Abstract

In the 21st century, urban road congestion has become a significant issue. This issue is worsened by urbanization and population growth. Traditional solutions such as expanding infrastructure can be counterproductive due to the effect of induced demand and the resource requirements associated with the expansion. Hence, when aiming to reduce the traffic-related climate impact, these approaches are less viable, motivating the search for alternate approaches.

This thesis proposes a method of optimizing traffic management by making better use of the existing infrastructure. This is done by determining optimal speed limits and traffic light settings, referred to as control parameters. To this end, a flexible mathematical method that models the densities of vehicles in a traffic-flow network is introduced. The flexibility of this method, allows for a prioritization of different network components. It is, for instance, possible to reduce the delays of buses, enhancing public transport efficiency, or to instead improve the general flow of traffic.

A macroscopic traffic model, treating traffic flow as a continuous "fluid", is developed for a unidirectional road. The model is then extended to realistic road networks, including traffic merging and yielding rules, as well as traffic lights and roundabouts. Different traffic-flow networks are considered, and locally optimal control parameters are determined using gradient information obtained through automatic differentiation.

Sammendrag

I det 21. århundres byer har kødannelse blitt et betydelig problem. Dette problemet forverres av økt urbanisering og befolkningsvekst. En tradisjonell løsning som baserer seg på å ekspandere infrastrukturen kan virke mot sin hensikt grunnet indusert etterspørsel og ressursbruken knyttet til utvidelsen av veinettverekt. Dersom målet er å redusere trafikkrelatert klimabelastning, er denne løsning altså ikke ideell.

I denne masteroppgaven blir en alternativ metode for å forbedre bruken av den eksisterende infrastrukturen foreslått. Denne metoden går ut på å velge optimale fartsgrenser og bestemme beste instillinger av trafikklys. Vi foreslår en fleksibel matematisk metode for å modellere tettheten av kjøretøy i et veinettverk. Fleksibiliteten til modellen muliggjør prioritering av ulike deler av veinettverket. Det er for eksemple mulig å redusere forsinkelse av kollektiv transport som busser, eller å forbedre den generelle flyten av trafikk.

En makroskopisk modell som behandler trafikkflyt som en kontinuerlig "væske" introduseres for en enveiskjørt vei, før modellen blir utvidet til å omhandle mer avanserte komponenter som kryss, trafikklys og rundkjøringer. I tilleg ser vi på hvordan en kan ta hensyn til vikepliktsregler og fletting av trafikk. Vi undersøker flere ulike veinettverk, og forsøker å redusere forsinkelse av kollektiv transport. Til dette formål bestemmes lokalt optimale fartsgrenser og innstillenger av trafikklys gjennom en gradient-basert optimeringsmetode, hvor gradientinformasjonen er beregnet ved hjelp av automatisk differensiering.

Preface

This thesis concludes my five year education to complete my Master of Science in Physics and Mathematics at the Norwegian University of Science and Technology, NTNU. The last three years, I have undergone a specialization in Industrial Mathematics, and in particular in the field of numerical mathematics. This thesis is a collaboration between NTNU and the independent research organization SINTEF.

I would like to thank my supervisors at SINTEF, Franz Georg Fuchs, Knut-Andreas Lie and Kjetil Olsen Lye, for their invaluable advice and close guidance. Without all of their feedback and help in guiding my focus, the completion of this thesis would not have been possible. Lastly, I would like to thank my family and friends not only for supporting me through the work on this thesis, but for the support throughout my five years of study.

Contents

Figures

Tables

List of Algorithms

Chapter 1

Introduction

In the modern cities of the 21st century, road congestion is a common problem. In larger cities, drivers commuting to and from work enhances the congestion, which leads to both increased fuel consumption and longer travel times. Urbanization and population growth cause cities to continue expanding, which enhances the congestion problem. The congestion problems are typically due to inadequate public transport, poor traffic management or insufficient road infrastructure. Several approaches are being tried to amend these issues. One natural approach to address the congestion problem is to assume that the road infrastructure is insufficient, and hence try to enhance the existing infrastructure. This can be done by, e.g., increasing the number of lanes on each road, or by simply building more roads. Although one would think that enhancing the infrastructure would help alleviate the congestion problems, this might not be the case in the long run. This is due to the effect of induced demand [1]. In addition, enhancing the infrastructure will in itself require a lot of resources, and so if the goal is to reduce the climate toll from a traffic network, enhancing the infrastructure may not be the way to go.

An alternate approach is to improve the management of traffic by making better use of the existing infrastructure. This can be done by setting the best possible speed limits, and finding the best settings of the traffic lights in the traffic network being considered. We will throughout this thesis refer to the speed limits of the roads and the settings of the traffic lights as the control parameters of a traffic-flow network. While finding a good choice of control parameters may seem like a daunting challenge, we will in this thesis propose a mathematical method for determining the best control parameters of a traffic-flow network. As we shall see, this mathematical approach is flexible, thus allowing for a prioritization of different parts of the traffic-flow network. As stated, one of the causes for road congestion can be inadequate public transport. Therefore, one of the main focuses of this thesis will be to look for the best control parameters minimizing the delays of buses.

The first part of the mathematical approach is to derive an accurate traffic model in order to approximate the traffic behaviour in a traffic-flow network.

Extensive work has been done in traffic modelling, and this work is typically divided into two regimes: microscopic and macroscopic modelling. A microscopic traffic model aims to model each individual car driving through a traffic-flow network. When considering a network with heavy traffic, the number of cars is very high. Hence, a microscopic model will typically be very computationally expensive. A macroscopic traffic model does not consider individual cars, but rather models the density of cars on each road as a continuous "fluid". We will in Chapter 2 discuss and derive a continuum model that aims to approximate the behaviour of traffic in a traffic-flow network.

The second part of the mathematical approach is to make use of the macroscopic model derived in Chapter 2 to determine at least locally optimal control parameters. There are many approaching that can be used to solve this optimal control problem. We will in this thesis make use of gradient information when searching for good choices for the control parameters. We will discuss how automatic differentiation (AD) can be used to compute the necessary gradients in Chapter 3.

The macroscopic models discussed in Chapter 2 are only relevant to unidirectional roads. Hence, we will in Chapter 4 extend them to include more components of real road networks. In particular, we will introduce methods for handling merging of traffic and yielding rules that to best of our knowledge have not been considered previously. As we have briefly discussed, the modelling approach is flexible, allowing us to choose what parts of the traffic-flow network we want to prioritize. We will in Chapter 5 introduce some ways of determining what constitutes a good choice of control parameters, in addition to describing methods for finding said parameters. In Chapter 6 we will discuss how the code used for simulating traffic in different road networks was implemented. We also discuss how we implement the algorithm for finding a good choice of control parameters. In Chapter 7, we introduce some concrete traffic-flow networks and use these how one can use mathematical optimization to search for optimal control parameters. Finally, we summarize our findings in Chapter 8.

Relationship to UN Sustainable Development Goals

The objective of this thesis can be related to UN's sustainability goals [2]. One of these goals is "*11: Make cities and human settlements inclusive, safe, resilient and sustainable*". In, particular making cities sustainable closely relates to the objectives of this thesis. By aiming to find a choice of control parameters that reduces the delay of buses, public transport will become a more attractive option. This might in turn motivate more people to make the switch to using public transport, thereby reducing fuel consumption. This reduced fuel consumption and the resulting reduction in pollution makes the city more sustainable.

Marking Conventions for Content Incorporated from Previous Work

This thesis builds directly on the foundation laid in my previous semester project [3]. Since NTNU does not publish reports from such semester projects openly, a copy of the original report has been uploaded to GitHub for reference. However, to ensure the completeness and coherence of the presentation, certain sections of the original project report have been incorporated herein, either verbatim or with modifications. This integration has been done in accordance with NTNU's policies to provide a comprehensive and self-contained narrative of the work undertaken.

- Chapters or sections that contain ideas, concepts, and/or discussions from the project thesis that herein are presented in revised form are marked with an asterisk ($*$).
- Chapters and sections that are more or less verbatim copies of content from the project thesis are marked with a double asterisk ($**$)

If a chapter is marked with either a single or a double asterisk, all sections and subsections within that chapter are subject to the same marking, even if not explicitly indicated. The same rule applies to sections and their subsections.

Source code

The computer code used to perform the numerical experiments reported in this thesis was developed from scratch. To support the important principle of reproducible research, a complete set of source codes can be freely downloaded from GitHub: `github.com/TorjeiHelset/master_project`. In addition, animations showing the results of some of the numerical experiments we consider in Chapter 7 can be found at `torjeihelset.github.io/master_project`.

Chapter 2

Basic Models and Discretization*

We will in this chapter discuss some basic macroscopic traffic models. We will also briefly discuss how these models can be discretized and solved numerically. The material presented in this chapter is a subset of a more comprehensive discussion found in the report from a proceeding semester project [3].

2.1 Models

As described in the introduction, one of the regimes of traffic-flow modelling approximates the density of cars as a continuous "fluid". This section will start by deriving some first-order models, as well as one second-order model. These are all examples of first-order, quasilinear hyperbolic conservation laws, which admit discontinuous solutions. We therefore end the section by introducing the concept of weak solutions, which is needed to formally define the solutions of these models, as well as some conditions introduced to ensure that the weak solution are unique.

