming Model

The LP model does not prove to work as expected when we use it as prediction model. We have noticed that the MPC implementation gives incorrect results, even for very small networks where no routing is required and with $w_\mathrm{TTS} = 0$, in which case we expected the outcomes to be exactly the same as the results obtained with the MLD model used as prediction model. To get a better understanding of what the underlying reason for these incorrect results is, we provide an analysis of the equations that arise when we use the LP model in an N-step-ahead simulation.

Setup This analysis is performed on a very small network with one single-lane link, one segment, one origin, and one destination. We solve the N-step-ahead simulation with $N = 2$. We define the cost function for this problem as

$$J_\mathrm{LP} = \sum_{k=0}^{N-1} \left(w_{\hat{V}} \hat{V}(k) + w_{\hat{q}^+} \hat{q}^+(k) + w_{\hat{q}^-} \hat{q}^-(k) - w_{\hat{q}_o} \hat{q}_o(k) \right), \tag{4-1}$$

with $w_{\hat{V}} = 1$, $w_{\hat{q}^+} = 1$, $w_{\hat{q}^-} = 0.1$, and $w_{\hat{q}_o} = 1$. For clarity, we repeat the update equations for the LP model. Since we only have one link, we leave out the link index m and write

$$w_o(k+1) = w_o(k) + T_\mathrm{s}(d_o(k) - q_o(k)),$$

$$\rho_i(k+1) = \rho_i(k) + \frac{T_\mathrm{s}}{L\lambda}[q_{i-1}(k) - q_i(k)],$$

and

$$\begin{aligned} v_i(k+1) = v_i(k) &+ \frac{T_\mathrm{s}}{\tau}[V[\rho_i(k)] - v_i(k)] \\ &+ \frac{T_\mathrm{s} v_i(k)[v_{i-1}(k) - v_i(k)]}{L} \\ &- \frac{T_\mathrm{s}\eta[\rho_{i+1}(k) - \rho_i(k)]}{\tau L(\rho_i(k) + \kappa)}, \end{aligned}$$

Since we only use one lane and one segment, we have $\lambda = 1$ and $i = 1$. Furthermore, we have the following boundary conditions for this problem:

$$\hat{q}_0(k) = \hat{q}_o(k), \quad \rho_2(k) = \rho_1(k), \quad \text{and} \quad v_0(k) = v_1(k).$$

We substitute these equations in the update equations and obtain

$$\begin{aligned} w_o(k+1) &= w_o(k) + T_\mathrm{s}(d_o(k) - \hat{q}_o(k)) \\ \rho_1(k+1) &= \rho_1(k) + \frac{T_\mathrm{s}}{L}[\hat{q}_o(k) - \hat{q}_1(k)] \\ v_1(k+1) &= v_1(k) + \frac{T_\mathrm{s}}{\tau}[\hat{V}_1(k) - v_1(k)]. \end{aligned}$$

The number of regions in the PWA approximations of V and q are chosen as small as possible. We use a PWA approximation V_PWA in two regions, q^+_PWA in one region, and q^-_PWA in two regions; so we have:

$$\begin{aligned} V_\mathrm{PWA}(k) &= \begin{cases} a_V \rho_1(k) + b_V & \text{if } \rho_1(k) \leq s_V \\ 0 & \text{if } \rho_1(k) > s_V \end{cases} \\ q^+_\mathrm{PWA}(k) &= \quad a_q(\rho_1(k) + v_1(k)) \\ q^-_\mathrm{PWA}(k) &= \begin{cases} -a_q(\rho_1(k) - v_1(k)) & \text{if } \rho_1(k) - v_1(k) \leq 0 \\ \ \ a_q(\rho_1(k) - v_1(k)) & \text{if } \rho_1(k) - v_1(k) > 0. \end{cases} \end{aligned}$$

Here, a_V and a_q indicate the slope and b_V indicates the y-intercept of the corresponding affine function. Furthermore, s_V is the x-coordinate of the intersection of the two affine functions in V_PWA.

Note that the approximations for q^+ and q^- are based on the same parameter a_q and are mirrored over the vertical axis, as has been explained in Section 3-2-2 and visualised in Figure 3-4a.

The constraints to which the dummy variables in the LP model are subject at a given time step k are:

$$
\begin{aligned}
\hat{V}_1(k) &\geq a_V \rho_1(k) + b_v \\
\hat{V}_1(k) &\geq 0 \\
\hat{q}_o(k) &\leq d_o(k) + w_o(k)/T_s \\
\hat{q}_o(k) &\leq C_o \\
\hat{q}_o(k) &\leq C_o \left(\frac{\rho_{\text{jam}} - \rho_1(k)}{\rho_{\text{jam}} - \rho_{\text{cr}}} \right) \\
\hat{q}_1^+(k) &\geq a_q(\rho_1(k) + v_1(k)) \\
\hat{q}_1^-(k) &\geq \; - a_q(\rho_1(k) - v_1(k)) \\
\hat{q}_1^-(k) &\geq a_q(\rho_1(k) - v_1(k))
\end{aligned}
\tag{4-2}
$$

Next, we state the values of the variables in the update equations (ρ, v, and w_o) at time step $k = 1$ as a function of the initial condition. For given $\rho(0)$, $v(0)$, and $w_o(0)$, their values at the next time step are:

$$
\begin{aligned}
w_o(1) &= w_o(0) + T_{\text{s}}(d_o(0) - \hat{q}_o(0)), \\
\rho_1(1) &= \rho_1(0) + \frac{T_{\text{s}}}{L}[\hat{q}_o(0) - (\hat{q}_1^+(0) - \hat{q}_1^-(0))] \\
v_1(1) &= v_1(0) + \frac{T_{\text{s}}}{\tau}[\hat{V}_1(0) - v_1(0)].
\end{aligned}
$$

We aim to write the constraints in the MPC problem as a function of the initial condition, i.e. the values of the state variables at $k = 0$. This way, we get a good insight in the optimisation problem and we will be able to see where it contradicts.

We write the constraints on the dummy variables for the two time steps in the prediction horizon by evaluating (4-2) for $k = 0$ and $k = 1$ and obtain the following inequality constraints:

$$
\begin{aligned}
\hat{V}_1(0) &\geq a_V \rho_1(0) + b_v & \hat{V}_1(1) &\geq a_V \rho_1(1) + b_v \\
\hat{V}_1(0) &\geq 0 & \hat{V}_1(1) &\geq 0 \\
\hat{q}_o(0) &\leq d_o(0) + w_o(0)/T_s & \hat{q}_o(1) &\leq d_o(1) + w_o(1)/T_s \\
\hat{q}_o(0) &\leq C_o & \hat{q}_o(1) &\leq C_o \\
\hat{q}_o(0) &\leq C_o \left(\frac{\rho_{\text{jam}} - \rho_1(0)}{\rho_{\text{jam}} - \rho_{\text{cr}}} \right) & \hat{q}_o(1) &\leq C_o \left(\frac{\rho_{\text{jam}} - \rho_1(1)}{\rho_{\text{jam}} - \rho_{\text{cr}}} \right) \\
\hat{q}_1^+(0) &\geq a_q(\rho_1(0) + v_1(0)) & \hat{q}_1^+(1) &\geq a_q(\rho_1(1) + v_1(1)) \\
\hat{q}_1^-(0) &\geq \; - a_q(\rho_1(0) - v_1(0)) & \hat{q}_1^-(1) &\geq \; - a_q(\rho_1(1) - v_1(1)) \\
\hat{q}_1^-(0) &\geq a_q(\rho_1(0) - v_1(0)) & \hat{q}_1^-(1) &\geq a_q(\rho_1(1) - v_1(1)).
\end{aligned}
\tag{4-3}
$$

Furthermore, we can write the constraints in the second time step as a function of the variables in the previous time step. We substitute the equations that we have for $\rho(1)$, $v(1)$, and $w_o(1)$ in the second column of inequalities in (4-2). The set of constraints that results from this substitution can be further simplified and is stated next (we leave working out these equations to the reader). We have:

$$\begin{aligned}
\hat{V}_1(1) \geq & a_V \left(\rho_1(0) + \frac{T_\mathrm{s}}{L}[\hat{q}_o(0) - \hat{q}_1^+(0) + \hat{q}_1^-(0)]\right) + b_V \\
\hat{V}_1(1) \geq & 0 \\
\hat{q}_o(1) \leq & \frac{w_o(0)}{T_\mathrm{s}} + d_o(0) + d_o(1) - \hat{q}_o(0) \\
\hat{q}_o(1) \leq & C_o \\
\hat{q}_o(1) \leq & C_o \left(\frac{\rho_\mathrm{jam} - \left(\rho_1(0) + \frac{T_\mathrm{s}}{L}[\hat{q}_o(0) - \hat{q}_1^+(0) + \hat{q}_1^-(0)]\right)}{\rho_\mathrm{jam} - \rho_\mathrm{cr}} \right) \\
\hat{q}_1^+(1) \geq & a_q \left(\rho_1(0) + \frac{T_\mathrm{s}}{L}[\hat{q}_o(0) - \hat{q}_1^+(0) + \hat{q}_1^-(0)] + v_1(0) + \frac{T_\mathrm{s}}{\tau}[\hat{V}_1(0) - v_1(0)]\right) \\
\hat{q}_1^-(1) \geq & - a_q \left(\rho_1(0) + \frac{T_\mathrm{s}}{L}[\hat{q}_o(0) - \hat{q}_1^+(0) + \hat{q}_1^-(0)] - v_1(0) + \frac{T_\mathrm{s}}{\tau}[\hat{V}_1(0) - v_1(0)]\right) \\
\hat{q}_1^-(1) \geq & a_q \left(\rho_1(0) + \frac{T_\mathrm{s}}{L}[\hat{q}_o(0) - \hat{q}_1^+(0) + \hat{q}_1^-(0)] - v_1(0) + \frac{T_\mathrm{s}}{\tau}[\hat{V}_1(0) - v_1(0)]\right).
\end{aligned} \tag{4-4}$$

Evaluation We have deliberately chosen not to use the nominal values J_nom, defined in (3-40), in this evaluation. Since we have $w_\mathrm{TTS} = 0$, we expect that the results we obtain with a single control optimisation are independent of the weights on the dummy variables. Regardless of what values we pick for these weights, we expect that the dummy variables are pushed on the PWA function lines and evaluate the model progression correctly (which means that they have exactly the same values as the MLD equivalent of this problem).

Next, we evaluate the N-step-ahead simulation with initial condition

$$\begin{aligned}
w_o(0) &= 0 \\
\rho_1(0) &= 0 \\
v_1(0) &= 106.78.
\end{aligned}$$

Here, $v_1(0) = V_\mathrm{PWA}(0)$. When we specify the initial condition, we can compare the result that we expect from the MLD solution with the optimal solution that we obtain with the LP model as prediction model in a *single* LP problem.

We ran the simulation with the LP model in a single LP problem and compared the result with the values based on the MLD solution. In Table 4-1 we state all the values of the state and dummy variables for both the expected and the unexpected solution. For the cost functions we found the following values:

$$\begin{aligned}
J_\mathrm{LP,e} &= 6303 \\
J_\mathrm{LP,u} &= 6142,
\end{aligned}$$

where $J_\mathrm{LP,e}$ denotes the MLD-based solution and $J_\mathrm{LP,u}$ is the cost of the solution to the single LP problem.

	MLD-based solution		*single* LP solution	
k	0	1	0	1
$w_o(k)$	0	0	0	2.77
$\rho_1(k)$	0	5.56	0	0
$v_1(k)$	106.78	106.78	106.78	106.78
$\hat{V}_1(k)$	106.78	106.78	106.78	106.78
$\hat{q}_o(k)$	1000	1000	0	2000
$\hat{q}^+(k)$	3603.9	3791.4	3603.9	3603.9
$\hat{q}^-(k)$	3603.9	3416.4	3603.9	3603.9

Table 4-1: This table shows the values of the state and dummy variables of an N-step-ahead simulation with $N = 2$. The values for which the LP model deviates from the MLD-based solution are indicated in orange. The cost for the MLD-based solution is $J_{\mathrm{LP,e}} = 6303$ and the cost for the LP solution is $J_{\mathrm{LP,u}} = 6142$.

Analysis In order to understand why the results from this N-step-ahead optimisation with the single LP problem differ from the MLD-based solution, we need to analyse (4-4).

By the substitution we made in (4-4), we exposed the dependency of the dummy variables at one time step on the values of these variables on the previous time step(s). This dependency is the underlying reason why the solution to the 1-step-ahead LP problem is correct, and the solutions to N-step-ahead simulations with $N > 1$ are mostly incorrect. In order to find the correct solution to an N-step-ahead simulation, we should solve N LP problems consecutively, instead of a single LP problem, as we do now.

Multilevel Programming The N-step-ahead simulation should be written in the form of a (linear) multilevel programming problem. This type of problem is the generalisation of the more commonly researched bilevel programming problem. A survey on this type of problem is given by Colson et al. [14]. In such a hierarchical framework, the optimisation problem of the upper level U with upper part optimisation variables y, contains another optimisation problem. This nested optimisation problem of the lower level L with optimisation variables z is to be solved first, after which its solution can be used to solve the upper level problem. We write a bilevel optimisation problem as

$$\begin{aligned} \min_{y,z} \quad & V_{\mathrm{U}}(y,z) \\ \text{s.t.} \quad & G_{\mathrm{U,I}}(y,z) \leq 0 \\ & G_{\mathrm{U,E}}(y,z) = 0 \\ & z = \operatorname*{argmin}_{z} V_{\mathrm{L}}(y,z) \\ & \quad \text{s.t.} \quad G_{\mathrm{L,I}}(y,z) \leq 0 \\ & \quad \phantom{\text{s.t.}} \quad G_{\mathrm{L,E}}(y,z) = 0, \end{aligned} \tag{4-5}$$

where I and E respectively indicate the inequality and equality constraints.

The problem as formulated in (4-5) is known to be NP-hard, even in linear form [47]. Furthermore, to the best of our knowledge, we cannot specify the best solution approach for the linear bilevel optimisation problem. This is due to the fact that, even though many algorithms exist, there is not a single algorithm that consistently performs the best for all linear bilevel programming problems [72].

The bilevel problem formulation of (4-5) will now be extended to a linear multilevel programming problem for our specific case where we combine the LP model in (3-37) with MPC. Each time step on the receding horizon of the MPC problem corresponds to a level in the hierarchical problem. We write the linear multilevel programming problem as

$$\begin{aligned}
\min_{\tilde{x}_N} \quad & c_N^T[\tilde{x}_1^T, \tilde{x}_2^T, ..., \tilde{x}_N^T]^T \\
\text{s.t.} \quad & \tilde{x}_{N-1} = \operatorname*{argmin}_{\tilde{x}_{N-1}} c_{N-1}^T[\tilde{x}_1^T, \tilde{x}_2^T, ..., \tilde{x}_N^T]^T \\
& \qquad \vdots \\
& \qquad \text{s.t.} \quad \tilde{x}_1 = \operatorname*{argmin}_{\tilde{x}_1} c_1^T[\tilde{x}_1^T, \tilde{x}_2^T, ..., \tilde{x}_N^T]^T \\
& \qquad \qquad G_{\text{I}}[\tilde{x}_1^T, \tilde{x}_2^T, ..., \tilde{x}_N^T]^T \leq b_{\text{I}} \\
& \qquad \qquad G_{\text{E}}[\tilde{x}_1^T, \tilde{x}_2^T, ..., \tilde{x}_N^T]^T = b_{\text{E}},
\end{aligned} \tag{4-6}$$

where $\tilde{x}_k$ is a stacked optimisation variable of the state x, control inputs u, and dummy variables $\tilde{v}$ at time step k. Moreover, c_k^T is the cost vector at time step k, corresponding to the objective function as specified in (3-40). The (in)equality constraints can be decoupled from the separate optimisation problems at each level, that is, the constraints are specified at all levels of the multilevel problem at the same time. We can do this, because the constraints are linear and can be determined beforehand; we know how the next time step is dependent on the previous time step and furthermore, the PWA constraints on the dummy variables all have the same form, independent of the time step.

We will come back to this multilevel programming problem formulation in Section 5-3, where we provide some recommendations on how to solve it.

4-3 Case Studies on Bridge Elements in METANET

In this section, we evaluate how well the newly proposed bridge elements simulate the queueing of cars in front of an open bridge in METANET in a case study using both the nonlinear and the MLD model. We do not use the LP model in this case study because we have not specified how to model the store-and-forward bridges in this framework. Furthermore, we have not specified a way to solve the linear multilevel programming problem that arises when we use the LP model as prediction model.

Minimum Speed Function Recall from Remark 3.1 that a zero-length bridge is a special case of a store-and-forward bridge with $M_{w,b} = 0$. In other words, from the moment the maximum queue length is reached in the store-and-forward model, the bridge element will behave as a zero-length bridge. From preliminary tests, we learned that the speeds in METANET can become negative relatively easy in such a scenario, which is undesirable. Fortunately, this is a commonly encountered phenomenon, and as a countermeasure we add a maximum function with a lower speed limit. We have substituted an extra variable $\tilde{v}_{m,i}(k)$ in the update equation for the speed of a METANET segment

$$\begin{aligned}\tilde{v}_{m,i}(k) = v_{m,i}(k) &+ \frac{T_\mathrm{s}}{\tau}[V[\rho_{m,i}(k)] - v_{m,i}(k)] \\ &+ \frac{T_\mathrm{s} v_{m,i}(k)[v_{m,i-1}(k) - v_{m,i}(k)]}{L_m} \\ &- \frac{T_\mathrm{s}\eta[\rho_{m,i+1}(k) - \rho_{m,i}(k)]}{\tau L_m(\rho_{m,i}(k) + \kappa)},\end{aligned} \tag{4-7}$$

so that now we have the new update equation

$$v_{m,i}(k+1) = \max(\tilde{v}_{m,i}(k), v_\mathrm{min}). \tag{4-8}$$

Here v_min is the lower speed limit, for which we use

$$v_\mathrm{min} = 4 \text{ km/h}.$$

Moreover, we discuss how to ensure a minimum speed in the MLD model. We use the variable $\tilde{v}_{m,i}(k)$ as before in (4-7). However, the maximum function in the update equation needs to be rewritten. We have

$$v_{m,i}(k+1) = \delta_{\tilde{v}_{m,i}}(k)\tilde{v}_{m,i}(k) + (1 - \delta_{\tilde{v}_{m,i}}(k))v_\mathrm{min}, \tag{4-9}$$

where $\delta_{\tilde{v}_{m,i}}$ is a binary decision variable defined as follows:

$$[\delta_{\tilde{v}_{m,i}} = 1] \Leftrightarrow v_\mathrm{min} - \tilde{v}_{m,i}(k) \leq 0. \tag{4-10}$$

Working out these equations in the form of an MLD model is similar to the conversions given in Appendix A-3 and is left to the reader.

Setup

We will perform a number of case studies on a single-lane network stretch of 10 km (equalling 20 segments) with a single origin on segment 1 and a single destination on segment 20. A bridge is located between upstream segment $i_\mathrm{u} = 8$ and downstream segment $i_\mathrm{d} = 9$. In the case study, we will open a bridge for 5 minutes. The initial condition is a steady-state traffic flow resulting from a constant demand of $d_o = 1000$ veh/h. We will analyse simulations with the nonlinear model and with the MLD model.

4-3-1 Zero-Length Bridges

Nonlinear Model

We commence with a proof of concept where we open a zero-length bridge for 5 minutes (equalling 30 time steps). With a constant traffic demand of 1000 veh/h, after a certain amount of time, a steady state is reached. We start the simulation from this steady state

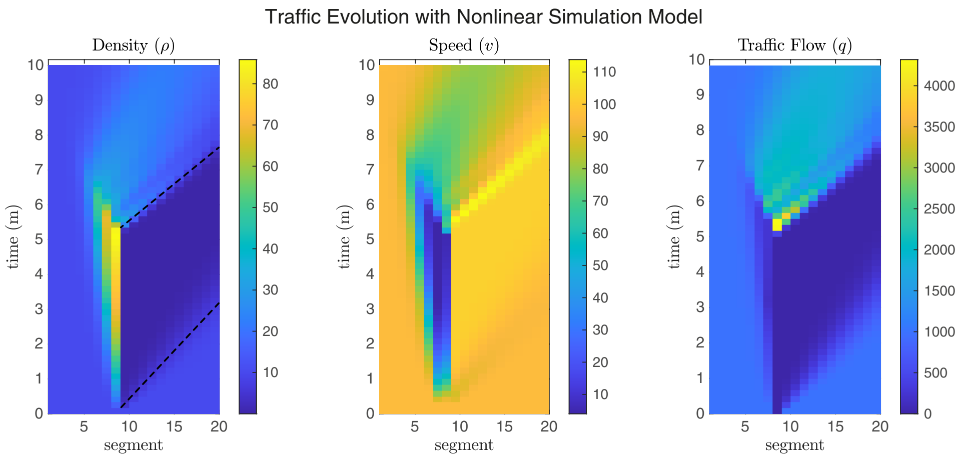


Figure 4-1: Case with a zero-length bridge being open for 5 minutes, starting from a steady state solution (where $\rho = 10.42$ veh/km and $v = 96.01$ km/h) that resulted from a constant inflow of 1000 veh/h. The maximum value of the density is $\rho = 85.84$ veh/km (reached one time step before the bridge closes again). Linear speed trajectories are shown in the density plot with black dashed lines. The TTS in this simulation is 22.18 veh·h.