2.1.1 First-Order Models*

Before looking at more complicated traffic models, we will start with the simplest case, the first-order models. The most famous first-order model is the LWR model, which we will derive in the subsequent subsection.

The LWR Model

The first macroscopic model for traffic flow was introduced by Lighthill and Whitham [4, 5] and Richards [6]. The underlying idea was to replace individual cars with a continuous density and use a simple conservation law, and in this way, a model equation can be derived. The derivation procedure will be explained in detail for a simple case.

We consider a single-lane road of length L without intersections. As mentioned, we replace the discrete collection of vehicles with a continuous medium. To this end, we introduce the density of cars $\tilde{\rho}(\tilde{x},\tilde{t})$. We will use the definition for the density at point x and time t that was used in [4]. The idea is to draw two lines across the road a short distance apart with the point x in the middle of the two lines. We denote this short distance by dx. Then consider also a time interval of moderate length τ centered around the time t. Let the number of cars crossing the two lines in the time interval by n, and let the time car i used to cross the interval between the two lines be denoted by dt_i. Then, the concentration $\tilde{\rho}(\tilde{x},\tilde{t})$ at the point (x,t) is given by

$$\tilde{\rho}(\tilde{x},\tilde{t}) = \frac{1}{\tau}\sum_{i=1}^{n} dt_i.$$

We assume that $\tilde{\rho}$ is bounded as

$$0 \leq \tilde{\rho} \leq \rho_{\max},$$

where $\rho_{\max}$ denotes the maximum density. We further let $v_{\max}$ denote the speed limit on the single-lane road. We assume that the speed of traffic only depends on the density of traffic and the speed limit of the road, i.e., we write the speed as

$$\tilde{v}(\tilde{\rho},\tilde{x},\tilde{t};v_{\max}) = \tilde{v}\big(\tilde{\rho}(\tilde{x},\tilde{t});v_{\max}\big).$$

We can now introduce the flux of cars, i.e., the number of cars passing a point per time unit, as

$$\tilde{f}(\tilde{x},\tilde{t}) = \tilde{\rho}(\tilde{x},\tilde{t})\,\tilde{v}\big(\tilde{\rho}(\tilde{x},\tilde{t};v_{\max})\big).$$

Again, following [4], we give a definition of what this flux represents. We recall the idea of drawing two lines a distance dx apart and centered around the point x, and considering a time interval τ centered around the point t. Letting n again denote the number of cars crossing the lines during the time interval, the flux $\tilde{f}(\tilde{x},\tilde{t})$ is given by

$$\tilde{f}(\tilde{x},\tilde{t}) = \frac{n}{\tau}.$$

We will now derive a model describing the density $\tilde{\rho}(\tilde{x},\tilde{t}) \in [0,\rho_{\max}]$ of cars at position $\tilde{x} \in [0,L]$ at time $\tilde{t} \in \mathbb{R}^+$. To this end, we consider the sub-interval of the road $(a,b) \subset [0,L]$. On any such sub-interval, the change in mass is equal to the flow entering the system minus the flow exiting the system. In integral form, the conservation law can be written as

$$\frac{d}{d\tilde{t}}\int_a^b \tilde{\rho}(\tilde{x},\tilde{t})\,d\tilde{x} = -\big[\tilde{f}(b,\tilde{t}) - \tilde{f}(a,\tilde{t})\big]. \tag{2.1}$$

Applying the fundamental theorem of calculus, we get the differential form

$$\tilde{\rho}(\tilde{x},\tilde{t})_{\tilde{t}} + \big(\tilde{\rho}(\tilde{x},\tilde{t})\tilde{v}(\tilde{\rho}(\tilde{x},\tilde{t});v_{\max})\big)_{\tilde{x}} = 0. \tag{2.2}$$

Omitting function parameters, we write the differential form as

$$\tilde{\rho}_{\tilde{t}} + \left(\tilde{\rho}\tilde{v}\right)_{\tilde{x}} = 0. \tag{2.3}$$

So far, we have said that the speed depends only on the density and the speed limit. We further assume that when the density of cars is low, the speed approaches the speed limit, and that as the density increases, the speed will decrease. A multitude of different models satisfying these assumptions have been used in prior works, see e.g. [7, 8] for a comparison of some of these models. Maybe the simplest one is a linear interpolation that leads to the simple relation

$$\tilde{v}(\tilde{\rho}; v_{\max}) = v_{\max}(1 - \tilde{\rho}). \tag{2.4}$$

So far, we have used the tilde notation ($\tilde{x}$) for all variables without explaining its meaning. This notation is used to signify that these variables are representative physical variables, relating directly to the road system being modelled. However, instead of using these variables directly in the simulation, we will introduce new *scaled* variables $x \in [0, 1]$, $\rho \in [0, 1]$ and $v \in [0, 1]$ that satisfy

$$\begin{cases} \tilde{x} = Lx, \\ \tilde{\rho} = \rho_{\max}\rho, \\ \tilde{v} = v_{\max}v. \end{cases} \tag{2.5}$$

We insert the new variables using relation (2.5) and obtain

$$\frac{\partial\tilde{\rho}}{\partial\tilde{t}} + \frac{\partial(\tilde{\rho}\tilde{v})}{\partial\tilde{x}} = \rho_{\max}\frac{\partial\rho}{\partial\tilde{t}} + \frac{\rho_{\max}v_{\max}}{L}\frac{\partial(\rho v)}{\partial x} = 0. \tag{2.6}$$

Thus, we can write a new model equation as

$$\rho_{\tilde{t}} + \gamma(\rho v)_x = 0, \tag{2.7}$$

where $\gamma = v_{\max}/L$.

The relation between the new speed and density variables using the linear interpolation model will now be

$$v(\rho) = 1 - \rho. \tag{2.8}$$

Even though we keep the physical variable $\tilde{t}$, we will drop the tilde notation in subsequent sections. With this, we rewrite the transport equation for the traffic flow as

$$\rho_t + \gamma(\rho v)_x = 0. \tag{2.9}$$

Time-delayed LWR model

The classical LWR model is a simplification of reality and fails to capture some phenomena observed in traffic. Goatin et al. [9] showed that the LWR model fails to qualitatively match experimental data, especially at higher densities.

To overcome some of the issues with the LWR model, several second-order models have been proposed. Before we look at some of these, we will look at one way of improving the first-order model by adding a time delay term in the flux function as introduced by Newell [10]. This delay time takes into account that velocities can not change instantaneously, which is one of the main issues not considered in the LWR model.

One such time-delayed model was derived from a microscopic model in [11]. Letting $T \geq 0$ be the delay and keeping the delay time explicit, the delayed LWR model can be formulated as

$$\rho(x,t)_t + \big[v\big(\rho(x,t-T)\big)\rho(x,t)\big]_x = 0. \tag{2.10}$$

In [12], some of the theoretical and numerical properties of this model were investigated. It was shown that this model conserves mass and positivity of density. However, one drawback is that it no longer guarantees a maximum density, meaning that $\rho > \rho_{\max}$ is possible.

Comparing this model to the LWR model, it was shown that for the right choice of parameters that the time-delayed model was able to reproduce *stop-and-go waves*, which cannot be reproduced by the LWR model. A stop-and-go wave occurs when a car in heavy traffic slows down slightly, which causes the car behind to slow down even more, and this slowing down propagates backwards while getting increasingly worse. This phenomena causes traffic jams to form without any external reason, like a traffic light or a car accident. This phenomena was analysed by, e.g., Laval and Leclercq [13] and Ros et al. [14].

2.1.2 Higher-Order Models**

With the goal of addressing some of the known drawbacks of the first-order LWR model, we will now look at a second-order macroscopic model. Some models that attempted to correct the earliest prototypes of a second-order model were introduced by Aw and Rascle [15] and Colombo [16]. Following the work done in [9], we will now look at how the Aw–Rascle model can be combined with the LWR model.

Aw–Rascle Coupled with LWR

Even though the Aw–Rascle model [15] is more suitable than LWR in some cases, it also has some drawbacks. Goatin et al. [9] attempted to fix some of these drawbacks by combining it with the LWR model. One major drawback is that the maximal speed reached on an empty road depends on initial data, which is clearly wrong. Therefore, this model is not suitable for lower densities. Since the two models appear to have their separate domains where they are suitable, the idea is to use the LWR for lower densities and the Aw–Rascle model for higher densities. The two density regimes can be described as free flow and congested flow. Let Ω_f denote the free-flow regime, and let Ω_c denote

the congested-flow regime. The full model can then be given as

$$\begin{cases} \rho_t + (\rho v)_x = 0, & v = v(\rho), & \text{if } (\rho, v) \in \Omega_f \\ \rho_t + (\rho v)_x = 0, & y_t + (yv)_x = 0, & \text{if } (\rho, v) \in \Omega_c \end{cases} \tag{2.11}$$

where $y = \rho\big(v + p(\rho)\big)$ is a generalized moment of cars and $p(\rho) = V_{\text{ref}} \ln(\rho / V_{\max})$ is a pressure function. (Can also be on a different form as long as some conditions are met.) The constant $V_{\max}$ is the maximal speed, and V_{ref} is some reference speed.

In Ω_c, we no longer assume the simple relation between ρ and v. The benefit of this model is that it fixes the issues with the Aw–Rascle model at lower densities, while at the same time giving results that fit experimental data better than the LWR model does. In future work, it would be interesting to use a more accurate model, such as this one, for modelling the flow of traffic. However, for the purposes of this work we will stick to the simple first-order LWR model.

2.1.3 Weak Solutions and Uniqueness**

In this section, we introduce the concept of weak solutions and discuss some criteria that are needed to obtain a unique solution to the LWR model (2.9) we introduced in Section 2.1.1. This model equation is a special case of a scalar, quasilinear, hyperbolic conservation law, which takes the form

$$\rho_t + f(\rho)_x = 0, \tag{2.12}$$

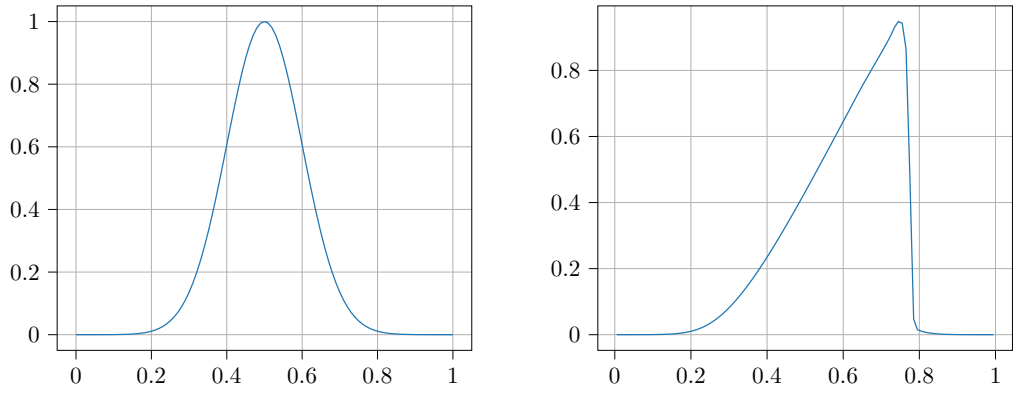
in general. If the flux function f in (2.12) is nonlinear, smooth or classical solutions may not exist. However, since these equations typically are derived from more fundamental integral equations that model physical phenomena using a limiting argument, some type of solution does exist. When a classical solution does not exist, we need to introduce the notion of a *weak solution*.

We investigate what is meant by a weak solution by consider a specific example. The simplest nonlinear equation on the form of (2.12) is know as the inviscid Burgers' equation, and is given as

$$\rho_t + \left(\frac{\rho^2}{2}\right)_x = 0. \tag{2.13}$$

This equation occurs in, e.g., fluid and gas dynamics, and will in some cases develop discontinuities in finite time, known as shocks, even if the initial condition is smooth. In addition, rarefaction waves may form in finite time. The name rarefaction wave comes from the fact that the fluid becomes more rarefied when the density decreases. Figure 2.1 shows that both a rarefaction and a shock wave can form in finite time from smooth initial data.

We will now give a precise description of what is meant by weak solution. Suppose for the moment that (2.12) has a classical solution. Let $\phi \in C_c^1(\mathbb{R} \times$



(a) Initial data $\rho_0(x)$.

(b) Solution $\rho(x,t)$ at time $t = 0.3$.

Figure 2.1: The figure shows the approximate solution of Burgers' equation with initial data $\rho_0(x) = e^{-5(x-\frac{1}{2})^2}$.

$\mathbb{R}^+$), where $C_c^1(\mathbb{R} \times \mathbb{R}^+)$ is the space of all functions that are continuously differentiable with compact support. If a classical solution ρ of (2.12) exists, the equation must also hold if we multiply by the smooth test function ϕ and integrate over the domain. That is, the equation

$$\int_{\mathbb{R}\times\mathbb{R}^+} \big(\rho_t\phi + f(\rho)_x\big)\phi \, dxdt = 0, \tag{2.14}$$

must also hold true. We apply integration by parts to (2.14) and obtain

$$\int_{\mathbb{R}\times\mathbb{R}^+} \big(\rho\phi_t + f(\rho)\phi_x\big) \, dxdt + \int_{\mathbb{R}} \rho_0(x)\phi(x,0) \, dx = 0. \tag{2.15}$$

Since ϕ was chosen arbitrarily from the space $C_c^1(\mathbb{R} \times \mathbb{R}^+)$, the identity (2.15) holds true for all test functions ϕ and will be used to define what is meant by a weak solution. Following [17] we have the following definition of a weak solution.

Definition 2.1.1 (Weak solution)**.** *A function $\rho \in L^\infty(\mathbb{R} \times \mathbb{R}^+)$ is a weak solution of* (2.12) *with initial data $\rho_0 \in L^\infty$ if the identity* (2.15) *holds for all test functions $\phi \in C_c^1(\mathbb{R} \times \mathbb{R}^+)$.*

We can show that if a weak solution ρ of (2.12) is also differentiable, it will satisfy (2.12) point-wise. This means that the class of weak solutions will contain all classical solutions, or in other words, all classical solutions are also weak solutions.

We observe that the definition of a weak solution ρ does not require ρ to be differentiable or even continuous. This means that the solution can contain discontinuities, or shock waves. In general, the weak solution to a conservation

law is not unique. Hence, to regain a unique solution, some extra conditions are needed.

To get an idea of what conditions must be satisfied by shock waves, we consider the case where the weak solution ρ consists of two smooth regions with a shock wave separating them, and investigate which conditions can be assumed on such a solution. As before we let $\phi \in C_c^1(\mathbb{R}\times\mathbb{R}^+)$ be a test function. We assume ϕ has support in Ω for some open set Ω. We define the two regions where the weak solution is smooth as Ω^+ and Ω^-. Since the weak solution is smooth on the two regions, we have $\rho \in C^1(\Omega^+)$ and $\rho \in C^1(\Omega^-)$. Since ρ is a weak solution, it will satisfy the identity (2.15). Using the compact support of ϕ, integration by parts and splitting Ω into the two regions we rewrite (2.15) as

$$\begin{aligned}\int_\Omega \big(\rho\phi_t + f(\rho)\rho_x\big)\,d\Omega &= \int_{\Omega^+} \big(\rho\phi_t + f(\rho)\rho_x\big)\,d\Omega + \int_{\Omega^-} \big(\rho\phi_t + f(\rho)\rho_x\big)\,d\Omega \\ &= -\int_{\Omega^+} (\rho_t + f(U)_x)\phi\,d\Omega + \int_{\partial\Omega^+} \big(\rho^+(t)\nu^t + f(\rho^+(t))\nu^x\big)\rho\,d\Omega \\ &\quad - \int_{\Omega^-} \big(\rho_t + f(U)_x\big)\phi\,d\Omega + \int_{\partial\Omega^-} \big(\rho^-(t)\nu^t + f(\rho^-(t))\nu^x\big)\rho\,d\Omega = 0,\end{aligned}$$

where $\rho^+(t)$ and $\rho^-(t)$ are the trace values of ρ on the respective sides of the discontinuity and ν is the unit outward normal of the shock wave.

For the unit normal of the shock wave $\sigma(t)$, we have

$$(\nu^t, \nu^x) = (-s(t), 1),$$

where $s(t) = \sigma'(t)$ is the speed of the shock curve.

We assumed the weak solution to be smooth in Ω^+ and Ω^-, and so (2.12) is satisfied point-wise. This implies that

$$\int_{\Omega^+\cup\Omega^-} \big(\rho_t + (f(\rho))_x\big)\phi\,d\Omega = 0.$$

Inserting this into the above identity gives

$$\int_{\partial\Omega} \big[s(t)\big(\rho^+(t) - \rho^-(t)\big) - \big(f(\rho^+(t)) - f(\rho^-(t))\big)\big]\phi\,d\Omega = 0.$$

Since ϕ can be chosen arbitrarily, this implies that the integrand must be equal to zero. That is, the speed of the shock curve, $s(t)$, satisfies

$$s(t) = \frac{f(\rho^+(t)) - f(\rho^-(t))}{\rho^+(t) - \rho^-(t)}, \tag{2.16}$$

which is known as the *Rankine–Hugoniot* condition. This condition can be used to obtain a unique solution when a shock wave appears.

In this example, we only considered the formation of one shock wave, but as mentioned before, rarefaction waves may also start to form in finite time.

The Rankine–Hugoniot condition is not enough to uniquely determine a weak solution in this case.

Motivated by the second law of thermodynamics, which states that the entropy of a system must be non-decreasing with time, one must also impose some so-called *entropy conditions* to regain uniqueness [17]. The first entropy condition, is:

Lax entropy condition: *For a convex scalar conservation law, a discontinuity propagating with speed s given by* (2.16) *satisfies the Lax entropy condition if*

$$f'(\rho^-) > s > f'(\rho^+). \tag{2.17}$$

A second entropy condition comes from the quantification of the rate of spreading of the characteristics.