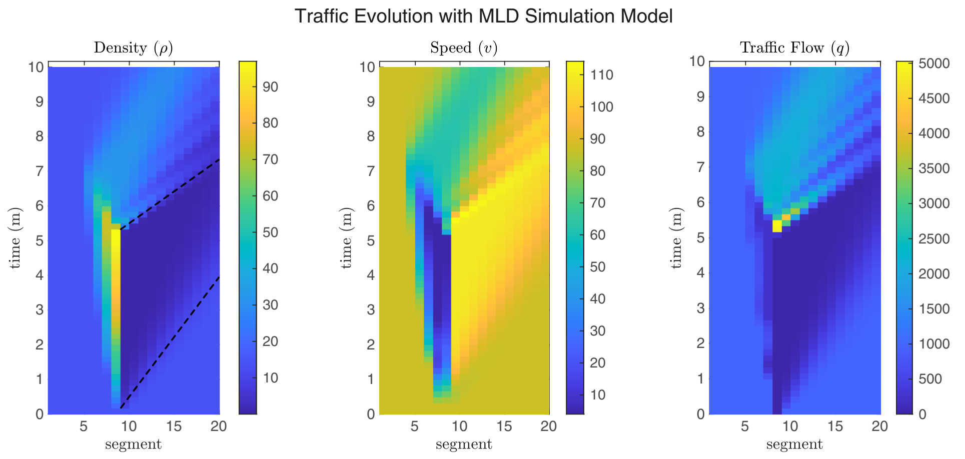


Figure 4-2: Case with a zero-length bridge being open for 5 minutes, starting from a steady state solution (where $\rho = 14.73$ veh/km and $v = 87.23$) that resulted from a constant inflow of 1000 veh/h. The maximum value of the density is $\rho = 96.97$ veh/km (reached one time step before the bridge closes again). Linear speed trajectories are shown in the density plot with black dashed lines. The TTS in this simulation is 27.27 veh·h.

solution, where $\rho = 10.42$ veh/km and $v = 96.01$ km/h for all segments in the network.

The traffic evolution is shown in Figure 4-1. We observe a build up of traffic in the segments upstream from the bridge from the moment it opens until it closes.

Furthermore, what stands out is that the maximum value of the density is 85.84 veh/km at segment i_{u}, when the bridge has been open for 5 minutes. The desired speed at that value equals $V(85.84) = 4.5$ km/h. However, when the bridge closes at the subsequent time step, the speed at that upstream segment increases to 50.38 km/h, while the density hardly changes. This results in an extremely high flow, with a maximum of 4318 veh/h (normally, the maximum flow is 2000 veh/h) which only lasts for 2 time steps. We note that this is unavoidable METANET behaviour that is due to the relatively large density difference between the upstream and the downstream segment. Concluding, we can state that the results in Figure 4-1, obtained with the nonlinear model, are satisfactory.

Mixed Logical Dynamical Model

In this section, we perform the same case study as before, but now using the MLD model as simulation model. We use the settings with the PWA configuration with the highest accuracy in Figure 3-4, i.e. we approximate $\hat{V}$ in 3 regions and $\hat{q}$ in 5 regions. We use the same network and the same parameters as before, however, when we start with the same constant traffic demand of $d_o = 1000$ veh/h, we arrive at a steady state with $\rho = 14.73$ veh/km and $v = 87.23$ km/h (these values differ from the nonlinear model due to the PWA approximations in the MLD model).

The result of this simulation is plotted in Figure 4-2. Overall, this traffic evolution is an acceptable approximation of the traffic evolution in the previous case study, visualised in Figure 4-1. A difference can be observed in the traffic speeds. The linear trajectories plotted in Figures 4-1 and 4-2 provide information about the pace of the traffic evolution. The steeper this trajectory, the slower the traffic moves. When we compare the lower trajectory in both images, we observe that the speed in the MLD model is lower than in the nonlinear model, given a steady-state flow of 1000 veh/h on all segments. On the other hand, the speed at which the traffic starts up after the bridge closes is a little higher in the MLD model than in the nonlinear model (which can be observed from the upper trajectory in both figures).

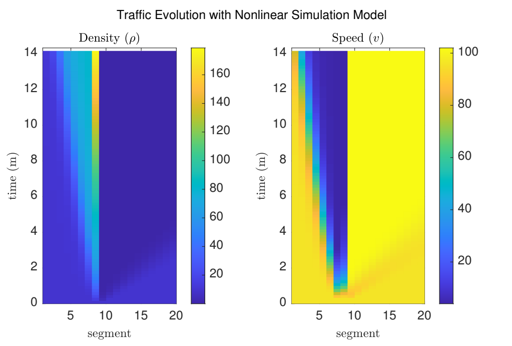
Furthermore, the maximum density, observed to be 96.97 veh/km, is bit higher than the maximum density observed in the simulation with the nonlinear model, which is 85.84 veh/km. This happens partly because the steady state solution obtained with the MLD model (the initial condition of the simulation) has a higher density than the steady state solution that we obtained with the nonlinear model, and partly because of the approximation error that we have with the MLD model.

Lastly, we compare the maximum flow. In the simulation with the MLD model, we find a maximum flow of 5027 veh/h. This was 4318 veh/h in the simulation with the nonlinear model.

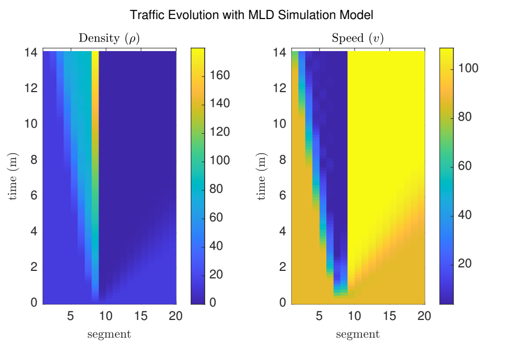
In Section 3-1, we have stated that our anticipation was a 15% error in the TTS w.r.t. the nonlinear model when the only approximation used is the flow function with a PWA function with 4 regions. Comparing the two simulations in Figures 4-1 and 4-2, we have a 19% relative error in the TTS w.r.t. the nonlinear model. To put this into perspective, we note that the calculation of the error is more fair when we start the MLD simulation with the same initial condition as the nonlinear model. In doing so, we obtain a TTS for the simulation with the MLD model of 23.22 veh·h, which is only a 5.2% relative error w.r.t. the nonlinear model. Furthermore, this result could be improved by simply adding more regions to the PWA approximation of the flow function. Note however, that the aim is to use the MLD model as prediction model in combination with MPC. We observe that the MLD model captures the dynamics of an open bridge to a certain extent, so that the model is suitable to be used as prediction model for routing in an MPC context. We conclude that the MLD approximation of the nonlinear model is acceptable, and if necessary can be further improved by adding more regions to the PWA approximation of the flow function.

Amount of Time It Takes To Reach the Maximum Density

Next, we determine how long a zero-length bridge can maximally be open, when no segment would be allowed to exceed the maximum density ρ_{jam}. In Section 3-3-2 we have described that we expected $\rho \leq \rho_{\text{jam}}$ would be the first (soft) bound to be crossed when a bridge is opened for a very long time. This simulation is performed to ensure that the model behaves as expected. Furthermore, we want to make sure that the maximum amount of time a bridge can be open after the bridge queue w_b has reached its maximum queue length $M_{w,b}$, is much larger than the 5 minutes we demand the bridge to be open.



(a) The density on segment i_{u} crosses the maximum density of $\rho_{\text{jam}} = 180$ veh/km after 86 time steps, i.e. 14.3 minutes. The TTS in this simulation is 39.98 veh·h.



(b) The density on segment i_{u} crosses the maximum density of $\rho_{\text{jam}} = 180$ veh/km after 86 time steps, i.e. 14.3 minutes. The TTS in this simulation is 46.08 veh·h.

Figure 4-3: Case study to determine the amount of time it takes to reach the maximum density ρ_{jam}, starting from a steady-state solution that resulted from a constant inflow of 1000 veh/h, on both the nonlinear and the MLD model.

In Figure 4-3 we have plotted an extreme scenario with both the nonlinear and the MLD model. In both cases we commenced with a steady-state solution with a demand $d_o = 1000$ veh/h and opened the bridge at the start of the simulation. We observe that in both solutions, the segment i_{u} exceeds the maximum density after 14.3 minutes. We note that for a higher traffic demand, this number is lower. We conclude that the MLD estimation of METANET is quite accurate.

We have observed that a bridge can be open for 14.3 minutes before the maximum density is reached. However, if this time window is to be further extended, we recommend reducing the minimum speed v_{min} during the time a bridge is open. In a preliminary case study with the nonlinear model used as simulation model where we use $v_{\text{min}} = 0,4$ km/h, we were able to open the bridge for 55 minutes without exceeding the maximum density on any segment.

4-3-2 Store-and-Forward Bridges

Maximum Queue Length

The value of the maximum queue length $M_{w,b}$ can be chosen relatively arbitrarily, based on the length of the queue area, the number of vehicles that fit in it, or the time the bridge needs to be open. We have chosen to set

$$M_{w,b} = 50.$$

We will use this value in the case studies below. Under an average traffic demand of $d_o = 1000$ veh/h, this implies a queue length of 278 m, which fills in 3 minutes. Furthermore, this means that when we open a bridge for more than 3 minutes, we can see the effect of both the store-and-forward and the zero-length bridge dynamics.

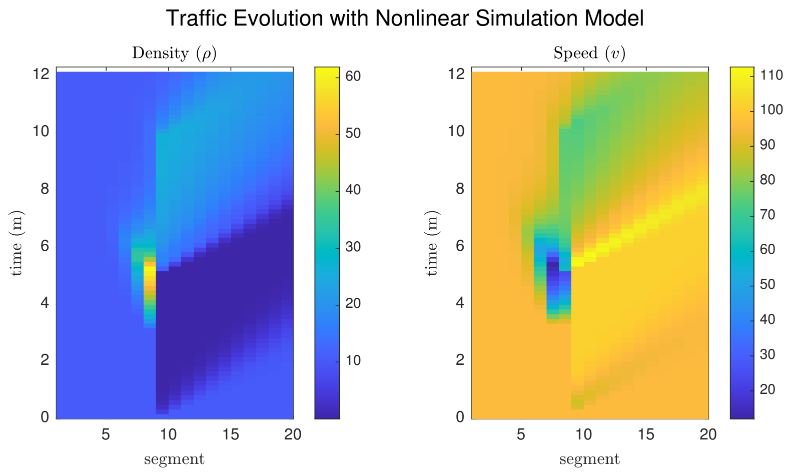


Figure 4-4: Case with a store-and-forward bridge, with $M_{w,b} = 50$, being open for 5 minutes, starting from a steady state solution (where $\rho = 10.42$ veh/km and $v = 96.01$ km/h) that resulted from a constant inflow of 1000 veh/h. The maximum value of the density is $\rho = 61.90$ veh/km (reached the first time step after the bridge is closed again). The TTS in this simulation is 22.58 veh·h.

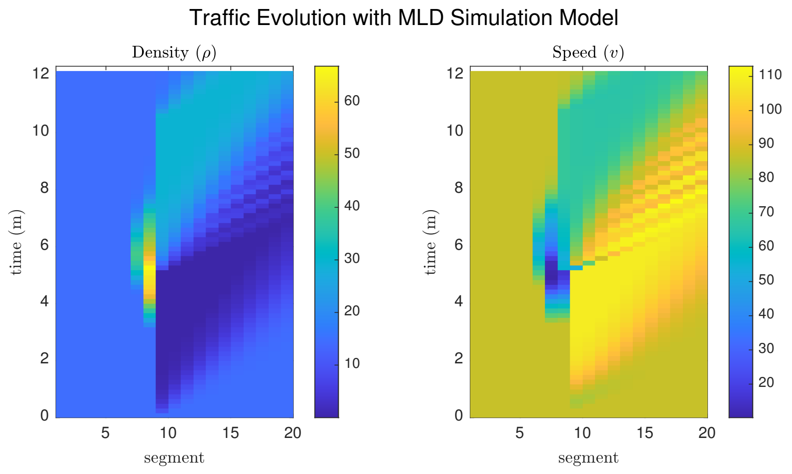


Figure 4-5: Case with a store-and-forward bridge, with $M_{w,b} = 50$, being open for 5 minutes, starting from a steady state solution (where $\rho = 14.73$ veh/km and $v = 87.23$ km/h) that resulted from a constant inflow of 1000 veh/h. The maximum value of the density is $\rho = 66.84$ veh/km (reached the first time step after the bridge is closed again). The TTS in this simulation is 28.97 veh·h.