Oleinik entropy condition: *$\rho(x,t)$ is the entropy solution to a scalar conservation law on the form* (2.12) *with $f''(\rho) > 0$ (convex) if there is a constant $E > 0$ such that for all $a > 0$, $t > 0$, and $x \in \mathbb{R}$,*

$$\frac{\rho(x+a,t) - \rho(x,t)}{a} < \frac{E}{t}. \tag{2.18}$$

As discussed, e.g., in[17], these conditions can be used to select the physically relevant solution, among the weak solutions satisfying (2.15).

2.2 Finite-Volume Methods*

The nonlinear transport equation (2.9) typically does not have an explicit formula for the solution. Therefore, the solution will be approximated numerically using a finite-volume method, as described in [17]. Since the transport equation (2.9) is nonlinear, solutions may contain shock waves as well as rarefactions. We therefore need the concept of weak solutions and a way to obtain a unique solution, as described in the previous section. One of the main reasons for choosing finite-volume methods as opposed to other numerical methods, is that the resulting solutions will converge to the unique entropy solution. This fact is guaranteed by the famous Lax–Wendroff theorem (see, e.g., [17]) under some assumptions on the scheme. We will also see that finite-volume methods are locally conservative so that they accurately preserve the integral form of the conservation law over each control volume. Since we are solving a conservation law, it is ideal to make use of a conservative scheme, as this will guarantee numerical conservation of conserved physical properties across the computational domain. We note that a finite-volume scheme effectively goes back the integral of the conservation (2.1) that we used to derive (2.9).

2.2.1 Basics of Finite-Volume Methods**

We introduce the ideas behind finite-volume methods. In our traffic-modelling setting, the discussion will be related to a unidirectional road with endpoints

at x_L and x_R. The first step of the finite-volume method is to discretize the spatio-temporal domain $[x_L, x_R] \times [0, T]$ in which we seek the solution. For simplicity, the spatial domain of the road $[x_L, x_R]$ is discretized uniformly with mesh size Δx. We also need some temporal discretization for the time interval $[0, T]$ for some terminal time T. Although the discretization in time could also have been chosen to be uniform, we will at any instance in time let the next time step be as large as possible to speed up the simulation process. The maximum value of this time step follows from the CFL condition (2.22) that we will come back to later.

Following a convention similar to that of [17], we divide the spatial domain $[x_L, x_R]$ into $N + 1$ segments of uniform length, where the length of each segment is defined as $\Delta x = \frac{x_R - x_L}{N+1}$. We denote the points at the edges of these segments as $x_{j-\frac{1}{2}} = x_L + j\Delta x$ for $j = 0, \ldots, N + 1$. The left and right edges of segment j will be $x_{j-\frac{1}{2}}$ and $x_{j+\frac{1}{2}}$, respectively. We will refer to each of the segments as computational cells, or *control volumes*, where each control volume is defined as

$$C_j = [x_{j-\frac{1}{2}}, x_{j+\frac{1}{2}}), \text{ for } j = 0, ..., N.$$

The finite-volume method makes uses of the control volumes, in contrast to the finite-difference method, which makes use of the grid points x_j directly. A visualization of a portion of such a grid is shown in Figure 2.2.

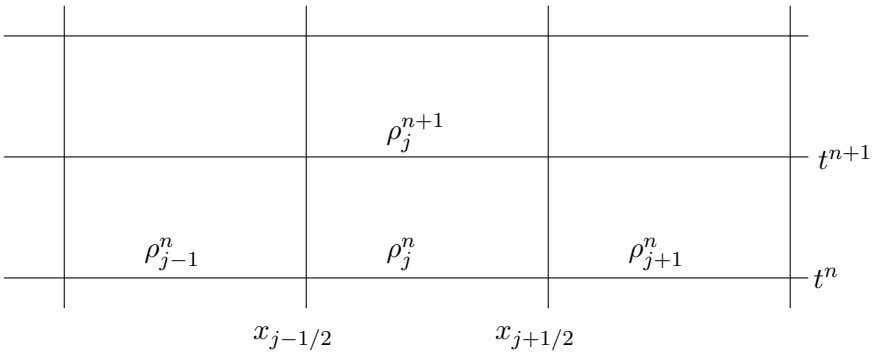


Figure 2.2: Example of grid that is used in a finite-volume method.

As we explained in Section 2.1, solutions of conservation laws will not be continuous in general, and point evaluations of the solution might thus not make sense. Therefore, a finite-difference method that aims to approximate point values of the solution may not be very well suited. The finite-volume method considers instead the average of the solution on a control volume. At each time level t^n, this cell average is approximated as

$$\rho_j^n \approx \frac{1}{\Delta x} \int_{x_{j-1/2}}^{x_j+1/2} \rho(x, t^n)\, dx.$$

The benefit of this change in perspective is that the cell averages are defined

for any integrable functions, and therefore will be defined for solutions to the conservation law.

The finite-volume method is a sequential procedure that aims to update the cell average at every step, using the approximation at the previous step. The initial cell averages are calculated as

$$\rho_j^0 = \frac{1}{\Delta x}\int_{x_{j-1/2}}^{x_j+1/2} \rho_0(x)\,dx,$$

where $\rho_0(x)$ defines the initial value of the solution.

Assuming that the cell averages ρ_j^n at time level t^n are known, we need some method to obtain the cell averages at the next time level t^{n+1}. To this end, we integrate the conservation law over the cell and in time. In this way, the cell average at the next time level can be computed. The integral equation reads

$$\int_{t^n}^{t^{n+1}}\int_{x_{j-1/2}}^{x_{j+1/2}} \rho_t\,dxdt + \gamma\int_{t^n}^{t^{n+1}}\int_{x_{j-1/2}}^{x_{j+1/2}} f(\rho)_x\,dxdt = 0.$$

We now apply the fundamental theorem of calculus to simplify the equation, giving

$$\int_{x_{j-1/2}}^{x_{j+1/2}} \rho(x,t^{n+1})\,dx - \int_{x_{j-1/2}}^{x_{j+1/2}} \rho(x,t^n)\,dx = \\ -\gamma\int_{t^n}^{t^{n+1}} f(\rho(x_{j+1/2},t))\,dt + \gamma\int_{t^n}^{t^{n+1}} f(\rho(x_{j-1/2},t))\,dt. \quad (2.19)$$

For shorter notation, we define the time average of the flux over the cell as

$$\bar{F}_{j+1/2}^n = \frac{1}{\Delta t}\int_{t^n}^{t^{n+1}} f(\rho(x_{j+1/2},t))dt, \quad (2.20)$$

and rewrite (2.19) as

$$\rho_j^{n+1} = \rho_j^n - \frac{\gamma\Delta t}{\Delta x}\big(\bar{F}_{j+1/2}^n - \bar{F}_{j-1/2}^n\big), \quad (2.21)$$

where again ρ_j^n denotes the cell average of the density.

From (2.21), it is apparent that as long as the fluxes $\bar{F}_{j+1/2}^n$ and $\bar{F}_{j-1/2}^n$ are known, the cell averages of the densities can be calculated. What remains is to establish some way of approximating these fluxes. The following sections will look into some different approximation methods.

We see from (2.19) that the flux approximations $\bar{F}_{j+1/2}^n$ and $\bar{F}_{j-1/2}^n$ appear when updating the density approximation on cell j. When updating the neighbouring cells $j+1$ and $j-1$, these will fluxes will again appear, but this time with opposite signs. Hence, when summing over all cells of the discretized domain, the fluxes across each of the interfaces will cancel out. This again means that the scheme will be conservative.

2.2.2 Godunov's Method**

With the goal of approximating the fluxes (2.20) in (2.21), Godunov realized that since the cell averages ρ_j^n are constant, each interface between two consecutive cells defines a so-called Riemann problem. This can be written as

$$\rho_t + \gamma f(\rho)_x = 0, \qquad \rho(x, t^n) = \begin{cases} \rho_j^n, & \text{if } x < x_{j+1/2}, \\ \rho_{j+1}^n, & \text{if } x > x_{j+1/2}. \end{cases}$$

In its original form, the Godunov method relies on solving this Riemann problem analytically, and then inserting this solution into (2.20) to compute the exact numerical flux.

The analytical solution can be found in some cases using the method of characteristics. The solution will consist of shock, rarefaction and compound waves, each with associated finite speeds of propagation. If the time step is chosen to be too large, waves from neighbouring interfaces will start to interact. To avoid this effect, the length of each time step is bounded so that no interactions occur. For any Riemann problem, the maximum speed of propagation is bounded from above by

$$\max_j |f'(\rho_j^n)|.$$

Thus, to ensure that waves from neighbouring cell interfaces do not interact, it is sufficient to impose the so-called Courant–Friedrichs–Lewy (CFL) condition

$$\nu^n = \max_j |f'(\rho_j^n)| \frac{\Delta t}{\Delta x} \leq \frac{1}{2}. \tag{2.22}$$

This condition will be used when choosing the timestep Δt.

2.2.3 A High-Resolution Scheme

In the report from the proceeding semester project [3], we gave a detailed derivation and discussion of different finite-volume schema and also analysed the properties of some of these schemes.