Nonlinear Model

Next, we perform a case study on the same single-road network using the nonlinear model with one store-and-forward bridge, located between segment 8 and 9. As in the previous simulations, we have a constant traffic demand of $d_o = 1000$ veh/h and we start with a steady-state solution that resulted from this demand. The result of the simulation with the nonlinear model is plotted in Figure 4-4. We observe an unaltered density with respect to the initial condition in the upstream segments during the first 3 minutes in which the bridge is open. After this time period, the bridge queue element is completely filled with traffic and the traffic starts stacking up in the upstream segments. After 5 minutes, the bridge closes again, and the queue element starts emptying. This process lasts until at 10 minutes (61 time steps), the queue is empty. We observe that, when compared to the zero-length bridges, mainly the upstream segments are less affected by the opening of a store-and-forward bridge.

Mixed Logical Dynamical Model

In Figure 4-5, we have performed the same case study, but now using the MLD model. We can again observe that the results are a good approximation of the results obtained with the nonlinear model (plotted in Figure 4-4). The bridge queue is empty at time step 64, which is reasonably close to the nonlinear model (61 time steps). Naturally, we observe the same speed differences as with the zero-length bridges: faster at the steady-state solution (with a flow of 1000 veh/h on all segments) and slower as the traffic flow starts up again after the closing of the bridge.

4-4 Summary

In this chapter, we have given a brief overview of how the LP and MLD models can be used as prediction models in an MPC context, where we mainly discussed the means to tune the controller. Thereafter, we have analysed an N-step-ahead simulation (on a simple network with no control applied) with the LP model and we have show that such a simulation is actually a linear multilevel programming problem. This problem is much harder to solve than the single LP problem that we expected it to be. Moreover, we have left solving this problem and improving the method for future research. Subsequently, we have performed a number of case studies on only the MLD and nonlinear model. These studies are a proof of concept of the store-and-forward bridges and of the zero-length bridges, a special case of the former. The store-and-forward bridges yielded satisfactory results: the outcome of the case studies was as expected. Furthermore, at this point the nonlinear model has to be tuned on a real traffic scenario in order to evaluate how accurately it approximates reality. Moreover, the results obtained with the MLD model provided an acceptable approximation of the results obtained with the nonlinear model.

Chapter 5

Conclusions and Recommendations

In the final chapter of this thesis, we give a project summary in Section 5-1, after which we highlight our contributions to the literature in Section 5-2. Lastly, in Section 5-3, we list a number of ideas for future research.

5-1 Project Summary and Discussion

In this thesis we have considered the following research questions:

> **How can we route traffic through a motorway network, where certain roads are blocked at known time periods, with as objective to minimise the total time spent by all vehicles in the network?**
> **How can this be done in the most computationally efficient way?**

We have regarded these questions as a dynamic traffic routing problem and addressed them as a search for an algorithm that efficiently controls the traffic flows through a network. We have focussed on finding a computationally fast solution that potentially could be implemented in real time, and in the best case could be extended to other fields of science. Our approach in answering these research questions is described next.

We have selected the macroscopic (i.e. relatively low-detailed) motorway model METANET to simulate a traffic network. Furthermore, since METANET had been successfully used before in combination with Model Predictive Control (MPC) [31,49], we have selected MPC as a promising control method for our research. From the perspective of computation times, METANET is a nonlinear and nonconvex model and therefore not necessarily the most straightforward choice for fast state predictions with MPC. However, a major motivation for this choice in our research was the work by Groot et al. [34, 35], who derived a simplification of METANET that consists of the Piecewise-Affine (PWA) approximation of the nonlinear functions in the model. This approximated model fits a Mixed Logical Dynamical (MLD) description that can be used in combination with MPC and that can be solved with Mixed Integer Linear Programming (MILP).

We have proposed a Linear Programming (LP) model that approximates METANET, i.e. a simulation model that can be evaluated with LP. The expectation beforehand was that we could write the MPC implementation with our LP model in a single LP problem, which we had expected to result in a major calculation speed improvement compared with Groot et al. [34, 35]. However, when this novel model is used in an N-step-ahead prediction (or in combination with MPC), N consecutive LP problems need to be solved instead of a single LP problem. It remains to be seen whether the resulting linear multilevel programming problem results in a speed improvement compared to the existing MLD model.

Furthermore, we have expanded the existing MLD model and the nonlinear model with a novel store-and-forward bridge element. The implementation of this element includes the opening and closing of a bridge and the queueing of arriving traffic in front of an open bridge. The element is therefore capable of simulating a predicted disturbance, and more specifically a blocked road condition. The effectiveness of this novel bridge element has been evaluated in a case study, from which we have concluded that the nonlinear store-and-forward bridges functioned as expected and that the MLD model provides an acceptable approximation of the nonlinear model.

To the best of our knowledge the most computationally efficient way of solving an N-step-ahead simulation, that has an underlying model that is an approximation of METANET, is given by Lu et al. [57, 58]. Their approximation of METANET is used in combination with MPC and is written in a single LP problem. This method has been used in real-time traffic control and has shown that it can, in some cases, reduce traffic congestion [74]. The approach uses a rough approximation of METANET and relies heavily on driver compliance, which they justify by stating that if the density is high enough even 30% driver compliance with the variable speed limits will force the rest of the drivers to also reduce their speed and, secondly, that when the variable speed limits are strictly enforced, the actual speed will be close to the designed variable speed limits. The desired variable speed limit at all segments in the network is used as a predetermined control variable. This value is determined on the basis of an elaborate design strategy that is partly based on either historic data or on the operator's experience. The desired variable speed limits are used to substitute the speed in the density update function (2-4) and to substitute the nonlinear desired speed function V in (2-5). Furthermore, the on-ramp and origin queue dynamics are predetermined by predicting the traffic demand. These predetermined variables are subsequently used to solve the MPC problem that is now an LP problem. The control inputs of this MPC problem are the coordinated ramp metering rates. Furthermore, the objective function is a combination of both minimising the Total Time Spent (TTS) and maximising the total travel distance.

In this thesis we have proposed a completely different approach for creating an approximation of METANET that can be evaluated with LP (that is, only when used as simulation model). From the field test in [74], we know that the control method proposed by Lu et al. did not improve the traffic flow in all cases. This suggests that there is room for improvement. Furthermore, since our approximation of METANET is closer to the original nonlinear model, we suggest that the prediction models in our approach could have a higher accuracy. We conclude that there is still a significant need for a model approximation or simplification of METANET (or nonlinear functions in general) that is relatively accurate and that can be used in real time in combination with MPC.

5-2 Contributions

With respect to the research questions, we summarise the contributions of this thesis:

1. **The LP model can equally well be used in a single-step-ahead simulation as the MLD model.**
   In this thesis, we have proposed a model description that is an approximation of METANET and that we can evaluate using LP. This model, when used as a *simulation* model, is equivalent with the already existing MLD model (which is evaluated with MILP). Moreover, the methods that we have used to capture the complexities of METANET in a simple LP format, could provide an alternative in many other engineering applications.

2. **When the LP model is used as prediction model in combination with MPC, a linear multilevel programming problem needs to be solved.**
   We have not managed to write the MPC implementation of the LP model in a computationally very light, single LP problem. Instead, we have presented the linear multilevel programming problem that solves a general N-step-ahead prediction with the LP model.

3. **METANET has been extended with a store-and-forward element.**
   We have added a store-and-forward element to METANET. This element can be used to model predicted disturbances. More specifically, we have used it to model bridges. Furthermore, we have specified a special case of the store-and-forward element with a zero-length queue ($M_{w,b} = 0$) which we have called a zero-length bridge. We have described how to add the store-and-forward element to METANET and to the MLD model, but not how to add it to the LP model.

5-3 Recommendations

In this section, we discuss some recommendations for future research. These are ideas that have been formed during the course of this project. Firstly, we discuss recommendations concerning the control implementation in Section 5-3-1. Secondly in Section 5-3-2, we have some recommendations concerning model improvements, mainly concerning the LP model. Finally, we have a number of recommendations with respect to the validation and implementation of our models on real traffic situations in Section 5-3-3.

5-3-1 Recommendations Concerning the Control Implementation

Implementing exponentially decreasing weighing factors in the LP-MPC problem
We propose a weight selection method that might enable solving an N-step-ahead simulation with the LP model in a single LP problem. When we consider Table 4-1 again, we observe that even in the first time step, the prediction of the single LP solution deviates from the correct MLD-based solution. Therefore, we suggest that a weight on the dummy variables, that decreases at each consecutive time step might improve the results. The corresponding cost function for the LP problem with time-dependent weights is given in (3-40).

To obtain a starting point for the selection of values of the weighting factors, we analyse the first constraint in (4-4), being

$$\hat{V}_1(1) \geq a_V \left(\rho_1(0) + \frac{T_\mathrm{s}}{L}[\hat{q}_o(0) - \hat{q}_1^+(0) + \hat{q}_1^-(0)] \right) + b_V. \tag{5-1}$$

Here, we recognize the recursiveness in these constraints when we write

$$\begin{aligned} \hat{V}_1(1) &\geq r[\hat{q}_o(0) - \hat{q}_1^+(0) + \hat{q}_1^-(0)] + \dots \\ \hat{V}_1(2) &\geq r[\hat{q}_o(1) - \hat{q}_1^+(1) + \hat{q}_1^-(1)] + \dots, \end{aligned} \tag{5-2}$$

where

$$r = \frac{a_V T_\mathrm{s}}{L}.$$

From here, we have a rough estimate of the weighing factors for $\hat{V}$ as

$$w_{\hat{V}}(k) = r^k. \tag{5-3}$$

With the same approach we can determine weighing factors on all the dummy variables.

A major disadvantage of these exponentially decreasing weighing factors is that, especially for larger prediction horizons, the weighing factors can become so small that numerical issues are very probable.

Solving the linear multilevel programming problem with the LP model used as prediction model

We come back to the linear multilevel programming problem posed in (4-6) and discuss some methods of solving it. Multilevel programming has been used in combination with MPC before [1,45,48], albeit not always in the same framework that we have in (4-6). Sometimes, the MPC problem is incorporated in one single level of the multilevel problem. In our definition however, each time step on the receding horizon of the MPC problem corresponds to a level in the hierarchical optimisation problem. Therefore, we recommend to use a general solver for a multilevel problem.

Many of the algorithms rely on the assumption that a lower-level problem that is convex and satisfies some specific regularity conditions can be rewritten using its equivalent Karush-Kuhn-Tucker conditions. The resulting problem can in the linear case be solved with MILP [1, 14]. This is a promising direction. Even though the MLD model that we have studied in this thesis was also solved using MILP, we can only conclude whether either of the two outperforms the other when a quantitative comparison on evaluation time is performed with both models on the same simulation problem.

In [8], a branch-and-bound algorithm is proposed to solve the linear multilevel programming problem.

Another method for solving this complex problem is provided in [67], where a linear multilevel problem is studied, similar to the problem we have in (4-6). They rewrite the problem so that it can be solved with fuzzy programming.

5-3-2 Recommendations Concerning Model Improvements

Adding store-and-forward bridges to the LP model

In this thesis we have derived the relations for zero-length bridges in the LP model. However, the relations for store-and-forward bridges in this model have not been derived. Rewriting these equations for usage with the LP model is a complex extra step. Since we have not (yet) succeeded in making the comparison with the MLD model to establish whether one of the two solution approaches outperforms the other, we do not know whether the linear multilevel problem can be solved more efficiently than the MILP approach that already exists. Therefore, we have left these store-and-forward bridge relations for future research.

Adding vehicular emissions and fuel consumption to the LP model

When we complement METANET by the VT-macro model [75], we can take vehicular emissions and fuel consumption into account. Groot et al. [34] use PWA approximations of the functions for the total emissions and fuel consumption that are compatible with the MLD formulation they use. These PWA approximations must be rewritten in such a way that they are convex before they are also compatible with the LP formulation of METANET. This could always be possible when an approximation error of a certain extent is accepted. The question whether a convex approximation can be found for the VT-macro model with a low approximation error, is left for future research.

Making the models robust for disturbance rejection in case of unexpected topology changes

In this thesis we have focussed on predictable disturbances. We have used bridges as a physical embodiment of this predicted disturbance. However, we could equally well have used the level crossing (an intersection with a railroad). A mathematical description of a level crossing in METANET would be practically identical to the bridge models that we have discussed in this thesis.

However, a blocked road due to a traffic accident does not fall under this category. Traffic lights, on the other hand, could be an interesting extension to this research. Controlling the traffic by routing it around the busy traffic lights whilst keeping the queue lengths minimal can be contradictory and it would probably be more efficiently solved with a distributed control method. Blocked roads due to road maintenance can be interesting, but it has a very different time scale from opening and closing bridges. Therefore, a model that queues traffic in front of the blocked road is most likely not needed, because the road is blocked for a very long time and a detour is suggested. However, it could be more efficient (system optimal) to dynamically reroute the traffic (over multiple routes) instead of providing a static detour.

Accurately determining the domain of $q^{\pm}_{\mathrm{PWA}}$

We have run some tests in the 3rd and 4th configuration in Figure A-3, without the imposed safety region. These tests yielded unsatisfactory results because some values fell outside the PWA regions. This can be explained by the difference between V and $\hat{V}$ and the fact that the METANET values do not lie exactly on the fundamental line but somewhere close. Therefore, we suggest that finding an accurate way of globally determining $\min(\rho \pm v)$ and $\max(\rho \pm v)$ would be valuable. In some configurations this might result in using less PWA regions in $q^{\pm}_{\mathrm{PWA}}$, which can be a computational benefit for the MLD model, especially when used as a prediction model in an MPC context.

5-3-3 Recommendations Concerning Model Validation and Real-Time Implementation

Testing the accuracy of the model for bridges on historical data
We have proposed a simple store-and-forward model to simulate bridges in METANET. However, in order to evaluate how well this model represents reality, i.e. real-live traffic queueing in front of an open bridge, a comparative study should be performed using historical data of a real traffic network. Using this data, our simulation model should be validated and tuned.

Testing the accuracy of the approximated model on historical data.
We can perform a case study on an existing motorway network. In order to do so, we need to collect data (traffic distribution, inflows, and splits) for a certain amount of time during which we have significant traffic demands in this network, where there is no routing applied. Then, if we extract the traffic splits at interchanges, we can subsequently compare the performance of our MPC implementation with the performance of the uncontrolled network. In this way, we can evaluate the difference in performance between an uncontrolled and a controlled traffic situation without doing it in real time.

A difficulty that is encountered when implementing a real-time solution in a real traffic network is that the routes that vehicles take can only be advised, but not controlled. A first step in that direction would be to use an advisory implementation on a motorway traffic network, which could be achieved with Dynamic Route Information Panels (DRIPs) [21, 42, 43].

In a situation with (future) smart cars, distributed control is preferably used to route the traffic. A low-level controller in the cars should take care of the faster dynamics that are associated with the driving of the car (think of lane changing and decelerating and accelerating in jams). A high-level controller then controls the slower dynamics and thus ensures efficient routing and that the pressure is spread equally over the network. However, designing this high-level controller is a challenging task. The hardest challenge of which is in the second research question; the computational efficiency. If we are ever to implement real-time MPC in a real traffic network using DRIPs that can provide the passing vehicles with traffic information, it is important that we have an accurate model that can be used in a predictive control implementation and that has a low time complexity.

Implementing an observer for real-time traffic control
Because we use a simulation model as a substitute for reality, we have full knowledge of the states and outputs at each time step when we perform a case study. If we were to implement the controller on a real motorway, we would have to suffice with real-time network measurements. Some system parameters, like the flow at a certain point, would be more easily obtained than others, like the TTS of the entire network at a moment in time.

When controlling a real network in real time, the only way to track how well we are doing is to translate these measurements to the outputs and states in our models. A filter (e.g. an Extended Kalman Filter) or an observer can help with that translation.

Appendix A

Approximations of Functions in METANET

In this chapter, we discuss the Piecewise-Affine (PWA) approximated functions in more detail. The exact values of the optimal solutions are given as well as the settings with which they were obtained. Section A-1 discusses the desired speed function V_{PWA} and Section A-2 does this for the traffic flow $q^{\pm}_{\text{PWA}}(\rho \pm v)$.

A-1 Piecewise-Affine Approximation of the Desired Speed Function

Equation (A-1) below gives the PWA approximation in 2 regions of $V(\rho)$. The parameters of (A-1) are obtained by solving the least-squares problem, given in (2-13) with a multi-start Levenberg-Marquardt algorithm (see Section 2-2-3 for the details about this optimisation). The PWA approximation of $V[\rho]$ in 2 regions that we have used is:

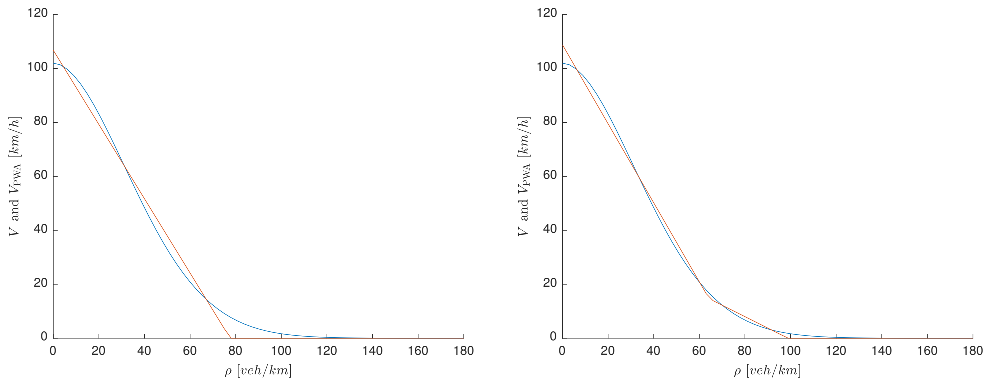
$$V_{\text{PWA}}(\rho) = \begin{cases} -1.377\rho + 106.8 & \text{for } 77.55 > \rho \geq 0 \\ 0 & \text{for } 77.55 \leq \rho. \end{cases} \tag{A-1}$$

Equation (A-2) gives the PWA approximation of V in 3 regions:

$$V_{\text{PWA}}(\rho) = \begin{cases} -1.465\rho + 108.8 & \text{for } 64.27 > \rho \geq 0 \\ -0.4239\rho + 41.90 & \text{for } 64.27 \leq \rho < 98.85 \\ 0 & \text{for } 98.85 \leq \rho. \end{cases} \tag{A-2}$$

We set the last region of each PWA approximation to $\rho = 0$. This makes sure that the approximated desired speed function and therefore the flows never become negative. For all density values $\rho > 77.55$ veh/m in the PWA approximation with two regions we have

$V_{\mathrm{PWA}}(\rho) = 0$ km/h and so the traffic is forced to a stop at a relatively low density. This is much sooner than when using the nonlinear model, where $V(77.55) = 7.8$ km/h. In the PWA function with 3 regions, we have a more accurate approximation with $V(98.85) = 1.8$ km/h.



(a) PWA approximation of desired speed function V in 2 regions.

(b) PWA approximation of desired speed function V in 3 regions.

Figure A-1: Two PWA approximations of the desired speed function V.

A-2 Piecewise-Affine Approximation of the Flow Function

In this section, we present the results from the four optimisation problems for the four different PWA approximations of flow function q, shown in Figure 3-4. They all were obtained by solving the least-squares problem posed in (2-13) with a multi-start Levenberg-Marquardt approach. Equations (A-3) to (A-6) give the results of the PWA approximations with 2 to 5 regions respectively:

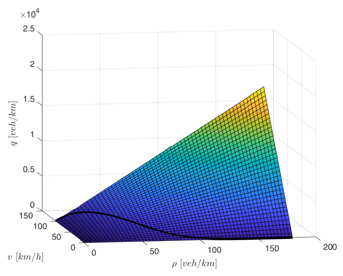
$$q^{\pm}_{\mathrm{PWA}}(\rho \pm v) = \begin{cases} -33.75(\rho \pm v) & \text{for } \rho < 0 \\ 33.75(\rho \pm v) & \text{for } \rho \geq 0 \end{cases} \tag{A-3}$$

$$q^{\pm}_{\mathrm{PWA}}(\rho \pm v) = \begin{cases} -58.04(\rho \pm v) - 3029 & \text{for} \quad -52.18 > \rho \\ 0 & \text{for} \quad -52.18 \leq \rho < 52.18 \\ 58.04(\rho \pm v) - 3029 & \text{for} \quad 52.18 \leq \rho \end{cases} \tag{A-4}$$

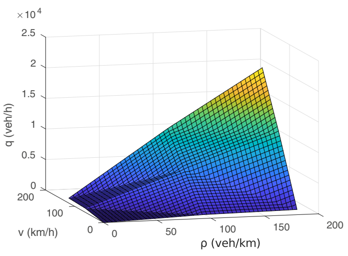
$$q^{\pm}_{\mathrm{PWA}}(\rho \pm v) = \begin{cases} -65.23(\rho \pm v) - 4050 & \text{for} \quad -80.91 > \rho \\ -15.17(\rho \pm v) & \text{for} \quad -80.91 \leq \rho < 0 \\ 15.17(\rho \pm v) & \text{for} \quad 0 \leq \rho < 80.91 \\ 65.23(\rho \pm v) - 4050 & \text{for} \quad 80.91 \leq \rho \end{cases} \tag{A-5}$$

$$q^{\pm}_{\mathrm{PWA}}(\rho \pm v) = \begin{cases} -71.32(\rho \pm v) - 4970 & \text{for} \quad -105.3 > \rho \\ -33.95(\rho \pm v) - 1036 & \text{for} \quad -105.3 \leq \rho < \; -30.52 \\ 0 & \text{for} \quad -30.52 \leq \rho < \; 30.52 \\ 33.95(\rho \pm v) - 1036 & \text{for} \quad 30.52 \leq \rho < \; 105.3 \\ 71.32(\rho \pm v) - 4970 & \text{for} \quad 105.3 \leq \rho. \end{cases} \tag{A-6}$$

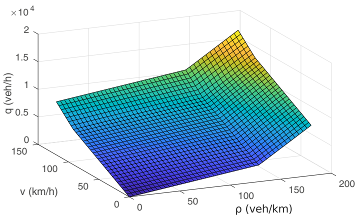
As opposed to the two-dimensional PWA identification method that has been used in this thesis (cf. Section 2-2-3), we show the difficulties in creating an accurate convex approximation of flow function q with a three-dimensional identification method [27] in Figure A-2. Running a least-squares optimisation in three dimensions, yields either an accurate solution that is nonconvex (and unsuitable for the Linear Programming (LP) model), or convex solution (suitable for the LP model) that is very inaccurate, cf. Figure A-2b and Figure A-2c.