While the Godunov scheme presented in the previous subsection has many appealing properties, it is often not possible to compute the underlying exact Riemann solutions. Instead, one often resorts to approximate Reimann solutions. One example how one can approximate the intercell fluxes in this way is the Rusanov flux, given by

$$\begin{aligned} F_{j+1/2}^n &= F^{\text{Rus}}(\rho_j^n, \rho_{j+1}^n) \\ &= \frac{f(\rho_j^n) + f(\rho_{j+1}^n)}{2} - \frac{\max(|f'(\rho_j^n)|, |f'(\rho_{j+1}^n)|)}{2}(\rho_{j+1}^n - \rho_j^n), \end{aligned} \tag{2.23}$$

which we will use for all the numerical experiments presented later in this thesis.

Substituting (2.23) into (2.21) gives us a fully discrete scheme that may be used to advance the solution a time step Δt^n from time t^n to t^{n+1}. The length of the the time step is determined by first prescribing a targeted Courant number $\nu^* < \frac{1}{2}$, and then using (2.22) to calculate resulting Δt^n.

The scheme just presented is able to resolve discontinuous solutions without introducing spurious oscillations that are typical for classical finite-difference schemes. The drawback is that the scheme is only formally first-order and will thus not be very accurate in smooth parts of the solution. One way to get a scheme that is accurate in both discontinuous and continuous parts of the solution is to introduce a so-called high-resolution scheme. One particular version discussed in [3] can be written on the form

$$\begin{aligned} \boldsymbol{\rho}^* &= \boldsymbol{\rho}^n + \Delta t \mathcal{L}(\boldsymbol{\rho}^n), \\ \boldsymbol{\rho}^{**} &= \boldsymbol{\rho}^* + \Delta t \mathcal{L}(\boldsymbol{\rho}^*), \\ \boldsymbol{\rho}^{n+1} &= \frac{1}{2}(\boldsymbol{\rho}^n + \boldsymbol{\rho}^{**}), \end{aligned} \tag{2.24}$$

where

$$\mathcal{L}(\boldsymbol{\rho})_j := -\frac{\gamma}{\Delta x}(F^-_{j+1/2}(t) - F^+_{j-1/2}(t)).$$

We here assume that the numerical fluxes are approximated using (2.23), and that we are using a piecewise linear reconstruction of the densities on each cell. For details on the linear reconstruction and the high resolution scheme among others see [3].

Chapter 3

Automatic Differentiation

As we discussed in the introduction, the goal of this work is not only to create a realistic traffic model but also to enhance traffic flow through optimal control and configuration of features restricting vehicle movement that will be discussed in more detail in Chapter 4. This includes identifying the best methods for controlling traffic lights, selecting optimal speed limits for various parts of the road network, and other similar measures. To this end, many different approaches are possible, but to reduce the number of times a model simulation needs to be run to find the optimal control parameters, a gradient-based optimization method will we used. We will come back to the details of this optimization method in Chapter 5.

In this chapter, we will introduce the concept of automatic differentiation (AD), also known as algorithmic differentiation. We will describe the basic concepts of AD and look at how AD can be applied to the problem of trying to find optimal control parameters for a road network.

3.1 Why Use Automatic Differentiation?

Making use of a gradient-based optimization method introduces the new problem of having to evaluate the gradient of some objective function with respect to the control parameters. Some possible objective functions are discussed in Section 5.1. Evaluating such a gradient can be done in several different ways, each method being suitable in different scenarios. The most naive approach is to manually evaluate the derivatives by hand. The problem of this approach is that manual differentiation is both time consuming and error prone, which makes it unfeasible for large numerical simulations.

A different approach is symbolic differentiation, for which there exist different frameworks such as SymPy [18]. The drawback of symbolic differentiation is that as the size of the numerical simulation increases, the symbolic expression of the gradient might grow too large, leading to long running times. This effect is commonly referred to as expression swell, and makes symbolic differentiation seem ill-suited for our purposes. We show this effect with an example

from [19]. Suppose we want to calculate the gradient of the function

$$f(x) = \prod_{i=1}^{n} x_i = x_1 \cdots x_n.$$

The symbolic expression of the gradient takes the form

$$\nabla f(x) = \begin{pmatrix} x_2 \cdot x_3 \cdots x_n \\ x_1 \cdot x_3 \cdot x_4 \cdots x_n \\ \vdots \\ x_1 \cdots x_{i-1} \cdot x_{i+1} \cdots x_n \\ \vdots \\ x_1 \cdots x_{n-2} \cdot x_{n-1}) \end{pmatrix}$$

It is clear that even though the original function appears quite simple, the exact symbolic expression for the gradient might be much larger.

Instead of trying to compute the exact symbolic expression, an alternative approach is to approximate the derivatives. A common approach is to approximate partial derivatives using difference quotients, which are discussed in detail in, e.g., [20]. Three common difference quotients are the forward, backward and the central difference quotients, which can be written on the forms

$$\begin{aligned} f'(\bar{x}) &= \frac{f(\bar{x}+h) - f(\bar{x})}{h} + \mathcal{O}(h), \\ f'(\bar{x}) &= \frac{f(\bar{x}-h) - f(x')}{h} + \mathcal{O}(h), \\ f'(\bar{x}) &= \frac{f(\bar{x}+h) - f(\bar{x}-h)}{2h} + \mathcal{O}(h^2). \end{aligned} \tag{3.1}$$

We see that these difference quotients include a truncation error on the form $\mathcal{O}(h)$ or $\mathcal{O}(h^2)$. In addition, the computational cost grows with the number of parameters of the function we are evaluating, which might lead to slow gradient computations if the number of input variables is large.

Although all three differentiation approaches we have considered so far have their uses, some of their properties are not ideal if we want to compute accurate gradients of potentially large numerical programs. This motivates the search for a different differentiation approach, namely automatic differentiation or AD. AD has some overlap with symbolic differentiation, but differs in that instead of computing symbolic expressions of the derivatives, symbolic derivation rules are used to produce numerical evaluations of the derivatives. By producing numerical evaluations of the derivatives, instead of the symbolic expressions, AD eliminates the problem of expression swell. This technique has some overlap with numerical differentiation in that numerical values are produced instead of symbolic expressions. However, as opposed to numerical differentiation, AD does not incur truncation errors, and is accurate to machine precision.

3.2 The Basics of Automatic Differentiation

Automatic differentiation makes use of the fact that any results from a numerical simulation is the result of a finite number of primitive operations for which the derivative is known. These operations can either be functions such as $\sin(\cdot), \cos(\cdot), \exp(\cdot), \log(\cdot), \ldots$, simple arithmetic operations such as addition, subtraction, multiplication or division, or combinations thereof. In either case, the chain rule can be used to combine the derivatives of these primitive operations to form the derivative of the compound result of the numerical simulation with respect to predetermined variables or input parameter. Since each primitive derivative is computed exactly, the compound derivatives calculated using AD are exact to machine precision.

We visualize the idea of using the chain rule by a simple example. Consider the composite function

$$f(x) = (f_3 \circ f_2 \circ f_1)(x) = f_3(f_2(f_1(x))),$$

where we assume that the derivatives $f_1'(x)$, $f_2'(x)$ and $f_3'(x)$ are known closed-form expressions that can be evaluated exactly for any input x. Suppose now that we want to evaluate the derivative $f'(x)$ at the point $\bar{x}$. We can make use of the chain rule to calculate the partial derivative as

$$\begin{aligned} f'(\bar{x}) &= \big(f_3 \circ f_2 \circ f_1\big)'(\bar{x}) = f_3'\Big((f_2 \circ f_1)(\bar{x})\Big) \cdot \Big(\big(f_2 \circ f_1\big)'\Big)(\bar{x}) \\ &= f_3'\Big((f_2 \circ f_1)(\bar{x})\Big) \cdot f_2'\Big(f_1(\bar{x})\Big) \cdot f_1'(\bar{x}), \end{aligned} \tag{3.2}$$

which we know how to compute as long as we know the expressions for f_3, f_2, f_1 and the derivatives. Alternatively, we can write this in the simplified way by first defining some intermediate variables

$$w_0 = \bar{x}, \quad w_1 = f_1(w_0), \quad w_2 = f_2(w_1), \quad w_3 = f_3(w_2) = f(\bar{x}) = y. \tag{3.3}$$

Making use of these intermediate variables we rewrite (3.2) as

$$\frac{\partial y}{\partial x} = f'(\bar{x}) = f_3'(w_2) \cdot f_2'(w_1) \cdot f_1'(\bar{x}). \tag{3.4}$$

The chain rule is a crucial component of AD, as we shall see. To be able to use it, the exact composition of the function f needs to be known. Thus, tracing the composition of the set of elementary functions that together constitute, e.g., a numerical simulation is a key part of AD. The particular sequence all operations of a numerical function are evaluated in is often referred to as an *evaluation trace*, and may be visualized using a *computational graph*. The partial derivatives are calculated making use of the chain rule in a similar fashion to as in (3.4). Knowing the evaluation trace is crucial in making this work.

Table 3.1: Evaluation trace of the calculation of $y = f(x_1, x_2)$ with $f(x_1, x_2)$ given by (3.5).

v_{-1}	$=$	x_1	$=$	2.5000		
v_0	$=$	x_2	$=$	2.0000		
v_1	$=$	$\log(v_{-1})$	$=$	$\log(2.5000)$	$=$	0.3979
v_2	$=$	$v_1 \cdot v_0$	$=$	$0.3979 \cdot 2.0$	$=$	0.7958
v_3	$=$	$v_{-1} \cdot v_0$	$=$	$2.5000 \cdot 2.0$	$=$	5.0000
v_4	$=$	$\cos(v_3)$	$=$	$\cos(5.0000)$	$=$	0.2837
v_5	$=$	$v_2 + v_4$	$=$	$0.7958 + 0.2837$	$=$	1.0795
v_6	$=$	$\exp(v_0)$	$=$	$\exp(2.0000)$	$=$	7.3891
v_7	$=$	$v_3 - v_6$	$=$	$5.0000 - 7.3891$	$=$	-2.3891
v_8	$=$	$v_5 \cdot v_7$	$=$	$1.0795 \cdot (-2.3891)$	$=$	-2.5790
y	$=$	v_8	$=$	-2.5790		

There are two modes of AD, each referring to the direction the computational graph is traversed when evaluating the gradient. In the *forward mode* the computational graph is traversed from left to right, or inside out. That is, starting from the input and ending with the output. In the *backward mode*, the computational graph is traversed in the opposite direction, starting from the output and ending up at the input. Before looking at the two different modes, we will start by considering the evaluation trace and computational graph of a small example.

The following example, inspired by the example in chapter 1 from Griewank and Walther [21] gives an instructive introduction to the basic ideas of AD. Suppose that we are interested in calculating $y = f(x_1, x_2)$, with the function $f(x_1, x_2)$ given by

$$f(x_1, x_2) = \Big(\log(x_1) \cdot x_2 + \cos(x_1 \cdot x_2)\Big) \cdot \Big(x_1 \cdot x_2 - \exp(x_2)\Big). \tag{3.5}$$

Suppose that we have a computer program that calculates $y = f(x_1, x_2)$, starting from $x_1 = 2.5$ and $x_2 = 2.0$. Then, the sequence of operations listed in Table 3.1 may be executed by the program in the order listed in the table. We call this record of the operations the evaluation trace. For each new operation, we assign the value to a new variable, and the new variable may be used in subsequent operations. In this example, all numbers are represented using four decimal places. A visualization of the sequence of operations performed by the program can be seen by the computational graph visualized in Figure 3.1. We will now look at how partial derivatives of this small example can be computing using the two modes of AD.

3.2.1 Forward Mode

As mentioned, in the forward mode the derivatives are calculated starting from the input. Whenever a new operation is computed, we also compute the asso-

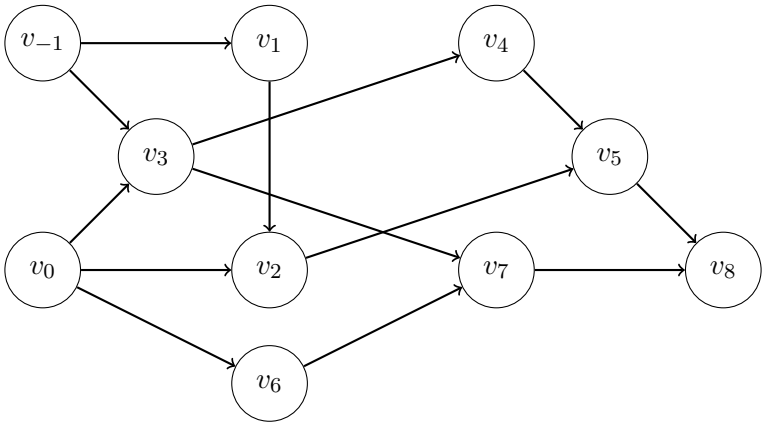


Figure 3.1: Computational graph of the evaluation trace of calculating $y = f(x_1, x_2)$ with $f(x_1, x_2)$ given by (3.5). The precise evaluation trace is listed in Table 3.1.

ciated partial derivative. In this subsection, similarly to as in the introduction of [21], we look at how the partial derivative of $y = f(x_1, x_2)$ with respect to x_1 can be computed, starting from $(x_1, x_2) = (2.5000, 2.0000)$ with $f(x_1, x_2)$ as in (3.5). To this end, we iteratively compute the numerical values of the partial derivatives of all of the intermediate variables v_i with respect to x_i. That is, for every v_i, we also associate the new variable $\dot{v}_i = \frac{\partial v_i}{\partial x_i}$. We see from the evaluation trace that the variable v_1 depends only on the variable v_{-1}. Hence, to compute the partial derivative $\dot{v}_1$, we can apply the chain rule to obtain

$$\dot{v}_1 = \frac{\partial v_1}{\partial v_{-1}} \dot{v}_{-1} = \frac{\dot{v}_{-1}}{v_{-1}}.$$

As stated, we only care about the partial derivatives with respect to x_1 in this subsection. Since $v_{-1} = x_1$ and $v_1 = x_2$, we immediately have $\dot{v}_{-1} = 1.0000$ and $\dot{v}_0 = 0.0000$.

Now, starting from $(x_1, x_2) = (2.5000, 2.0000)$, we perform the first computations of the evaluation trace, computing the partial derivatives as well. First, we set $v_{-1} = x_1$ and $v_1 = x_2$, giving $\dot{v}_{-1} = 1.0000$ and $\dot{v}_0 = 0.0000$. Then we move on to compute

$$v_1 = \log(v_{-1}) = \log(2.5000) = 0.3979.$$

We also compute the derivative as

$$\dot{v}_1 = \frac{\dot{v}_{-1}}{v_{-1}} = \frac{1.0000}{2.5000} = 0.4000.$$

As we have the relation $v_2 = v_1 \cdot v_0$, we compute

$$v_2 = v_1 \cdot v_0 = 0.3979 \cdot 2.0000 = 0.7958,$$

and the associated partial derivative

$$\dot{v}_2 = \frac{\partial v_2}{\partial v_0}\dot{v}_0 + \frac{\partial v_2}{\partial v_1}\dot{v}_1 = v_1\dot{v}_0 + v_0\dot{v}_1 = 0.0000 + 2.0000 \cdot 0.4000 = 0.8000.$$

We continue in a similar fashion to compute the rest of the variables v_i of the evaluation trace, as well as their associated partial derivatives $\dot{v}_i$, leading to the complete evaluation trace listed in Table 3.2.

In the same way, one can compute the partial derivative $\frac{\partial y}{\partial x_2}$ by computing $\dot{v}_i = \frac{\partial v_i}{\partial x_2}$ instead, starting from $\dot{v}_{-1} = 0.0000$ and $\dot{v}_0 = 1.0000$. Thus, to compute $\frac{\partial y}{\partial x_2}$, the evaluation trace has to be traversed once more for component x_2. That is, the complexity of calculating the gradient scales with the number of inputs we want to differentiate with respect to. On the other hand, if we had multiple output variables, the computational graph would still only have to be traversed once for each input variable.

3.2.2 Backward Mode

An alternative approach to the forward mode is the backward mode. We again consider the example (3.5) with evaluation trace listed in Table 3.1. Instead of computing the partial derivatives starting from one specified input variable, we start from an output variable and move backwards through the evaluation trace to obtain the partial derivatives. Similarly to the forward mode, for each intermediate variable v_i we associate a new variable $\bar{v}_i = \frac{\partial y}{\partial v_i}$. When comparing this to the forward mode, we see that instead of computing the partial derivative of the intermediate variables with respect to an input variable, we instead compute the partial derivative of the output variable with respect to the intermediate variable.

As we did when considering the forward mode, we will compute the first iterations of the backward mode before listing the complete evaluation trace. We start by traversing the evaluation trace in Table 3.1 forwards once, so that the intermediate variables v_i and the output y have been computed. Then, we note that $\bar{y} = 1$ by definition. Moving on to the last variable v_8 of the evaluation trace, we see that since $y = v_8$, we must have also $\bar{v}_8 = \bar{y} = 1$. We observe that the only place where v_7 appears on the right-hand side of the evaluation trace is when computing $v_8 = v_5 \cdot v_7$. Hence, the only way v_7 may influence the output y, is via v_8. Making use of the chain rule, we write

$$\bar{v_7} = \frac{\partial y}{\partial v_7} = \frac{\partial y}{\partial v_8} \cdot \frac{\partial v_8}{\partial v_7} = \bar{v_8} \cdot v_5 = 1.0000 \cdot 1.0795 = 1.0795.$$

Whenever the variable v_i appears on the right-hand side in the evaluation trace multiple times, it contributes to the output variables via multiple intermediate variables. Hence, when calculating the partial derivative, all of these contributions must be considered. We see this when trying to compute, e.g., $\bar{v}_3$, which appears on the right-hand sides for

$$v_4 = \cos(v_3) \text{ and } v_7 = v_3 - v_6.$$

Table 3.2: Evaluation trace of the calculation of $y = f(x_1, x_2)$ with $f(x_1, x_2)$ given by (3.5) including the forward trace used to calculate $\frac{\partial y}{\partial x_1}$.