(a) Function $q = \rho v$. The fundamental line, with equation $q_e = \rho V(\rho)$ is also shown.



(b) PWA, but nonconvex approximation of Figure A-2a.

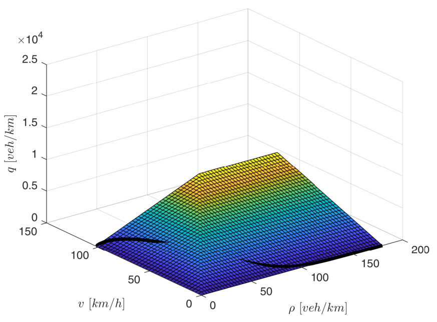


(c) PWA and convex, but inaccurate approximation of Figure A-2a.

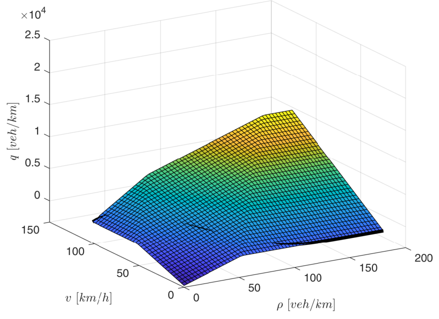
Figure A-2: These three figures show $q = \rho v$ and two PWA approximations of q. They are plotted for $\rho \in [0, 180]$ and $v \in [0, 120]$. These figures illustrate that a least-squares approximation that is both accurate and convex is not easily found.

The plots of the PWA functions against the original quadratic function $q^{\pm} = (\rho \pm v)^2$, as depicted in Figure 3-4, give a good representation of the accuracy of each configuration with different levels of simplification. In Figure A-3 however, the four configurations of the (relatively accurate and possibly nonconvex) PWA flow functions $q_{\mathrm{PWA}}(\rho, v) = q^{+}_{\mathrm{PWA}}(\rho+v) - q^{-}_{\mathrm{PWA}}(\rho - v)$, are plotted in $\mathbb{R}^3$ along with the fundamental line and $q = \rho v$.

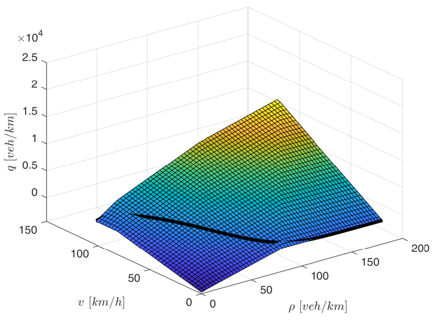
The first configuration, depicted in Figure A-3a, seems to be not even close to the original, but using only 1 decision variable, it is quite accurate around the fundamental line. The configuration in A-3b is closer to the fundamental line. Note however that it becomes negative for values in the vicinity of $\rho = 0 \;\wedge\; v = 0$, far from the fundamental line. The negative region is even larger in the configuration in A-3c, but again there is a safety region between the fundamental line and the negative values. The configuration in A-3d is the best configuration we used. It also has the closest resemblance to the original ($q = \rho v$). The negative region has even become smaller.



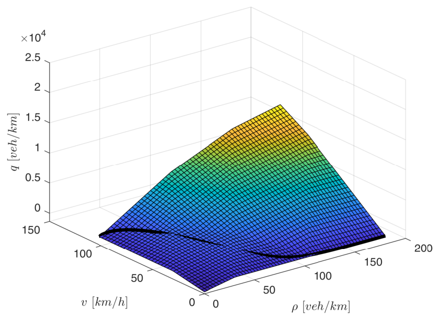
(a) Plot based on the PWA approximation of $q^{\pm}$ in 2 regions.



(b) Plot based on the PWA approximation of $q^{\pm}$ in 3 regions.



(c) Plot based on the PWA approximation of $q^{\pm}$ in 4 regions.



(d) Plot based on the PWA approximation of $q^{\pm}$ in 5 regions.

Figure A-3: Full function q and the four configurations of PWA approximations q_{PWA} plotted in three dimensions for $\rho \in [0, 180]$ and $v \in [0, 120]$. In each plot, the fundamental line $q_{\text{e}}(\rho) = \rho V(\rho)$ is shown. Some of the regions are negative even though we made choices specifically to prevent this.

A-3 Rewriting Equations in the Form of a Mixed Logical Dynamical Model

In this section we work out the conversion of two nonlinear equations in METANET to an Mixed Logical Dynamical (MLD) model. In Section A-3-1, we rewrite the desired speed function V and in Section A-3-2 the outflow q_o at an origin.

A-3-1 Desired Speed Function

As an example of how to rewrite a PWA approximation of a function to a set of equations that fit the MLD framework, we work out the equations for the PWA approximation of the desired speed function V, which is denoted as V_{PWA}. The conversion of the desired speed function V_{PWA} in two regions to MLD form is derived in this section. It is straightforward to expand this to more than two regions or dimensions.

The PWA approximation of $V(\rho)$ is denoted as

$$V_{\mathrm{PWA}}(\rho) = \begin{cases} a_1\rho + b_1 & \text{if } \rho \le s_1 \\ a_2\rho + b_2 & \text{if } \rho > s_1. \end{cases}$$

The decision variable $\delta_V(k)$ is used to indicate which region of the PWA function is active at time step k. It must change value at the intersection $\rho(k) = s_1$ where $a_1\rho(k)+b_1 = a_2\rho(k)+b_2$, so

$$\rho(k) \ge s_1 \Leftrightarrow [\delta_V(k) = 1],$$

which can be written as

$$g_V(x) \le 0 \Leftrightarrow [\delta_V(k) = 1],$$

where $g_V(x) = s_1 - \rho(k)$. This results in the last two constraints in (A-7).

The approximation $\hat{V}(\rho)$ is incorporated in the state space equations, as

$$\begin{aligned}\hat{V}(\rho) &= a_1\rho(k) + b_1 + [a_2\rho(k) + b_2 - [a_1\rho(k) + b_1]]\,\delta_V(k) \\ &= a_1\rho(k) + [b_2 - b_1]\delta_V(k) + [a_2 - a_1]z_V(k) + b_1,\end{aligned}$$

where $z_V(k) = \delta_V(k)\rho(k)$. This results in the first four constraints in (A-7). Note that the first PWA region is active unless $\delta_V(k) = 1$. This saves one decision variable compared to using a one for each region, i.e. $\delta_{V,i}(k) = 1$ if $a_i\rho(k) + b_i$ is active.

When we work out (2-17) and (2-19), we arrive at the following constraints for the MLD system:

$$\begin{aligned} z_V(k) &\le M_{z_V}\delta_V(k) \\ z_V(k) &\ge m_{z_V}\delta_V(k) \\ z_V(k) &\le \rho(k) - m_{z_V}[1 - \delta_V(k)] \\ z_V(k) &\ge \rho(k) - M_{z_V}[1 - \delta_V(k)] \\ g_V(x) &\le M_{g_V}[1 - \delta_V(k)] \\ g_V(x) &\ge \epsilon + [m_{g_V} - \epsilon]\delta_V(k), \end{aligned} \tag{A-7}$$

where the minima and maxima are evaluated as

$$\begin{aligned} m_{z_V} &= \rho_{\min}, & M_{z_V} &= \rho_{\mathrm{jam}} \\ m_{g_V} &= s_1 - \rho_{\mathrm{jam}}, & M_{g_V} &= s_1 - \rho_{\min}. \end{aligned}$$

A-3-2 Outflow at an Origin

This minimisation of three parameters, given in (2-8) can be written in MLD form too. The method we use is derived from [6]. For simplicity, we write this minimisation of three parameters as

$$q_o(k) = \min[f_1(k), f_2(k), f_3(k)].$$

We rewrite this equation in MLD form, with three decision variables. This is realised by using

$$\hat{q_o}(k) = f_1(k)\delta_{q_o,1}(k) + f_2(k)\delta_{q_o,2}(k) + f_3(k)\delta_{q_o,3}(k), \tag{A-8}$$

with

$$\begin{aligned}[\delta_{q_o,1}(k) = 1] &\Leftrightarrow \quad [f_1(k) \leq f_2(k) \text{ and } f_1(k) \leq f_3(k)] \\ [\delta_{q_o,2}(k) = 1] &\Leftrightarrow \quad [f_2(k) \leq f_1(k) \text{ and } f_2(k) \leq f_3(k)] \\ [\delta_{q_o,3}(k) = 1] &\Leftrightarrow \quad [f_3(k) \leq f_1(k) \text{ and } f_3(k) \leq f_2(k)].\end{aligned} \tag{A-9}$$

Note that these inequalities imply an exclusive or condition on the decision variables, i.e. exactly one of the three binary decision variables is equal to 1 at each time step. So we have the equality condition

$$\sum_{n=1}^{3} \delta_{q_o,n}(k) = 1 \ \forall k. \tag{A-10}$$

The conditions in (A-9) translate to the following inequality constraints:

$$\begin{aligned} f_1(k) - f_2(k) &\leq M_{12}[1 - \delta_{q_o,1}] \qquad & f_2(k) - f_3(k) &\leq M_{23}[1 - \delta_{q_o,2}] \\ f_1(k) - f_3(k) &\leq M_{13}[1 - \delta_{q_o,1}] \qquad & f_3(k) - f_1(k) &\leq M_{31}[1 - \delta_{q_o,3}] \\ f_2(k) - f_1(k) &\leq M_{21}[1 - \delta_{q_o,2}] \qquad & f_3(k) - f_2(k) &\leq M_{32}[1 - \delta_{q_o,3}] \end{aligned} \tag{A-11}$$

with

$$\begin{aligned} M_{12} &= \max[f_1(k) - f_2(k)], \quad & M_{21} &= \max[f_2(k) - f_1(k)], \quad & M_{31} &= \max[f_3(k) - f_1(k)] \\ M_{13} &= \max[f_1(k) - f_3(k)], \quad & M_{23} &= \max[f_2(k) - f_3(k)], \quad & M_{32} &= \max[f_3(k) - f_2(k)]. \end{aligned}$$

Lastly, (A-8) has three multiplications that we rewrite with the use of an auxiliary variable z, as we have explained before, by writing

$$z_{q_o,1}(k) = f_1(k)\delta_1(k), \quad z_{q_o,2}(k) = f_2(k)\delta_2(k). \quad z_{q_o,3}(k) = f_3(k)\delta_3(k),$$

So we obtain

$$\hat{q}_o(k) = z_{q_o,1}(k) + z_{q_o,2}(k) + z_{q_o,3}(k), \tag{A-12}$$

where $z_{q_o,1}(k)$, $z_{q_o,2}(k)$, and $z_{q_o,3}(k)$ are (as before) subject to the constraints given in (2-19):

$$\begin{aligned} z_{q_o,n}(k) &\leq M_{q_o,n}\delta_{q_o,n}(k) & &\text{for } \ n \in \{1,2,3\} \\ z_{q_o,n}(k) &\geq m_{q_o,n}\delta_{q_o,n}(k) & &\text{for } \ n \in \{1,2,3\} \\ z_{q_o,n}(k) &\leq f_n(k) - m_{q_o,n}[1 - \delta_{q_o,n}(k)] & &\text{for } \ n \in \{1,2,3\} \\ z_{q_o,n}(k) &\geq f_n(k) - M_{q_o,n}[1 - \delta_{q_o,n}(k)] & &\text{for } \ n \in \{1,2,3\}, \end{aligned} \tag{A-13}$$

where $M_{q_o,n} = \max f_n(k)$ and $m_{q_o,n} = \min f_n(k)$.