v_{-1}	$=$	x_1	$=$	2.5000		
$\dot{v}_{-1}$	$=$	$\dot{x}_1$	$=$	1.0000		
v_0	$=$	x_2	$=$	2.0000		
$\dot{v}_0$	$=$	$\dot{x}_2$	$=$	0.0000		
v_1	$=$	$\log(v_{-1})$	$=$	$\log(2.5000)$	$=$	0.3979
$\dot{v}_1$	$=$	$\frac{\dot{v}_{-1}}{v_{-1}}$	$=$	$\frac{1.0000}{2.5000}$	$=$	0.4000
v_2	$=$	$v_1 \cdot v_0$	$=$	$0.3979 \cdot 2.0$	$=$	0.7958
$\dot{v}_2$	$=$	$v_1 \cdot \dot{v}_0 + v_0 \cdot \dot{v}_1$	$=$	$0.0000 + 2.0000 \cdot 0.4000$	$=$	0.8000
v_3	$=$	$v_{-1} \cdot v_0$	$=$	$2.5000 \cdot 2.0$	$=$	5.0000
$\dot{v}_3$	$=$	$v_{-1} \cdot \dot{v}_0 + v_0 \cdot \dot{v}_{-1}$	$=$	$0.0000 + 2.0000 \cdot 1.0000$	$=$	2.0000
v_4	$=$	$\cos(v_3)$	$=$	$\cos(5.0000)$	$=$	0.2837
$\dot{v}_4$	$=$	$\sin(v_3) \cdot \dot{v}_3$	$=$	$\sin(5.0000) \cdot 2.0000$	$=$	-1.9178
v_5	$=$	$v_2 + v_4$	$=$	$0.7958 + 0.2837$	$=$	1.0795
$\dot{v}_5$	$=$	$\dot{v}_2 + \dot{v}_4$	$=$	$0.8000 - 1.9178$	$=$	-1.1178
v_6	$=$	$\exp(v_0)$	$=$	$\exp(2.0000)$	$=$	7.3891
$\dot{v}_6$	$=$	$v_6 \cdot \dot{v}_0$	$=$	$7.2891 \cdot 0.0000$	$=$	0.0000
v_7	$=$	$v_3 - v_6$	$=$	$5.0000 - 7.3891$	$=$	-2.3891
$\dot{v}_7$	$=$	$\dot{v}_3 - \dot{v}_6$	$=$	$2.0000 - 0.0000$	$=$	2.0000
v_8	$=$	$v_5 \cdot v_7$	$=$	$1.0795 \cdot (-2.3891)$	$=$	-2.5790
$\dot{v}_8$	$=$	$v_7 \cdot \dot{v}_5 + v_5 \cdot \dot{v}_7$	$=$	$2.3891 \cdot 1.1178 + 1.0795 \cdot 2.0000$	$=$	4.8295
y	$=$	v_8	$=$	-2.5790		
$\dot{y}$	$=$	$\dot{v}_8$	$=$	4.8295		

Once again, we make use of the chain rule to compute

$$\begin{aligned}\bar{v}_3 = \frac{\partial y}{\partial v_3} = \frac{\partial y}{\partial v_7} \cdot \frac{\partial v_7}{\partial v_3} + \frac{\partial y}{\partial v_4} \cdot \frac{\partial v_4}{\partial v_3} = \bar{v}_7 \cdot 1 + \bar{v}_4 \cdot \sin(v_3) \\ = 1.0795 - 2.3891 \cdot \sin(5.0000) = 3.3705.\end{aligned}$$

We proceed in a similar way to compute the rest of the associated variables $\bar{v}_i$. The complete evaluation trace, including the backward pass, is listed in Table 3.3. We note that both of the partial derivatives $\frac{\partial y}{\partial x_1}$ and $\frac{\partial y}{\partial x_2}$ have been computed using only one backward pass through the evaluation trace. We further note that $\frac{\partial y}{\partial x_1}$ computed using the forward and backward mode differs slightly, which is due to rounding errors as we are only using a precision of four decimal places.

Even though we so far have considered the computation of, e.g., $\bar{v}_0$ as one

Table 3.3: Evaluation trace of the calculation of $y = f(x_1, x_2)$ with $f(x_1, x_2)$ given by (3.5) including the backward trace used to calculate $\frac{\partial y}{\partial x_1}$ and $\frac{\partial y}{\partial x_2}$.

v_{-1}	$=$	x_1	$=$	2.5000		
v_0	$=$	x_2	$=$	2.0000		
v_1	$=$	$\log(v_{-1})$	$=$	$\log(2.5000)$	$=$	0.3979
v_2	$=$	$v_1 \cdot v_0$	$=$	$0.3979 \cdot 2.0$	$=$	0.7958
v_3	$=$	$v_{-1} \cdot v_0$	$=$	$2.5000 \cdot 2.0$	$=$	5.0000
v_4	$=$	$\cos(v_3)$	$=$	$\cos(5.0000)$	$=$	0.2837
v_5	$=$	$v_2 + v_4$	$=$	$0.7958 + 0.2837$	$=$	1.0795
v_6	$=$	$\exp(v_0)$	$=$	$\exp(2.0000)$	$=$	7.3891
v_7	$=$	$v_3 - v_6$	$=$	$5.0000 - 7.3891$	$=$	-2.3891
v_8	$=$	$v_5 \cdot v_7$	$=$	$1.0795 \cdot (-2.3891)$	$=$	-2.5790
y	$=$	v_8	$=$	-2.5790		
$\bar{v}_8$	$=$	$\bar{y}$	$=$	1.0000		
$\bar{v}_7$	$=$	$\bar{v}_8 \cdot v_5$	$=$	$1.0000 \cdot 1.0795$	$=$	1.0795
$\bar{v}_5$	$=$	$\bar{v}_8 \cdot v_7$	$=$	$1.0000 \cdot -2.3891$	$=$	-2.3891
$\bar{v}_6$	$=$	$-\bar{v}_7$	$=$	-1.0795		
$\bar{v}_3$	$=$	$\bar{v}_7$	$=$	1.0795		
$\bar{v}_0$	$=$	$\bar{v}_6 \exp(v_0)$	$=$	$-1.0795 \cdot \exp(2.0000)$	$=$	-7.9765
$\bar{v}_4$	$=$	$\bar{v}_5$	$=$	-2.3891		
$\bar{v}_2$	$=$	$\bar{v}_5$	$=$	-2.3891		
$\bar{v}_3$	$=$	$\bar{v}_3 + \bar{v}_4 \cdot \sin(v_3)$	$=$	$1.0795 - 2.3891 \cdot \sin(5.0000)$	$=$	3.3705
$\bar{v}_0$	$=$	$\bar{v}_0 + \bar{v}_3 \cdot v_{-1}$	$=$	$-7.9765 + 3.3705 \cdot 2.5000$	$=$	0.4498
$\bar{v}_{-1}$	$=$	$\bar{v}_3 \cdot v_0$	$=$	$3.3705 \cdot 2.0000$	$=$	6.7410
$\bar{v}_0$	$=$	$\bar{v}_0 + \bar{v}_2 \cdot v_1$	$=$	$0.4498 - 2.3891 \cdot 0.3979$	$=$	-0.5008
$\bar{v}_1$	$=$	$\bar{v}_2 \cdot v_0$	$=$	$-2.3891 \cdot 2.0000$	$=$	-4.7782
$\bar{v}_{-1}$	$=$	$\bar{v}_{-1} + \bar{v}_1 \cdot \frac{1}{v_{-1}}$	$=$	$6.7410 + \frac{-4.7782}{2.5000}$	$=$	4.8297
$\bar{x}_2$	$=$	$\bar{v}_0$	$=$	-0.5008		
$\bar{x}_1$	$=$	$\bar{v}_{-1}$	$=$	4.8297		

computation, these kinds of operations will often be split in different parts. That is, we may compute

$$\begin{aligned}
\bar{v}_0 &= \bar{v}_6 \cdot \exp(v_0) = \quad -1.0795 \cdot \exp(2.0000) = -7.9765, \\
\bar{v}_0 &= \bar{v}_0 + \bar{v}_3 v_{-1} = -7.9765 + 3.3705 \cdot 2.5000 = 0.4498, \\
\bar{v}_0 &= \quad \bar{v}_0 + \bar{v}_2 v_1 = \quad 0.4498 - 2.3891 \cdot 2.0000 = -4.7782.
\end{aligned}$$

This grouping is done to associate each part of the computation of $\bar{v}_1$ with one line of the original evaluation trace, and means that some of the variables $\bar{v}_i$ are both initialized and later incremented. In particular, since v_0 appears on the right-hand side of the original evaluation trace, it is given an initial assignment before being incremented twice. We observe in Table 3.3 that some of the variables $\bar{v}_i$ have been both initialized and incremented.

If we wanted to compute the partial derivatives of more than one output variable, the backward pass would have to be rerun for each output variables. Hence, whenever the number of input variables p is much smaller then the number of output variables m, $p \ll m$, calculating the full Jacobian using the forward mode is more efficient. On the other hand, if $p \gg m$, computing the full Jacobian using the backward mode will be more efficient.