Appendix B

Network Flow Problems

This chapter is an extension to Section 2-1-3, where we discussed the effectiveness of the use of a minimum-cost flow problem as a solution to the dynamic routing problem that we considered in this thesis. Here, we state the definitions of both the minimum-cost flow problem in Section B-1 and the multi-commodity flow problem in Section B-2. Furthermore, we rewrite the definition of the minimum-cost flow problem to a Linear Programming (LP) problem in Section B-3.

B-1 Minimum-Cost Flow Problem

The minimum-cost flow problem seeks for the optimal flow on each link that minimizes the cost function $W_{ij}F_{ij}$ when a required flow a is to be transported through the network from origin node s to destination node d. We define the problem as

$$\begin{array}{llcll} & \min\limits_{F_{ij}} \sum\limits_{(i,j)\in M} W_{ij}F_{ij} & & & \\ \text{s.t.} & & & & \\ \text{Inflow at node } s & \sum\limits_{j\in \mathrm{adj_o}(s)} F_{sj} & = & a & \\ \text{Outflow at node } d & \sum\limits_{i\in \mathrm{adj_i}(d)} F_{id} & = & a & \\ \text{Inflow equals outflow} & \sum\limits_{i\in \mathrm{adj_i}(u)} F_{iu} - \sum\limits_{j\in \mathrm{adj_o}(u)} F_{uj} & = & 0 & \forall\, u \in V \setminus \{s,d\} \\ \text{Capacity at each link} & F_{ij} - C_{ij} & \leq & 0 & \forall\, (i,j) \in M. \end{array} \tag{B-1}$$

The minimum-cost flow problem has been known and studied for a long time, e.g. by Ford and Fulkerson [29] and Klein [51].

B-2 Multi-Commodity Flow Problem

The multi-commodity flow problem is a generalisation of the minimum-cost flow problem that allows for a set $\mathbb{P}$ of origin-destination pairs. It is a set of paired origin and destination vertices, coupled with a their corresponding required flows. We indicate the n^{th} pair as $\text{OD}(n) = (s_n, d_n, a_n)$, a subset of $\mathbb{P}$ with a required flow a_n from source s_n to destination d_n. We have a flow adjacency matrix $F_{ij,n}$ for each origin-destination pair n. The complete flow adjacency matrix, summed over all origin-destination pairs becomes

$$F_{ij} = \sum_{n\in\mathbb{P}} F_{ij,n}.$$

We write the multi-commodity flow problem as

$$\begin{aligned}
&\min_{F_{ij,n}} \sum_{(i,j)\in M} \left[W_{ij}\big(\sum_{n\in\mathbb{P}} F_{ij,n}\big) \right] \\
\text{s.t.}\quad & \\
&\sum_{j\in\text{adj}_\text{o}(s_n)} F_{s_n j,n} && = a_n && \forall\, n\in\mathbb{P} \\
&\sum_{i\in\text{adj}_\text{i}(d_n)} F_{i d_n,n} && = a_n && \forall\, n\in\mathbb{P} \\
&\sum_{i\in\text{adj}_\text{i}(u)} F_{iu} - \sum_{j\in\text{adj}_\text{o}(u)} F_{uj} && = 0 && \forall\, u\in V\setminus\{s_n,d_n\}_{n\in\mathbb{P}} \\
&\big(\sum_{n\in\mathbb{P}} F_{ij,n}\big) - C_{ij} && \leq 0 && \forall\,(i,j)\in M.
\end{aligned} \tag{B-2}$$

B-3 Linear Programming Problem

The conversion to an LP problem is performed on the equations in (B-1). We rewrite the problem to

$$\begin{aligned}
&\min_{F_{ij}} \sum_{(i,j)\in M} W_{ij}F_{ij} \\
&\text{s.t.} \sum_{k\in V} F_{sk} = a \quad\text{and}\quad \sum_{k\in V} F_{dk} = -a \\
&\sum_{j\in V} F_{ij} = 0 \quad \forall\, i\in V\setminus\{s,d\} \\
&F_{ij} = -F_{ji} \quad \forall\, i,j\in V \\
&F_{ij} \leq C_{ij} \quad \forall\, i,j\in V.
\end{aligned} \tag{B-3}$$

We replaced the third constraint in (B-1) (inflow equals outflow) with two new constrains: the flow matrix has to be skew symmetric, i.e. $F_{ij} = -F_{ji}$, and the constraint that the sum of each row r in the flow matrix F_{rj} equals zero, except for the rows associated with s and d.

This conversion to an LP problem is not directly applicable to undirected graphs, however, it is possible after a transformation proposed by Lau [55]. A minor disadvantage of this transformation is that it increases the size of the LP problem significantly.

Bibliography

[1] R. Baker and C.L.E. Swartz. Interior point solution of multilevel quadratic programming problems in constrained model predictive control applications. *Industrial & engineering chemistry research*, 47(1):81–91, 2008.

[2] M. Barbehenn. A note on the complexity of dijkstra's algorithm for graphs with weighted vertices. *IEEE Transactions on Computers*, 47(2):263, 1998.

[3] R. Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16:87–90, 1958.

[4] A. Bemporad. Efficient conversion of mixed logical dynamical systems into an equivalent piecewise affine form. *IEEE Transactions on Automatic Control*, 49(5):832–838, May 2004.

[5] A. Bemporad, G. Ferrari-Trecate, and M. Morari. Observability and controllability of piecewise affine and hybrid systems. *IEEE Transactions on Automatic Control*, 45(10):1864–1876, 2000.

[6] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, 1999.

[7] A. Bemporad, F.D. Torrisi, and M. Morari. Optimization-based verification and stability characterization of piecewise affine and hybrid systems. In *International Workshop on Hybrid Systems: Computation and Control*, pages 45–58, 2000.

[8] H.P. Benson. On the structure and properties of a linear multilevel programming problem. *Journal of Optimization Theory and Applications*, 60(3):353–373, 1989.

[9] D. Braess, A. Nagurney, and T. Wakolbinger. On a paradox of traffic planning [Translated from German: D. Braess. Über ein Paradoxon aus der Verkehrsplanung. Unternehmensforschung, 12:258-268, 1968.]. *Transportation Science*, 39(4):446–450, 2005.

[10] P. Breton, A. Hegyi, B. De Schutter, and H. Hellendoorn. Shock wave elimination/reduction by optimal coordination of variable speed limits. In *Proceedings of the IEEE 5th International Conference on Intelligent Transportation Systems (ITSC'02)*, page 225–230, Singapore, Sept. 2002.

[11] C. Buisson, J.P. Lebaque, and J.B. Lesort. STRADA, a discretized macroscopic model of vehicular flow in complex networks based on the godunov scheme. In *Proceedings of the CESA'96 IEEE Conference*, France, 1996.

[12] M. Burger, M. van den Berg, A. Hegyi, B. De Schutter, and J. Hellendoorn. Considerations for model-based traffic control. *Transportation Research Part C*, 35:1–19, 2013.

[13] X. Cai, D. Sha, and C. K. Wong. *Time-Varying Network Optimization*, volume 103 of *International Series in Operations Research & Management Science*. Springer Science & Business Media, 2007.

[14] B. Colson, P. Marcotte, and G. Savard. Bilevel programming: A survey. *4or*, 3(2):87–107, 2005.

[15] K.L. Cooke and E. Halsey. The shortest route through a network with time-dependent internodal transit times. *Journal of Mathematical Analyis and Applications*, 14:493–498, 1966.

[16] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, Cambridge, MA, USA, 2009.

[17] D. Corona, I. Necoara, B. De Schutter, and T. van den Boom. Robust hybrid MPC applied to the design of an adaptive cruise controller for a road vehicle. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 1721–1726, San Diego, California, 2006.

[18] R. Courant, K. Friedrichs, and H. Lewy. On the partial difference equations of mathematical physics. *IBM Journal of Research and Development*, 11(3):215–234, March 1967.

[19] C.R. Cutler and B.L. Ramaker. Dynamic matrix control – a computer control algorithm. In *Proceedings of the 86th AIChE National Meeting*, Houston, TX, USA, 1979.

[20] C.F. Daganzo. The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory. *Transportation Research Part B: Methodological*, 28(4):269–287, 1994.

[21] F.P. Deflorio. Evaluation of a reactive dynamic route guidance strategy. *Transportation Research Part C*, 11(5):375–388, 2003.

[22] C. Diakaki, M. Papageorgiou, and T. McLean. Integrated traffic-responsive urban corridor control strategy in Glasgow, Scotland. *Transportation Research Record*, 1727(1):101–111, 2000.

[23] R. Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 2^{nd} edition, 2000.

[24] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[25] J. Doran. An approach to automatic problem-solving. *Machine Intelligence*, 1:105–124, 1967.

[26] D. Ferone, P. Festa, A. Napoletano, and T. Pastore. Reoptimizing shortest paths: from state of the art to new recent perspectives. In *18th International Conference on Transparent Optical Networks (ITCON)*, 2016.

[27] G. Ferrari-Trecate. Hybrid identification toolbox (HIT). `http://sisdin.unipv.it/lab/personale/pers_hp/ferrari/HIT_toolbox.php`, 2005.

[28] L.R. Ford, Jr. Network flow theory. *The RAND Corporation*, No. P-923, 1956.

[29] L.R. Ford, Jr. and D.R. Fulkerson. *Flows in Networks.* R-375-PR. The RAND Corporation, Santa Monica, CA, USA, 1962.

[30] M.L. Fredman and R.E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the Association for Computing Machinery*, 34(3):596–615, 1987.

[31] J.R.M Frejo and E.F. Camacho. Global versus local MPC algorithms in freeway traffic control with ramp metering and variable speed limits. *IEEE Transactions on Intelligent Transportation Systems*, 13(4):1556–1565, 2012.

[32] D. Frigioni, A. Marchetti-Spaccamela, and U. Nanni. Fully dynamic algorithms for maintaining shortest paths trees. *Journal of Algorithms*, 34:251–281, 2000.

[33] A.V. Goldberg and C. Harrelson. Computing the shortest path: A* meets graph theory. *ACM-SIAM Symposium on Discrete Algorithms*, 16:156–165, 2005.

[34] N. Groot, B. De Schutter, and H. Hellendoorn. Integrated model predictive traffic and emission control using a piecewise-affine approach. *IEEE Transactions on Intelligent Transportation Systems*, 14(2):587–598, 2013.

[35] N. Groot, B. De Schutter, S. Zegeye, and H. Hellendoorn. Model-based predictive traffic control: A piecewise-affine approach based on METANET. In *Proceedings of the 18th IFAC World Congress*, pages 10709–10714, Milan, Italy, 2011.

[36] P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[37] S.M. Hashemi, S. Mokarami, and E. Nasrabadi. Dynamic shortest path problems with time-varying costs. *Optimization Letters*, 4:147–156, 2010.

[38] A. Hegyi. *Model Predictive Control for Integrating Traffic Control Measures.* PhD thesis, Delft University of Technology, 2004.

[39] A. Hegyi, B. De Schutter, and H. Hellendoorn. Model predictive control for optimal coordination of ramp metering and variable speed limits. *Transportation Research Part C*, 13(3):185–209, 2005.

[40] A. Hegyi, B. De Schutter, H. Hellendoorn, and T. van den Boom. Optimal coordination of ramp metering and variable speed control – an MPC approach. In *Proceedings of the 2002 American Control Conference*, pages 3600–3605, Anchorage, Alaska, 2002.

[41] M. Herceg, M. Kvasnica, C.N. Jones, and M. Morari. Multi-Parametric Toolbox 3.0. In *Proceedings of the European Control Conference*, pages 502–510, Zürich, Switzerland, Jul. 2013. `http://control.ee.ethz.ch/~mpt`.

[42] B.H. Heutinck, M. van den Berg, J. Hellendoorn, and L.H. Immers. Dynamic route guidance during maintenance works, a case study. *IFAC Proceedings Volumes*, 39(12):380–385, 2006.

[43] S.P. Hoogendoorn. Optimal control of dynamic route information panels. *IFAC Proceedings*, 30(8):399–404, 1997.

[44] S.P. Hoogendoorn and P.H.L. Bovy. State-of-the-art of vehicular traffic flow modelling. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 215(4):283–303, 2001.

[45] M. Hovd. Multi-level programming for designing penalty functions for MPC controllers. *IFAC Proceedings Volumes*, 44(1):6098–6103, 2011.

[46] A. Jamshidnejad. *Efficient Predictive Model-Based and Fuzzy Control for Green Urban Mobility.* PhD thesis, Delft University of Technology, 2017.

[47] R.G. Jeroslow. The polynomial hierarchy and a simple model for competitive analysis. *Mathematical programming*, 32(2):146–164, 1985.

[48] C.N. Jones and M. Morari. Approximate explicit MPC using bilevel optimization. In *2009 European control conference (ECC)*, pages 2396–2401, 2009.

[49] A. Karimi, A. Hegyi, B. De Schutter, H. Hellendoorn, and F. Middelham. Integration of dynamic route guidance and freeway ramp metering using model predictive control. In *Proceedings of the 2004 American Control Conference*, pages 5533–5538, Boston, Massachusetts, June-July 2004.

[50] H.R. Kashani and G.N. Saridis. Intelligent control for urban traffic systems. *Automatica*, 19(2):191–197, 1983.

[51] M. Klein. A primal method for minimal cost flows with applications to the assignment and transportation problems. *Management Science*, 14(3):205–220, 1967.

[52] F.H. Knight. Some fallacies in the interpretation of social cost. *The Quarterly Journal of Economics*, 38(4):582–606, 1924.

[53] A. Kotsialos, M. Papageorgiou, C. Diakaki, Y. Pavlis, and F. Middelham. Traffic flow modeling of large-scale motorway networks using the macroscopic modeling tool METANET. *IEEE Transactions on Intelligent Transportation Systems*, 3(4):282–292, 2002.

[54] A. Kotsialos, M. Papageorgiou, and A. Messmer. Optimal coordinated and integrated motorway network traffic control. In *Proceedings of the 14th International Symposium of Transportation and Traffic Theory (ISTTT)*, pages 621–644, Jerusalem, Israel, Jul. 1999.

[55] H.T. Lau. A note on the shortest path through a given link in a network. *IEEE Transactions on Circuits and Systems*, 33(1):109–110, 1986.

[56] H.K. Lo, E. Chang, and Y.C. Chang. Dynamic network traffic control. *Transportation Research Part A*, 33(8):721–744, 2001.

[57] X.Y. Lu and S. Shladover. MPC-based variable speed limit and its impact on traffic with V2I type ACC. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 3923–3928, Maui, HI, USA, 2018.

[58] X.Y. Lu, P. Varaiya, R. Horowitz, D. Su, and S.E. Shladover. Novel freeway traffic control with variable speed limit and coordinated ramp metering. *Transportation Research Record: Journal of the Transportation Research Board*, 2229:55–65, 2011.

[59] R. Luo, T. J. van den Boom, and B. De Schutter. Multi-agent dynamic routing of a fleet of cybercars. *IEEE Transactions on Intelligent Transportation Systems*, 19(5):1340–1352, 2017.

[60] A. Messner and M. Papageorgiou. METANET: A macroscopic simulation program for motorway networks. *Traffic Engineering & Control*, 31(8-9):466–470, 1990.

[61] A. Orda and R. Rom. Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length. *Journal of the Association for Computing Machinery*, 37(3):607–625, 1990.

[62] A. Orda and R. Rom. Minimum weight paths in time-dependent networks. *Networks*, 21:295–319, 1991.

[63] G. Ramalingam. *Bounded Incremental Computation*, volume 1089 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1971.

[64] G. Ramalingam and T. Reps. An incremental algorithm for a generalization of the shortest paths problem. *Journal of Algorithms*, 21:267–305, 1996.

[65] G. Ramalingam and T. Reps. On the computational complexity of dynamic graph problems. *Theoretical Computer Science*, 158:233–277, 1996.

[66] J. Richalet, A. Rault, J.L. Testud, and J. Papon. Model predictive heuristic control: Applications to industrial processes. *Automatica*, 14:413–428, 1978.

[67] M. Sakawa, I. Nishizaki, and Y. Uemura. Interactive fuzzy programming for multilevel linear programming problems. *Computers & Mathematics with Applications*, 36(2):71–86, 1998.

[68] E.D. Sontag. Nonlinear regulation: The piecewise linear approach. *IEEE Transactions on Automatic Control*, 26(2):346–357, 1981.

[69] J.M. Thomson. Reflections on the economics of traffic congestion. *Journal of Transport Economics and Policy*, 32(1):93–112, 1997.

[70] M. van den Berg, B. De Schutter, J. Hellendoorn, and A. Hegyi. Influencing route choice in traffic networks: A model predictive control approach based on mixed-integer linear programming. In *Proceedings of the 17th IEEE International Conference on Control Applications*, pages 299–304, San Antonio, TX, USA, 2008.

[71] J.G. Wardrop. Some theoretical aspects of road traffic research. *Proceedings of the Institution of Civil Engineers*, 1(3):325–362, 1952.

[72] U.P. Wen and S.T. Hsu. Linear bi-level programming problems—a review. *Journal of the Operational Research Society*, 42(2):125–133, 1991.

[73] H. Williams. *Model Building in Mathematical Programming.* Wiley, New York, NY, USA, 3^{rd} edition, 2013.

[74] C.J. Wu, X.Y. Lu, J. Spring, and R. Horowitz. Field test implementation of coordinated ramp metering control strategy: a case study on SR-99N. In *Proceedings of the Transportation Research Board Meeting 96th Annual Meeting*, 2018.

[75] S. Zegeye, B. De Schutter, H. Hellendoorn, and E. Breunesse. Model-based traffic control for balanced reduction of fuel consumption, emissions, and travel time. In *Proceedings of the 12th IFAC Symposium on Transportation Systems*, pages 149–154, Redondo Beach, CA, USA, 2009.

Glossary

List of Acronyms

DRIP	Dynamic Route Information Panel
LP	Linear Programming
METANET	Modèle d'Ecoulement du Trafic Autoroutier: NETwork
MILP	Mixed Integer Linear Programming
MLD	Mixed Logical Dynamical
MPC	Model Predictive Control
PWA	Piecewise-Affine
TTS	Total Time Spent

List of Symbols

a_m	METANET model parameter
C_b	Outflow capacity (veh/h) of bridge b
C_o	Outflow capacity (veh/h) of origin o
d_o	Traffic demand (veh/h) at origin o
$F_{\nu,m}$	Split (veh/h) of traffic leaving vertex ν for link m
G	Graph
J_{TTS}	Total time spent (veh·h)
L_m	Length (km) of segments in link m
m	Link
$q_{m,i}$	Traffic flow (veh/h) on segment i of link m
$q_{\mathrm{in},b}$	Potential traffic inflow (veh/h) from segment i_{u} to the queue at bridge b
$q_{u,b}$	Traffic inflow (veh/h) from segment i_{u} to the queue at bridge b
$q_{\mathrm{out},b}$	Potential traffic outflow (veh/h) from the queue at bridge b to segment i_{d}
$q_{w,b}$	Traffic outflow (veh/h) from the queue at bridge b to segment i_{d}
q_o	Traffic outflow (veh/h) at origin o
Q_ν	Summed traffic outflow at vertex ν (veh/h)
r_o	Ramp metering rate at origin o
T_{s}	Sample time (s)
$v_{m,i}$	Traffic speed (km/h) on segment i of link m
v_{free}	Free-flow speed (km/h)
$V_{i,m}$	Desired speed (km/h) on segment i of link m
w_b	Queue (veh) at bridge b
w_o	Queue (veh) at origin o
$\beta_{\nu,m}$	Percentage of traffic that leaves vertex ν for link m
η	METANET model parameter (km^2/h)
κ	METANET model parameter (veh/km/lane)
λ_m	Number of lanes on link m
ν	Vertex
$\rho_{m,i}$	Traffic density (veh/km) on segment i of link m
$\rho_{\mathrm{cr},m}$	Critical traffic density (veh/km) on link m
ρ_{jam}	Maximum traffic density (veh/km)
τ	METANET model parameter (s)

Index

A
Adjacency, 6

B
Bilevel Programming, 42
Braess's paradox, 2

D
Directionality, 5
Downstream Density, 11
Dynamic Network, 6

L
Least-Squares Problem, 11

M
METANET, 9
Minimum-Cost Flow Problem, 7, 63
Model Predictive Control, 15
Multi-Commodity Flow Problem, 7, 64
Multilevel Programming, 42

N
Node-Node Adjacency Matric, 7

O
Origin-Destination Pair, 3, 64

S
Selfish Wardrop Equilibrium, 7, 15
Shortest Path Problem, 6
Social Wardrop Equilibrium, 7, 15
Static Network, 6
Store-and-Forward Bridge, 28

U
Upstream Velocity, 11

Z
Zero-length Bridge, 27