We have in this chapter only scratched the surface of the intricacies of AD. The interested reader may refer to, e.g., [21] for a more detailed discussion. In particular, a detailed explanation of the forward and backward modes can be found in chapter 3 of [21].

As discussed in e.g. [22], there are different memory requirements of the two modes. Computing the gradient in the backward mode entails having to store all intermediate variables, while in the forward mode this is usually not the case. Hence, there is usually a higher storage requirement associated with the backward mode. When computing the gradient of a long simulation, this memory requirement might grow too large.

Chapter 4

Model Extensions

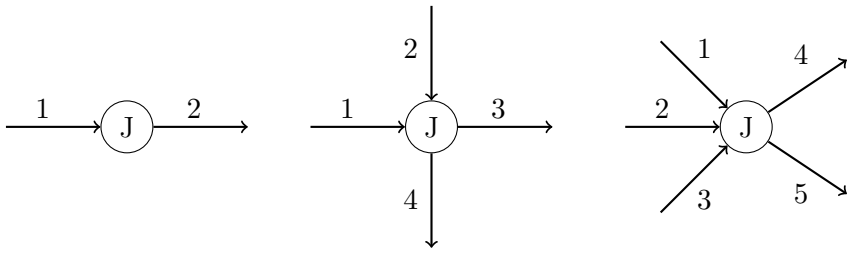
This chapter discusses how the models introduced in Chapter 2, which were all related to a single unidirectional road, can be extended to represent more complex traffic systems. If we want to model a more complex road network, these models need to be extended with features such as junctions, traffic lights, roundabouts, variable speed limits, and mixed-vehicle traffic (cars and buses). How to extend the basic model to describe each of these features will be discussed in the following sections.

4.1 Junctions

One of the most important components needed to extend a set of unidirectional roads to a full road network are the junctions that connect the roads. Hence, it is imperative to also consider what happens at junctions. There is a substantial amount of work on junction modelling. Theoretical results can be found in, e.g., [23], [24] or [25]. Alternatively, see, e.g., [26] for numerical results. We recall from Chapter 2 that we want to model the average density of traffic along each of the roads. The idea behind the junction modelling approach that we will discuss in this section, is to try to estimate the amount of traffic crossing the junction and use this to update the average densities on either sides of the junction.

We recall that the basic modelling assumption behind the models in Chapter 2 was the conservation of mass of vehicles. Clearly, this must also hold across junctions. However, as we will see, this assumption will not be enough to ensure a unique solution for a general junction. In this section, we will start by considering the simplest 1-to-1 junction, before considering increasingly complex junctions. As we will see, the usual entropy conditions do not hold in general for junctions, and new conditions need to be imposed to regain uniqueness. The two main rules we will impose are due to Garavello and Piccoli [24], and are summarized as

(A) There are some prescribed preferences of the drivers, so that the traffic



(a) 1-to-1 (b) 2-to-2 (c) 3-to-2

Figure 4.1: Different types of junctions

from incoming roads is distributed on outgoing roads according to fixed coefficients,

(B) Respecting (A), drivers choose so as to maximize fluxes.

These rules are unfortunately not enough to gain uniqueness for all junctions, so we need introduce extra rules to regain uniqueness. We will first consider the 1-to-1, 1-to-2 and the 2-to-1 junctions in extra detail before generalizing to the n-to-m junction. We will introduce a few ideas that to the best of our knowledge have not been considered before:

- In Section 4.1.3, we introduce a new way of determining how traffic merges.
- In Section 4.1.4, we introduce a new way of implementing yielding rules.
- In Section 4.1.5, we generalize this yielding rule and the new way of considering merging traffic to more general junctions.

Figure 4.1 shows some of the different junctions that we will develop a solution approach for.

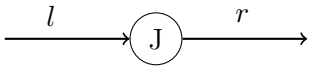


Figure 4.2: 1-to-1 junction

4.1.1 1-to-1 Junction

We first consider a 1-to-1 junction with incoming road l and outgoing road r as visualized in Figure 4.2. As explained in [27], letting the densities on the roads be denoted by ρ_1 and ρ_2, the LWR model introduced in Section 2.1.1 applied to a 1-to-1 junction is equivalent to the following problem:

$$\rho_t + (f(\rho))_x = 0, \quad \rho(0,x) = \rho_0(x), \tag{4.1}$$

with

$$\begin{aligned} f_l(\rho) = \gamma_l \rho v(\rho), \quad f_r = \gamma_r \rho v(\rho), \\ f(x,\rho) = H(x) f_r(\rho) + (1 - H(x)) f_l(\rho), \end{aligned} \tag{4.2}$$

where $H(x)$ is the Heaviside function

$$H(x) = \begin{cases} 0 \text{ if } x < 0, \\ 1 \text{ if } x \geq 0, \end{cases}$$

and $\gamma_l = v_l^{\max}/L_l$. Here f_l and f_r refer to the flux functions on roads l and r, which may in general be different.

In contrast to the usual LWR model equation, which could have discontinuities in the density, this problem can in general also have discontinuities in the flux function. The discontinuities in the flux function comes from the fact that roads may have different speed limits, different number of lanes, or simply are described by a different flux function. Because of the discontinuity of the flux function, the usual entropy conditions do not apply at a general 1-to-1 junction. Aiming to prove the well-posedness of the Cauchy problem with discontinuous flux function, we will focus on the conditions arising from so-called supply and demand functions, see for instance [28].

We still consider a 1-to-1 junction with roads l and r and with respective flux functions f_l and f_r. As before we assume $f_l, f_r \in C^2$ and strictly concave with $f_l(0) = f_r(0) = f_l(1) = f_r(1) = 0$. We assume that f_l and f_r attain their unique maxima at σ_l and σ_r and define the demand and supply functions as

$$D_l(\rho) = \begin{cases} f_l(\rho) & \text{if } \rho \leq \sigma_l, \\ f_l(\sigma_l) & \text{if } \rho > \sigma_l, \end{cases} \text{ and } \quad S_r(\rho) = \begin{cases} f_r(\sigma_r) & \text{if } \rho \leq \sigma_r, \\ f_r(\rho) & \text{if } \rho > \sigma_r. \end{cases} \tag{4.3}$$

We could instead have defined the demand and supply functions for a general road i. However, as we shall see, the demand function is associated with incoming roads, and the supply function is associated with outgoing roads. Thus, to avoid any confusions we for now only define D_l and S_r as in (4.3). The names demand and supply can also lead to some confusion. The demand relates to incoming roads, and can be understood as a measure of the amount of traffic that demands to exit road l through the junction. On the other hand, supply relates to outgoing roads, and can be understood as a measure of the amount of exit traffic road r supplies to the junction. A visualization of the demand and supply functions compared to the usual flux function can be seen in Figure 4.3. Here we have used the flux function $f(\rho) = \rho(1-\rho)$, which attains its unique maximum at $\sigma = 1/2$.

We observe that for the 1-to-1 junction, all drivers from road l will aim to go to road r, and so rule (A) will be trivially satisfied. The usual choice in traffic modelling is to choose the flux across the junction as the minimum of D_l and S_r as done in, e.g., [28]. We write the maximal flux across the junction as

$$f_{l,r} = \min\{D_l(\rho_l), S_r(\rho_r)\}. \tag{4.4}$$

Since $D_l(\rho_l)$ is the maximal flux exiting road l and $S_r(\rho_r)$ is the maximal flux entering road r, choosing the flux $f_{l,r}$ according to (4.4) satisfies rule (B).

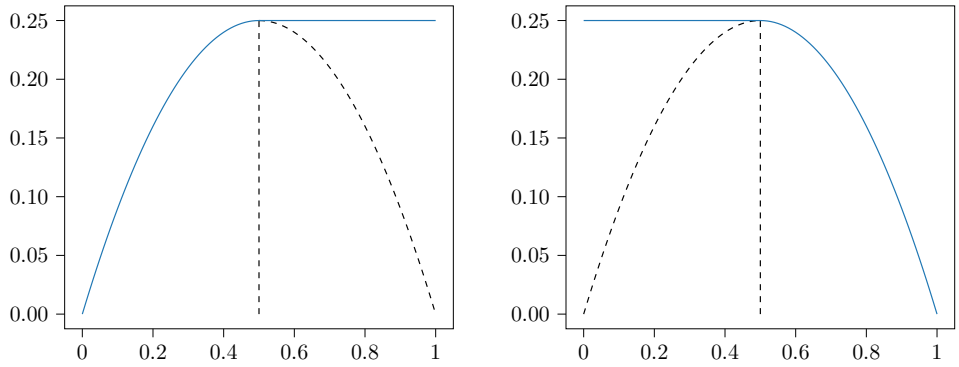


Figure 4.3: The demand and supply functions to the right and left, respectively, for flux function $f(\rho) = \rho(1-\rho)$. The dashed lines represent the flux functions $f(\rho)$ of the respective roads, and the colored lines represent the demand and supply functions of the junction.

In this and the following subsections, we will denote the flux going from road l to road r by $f_{l,r}$. We notice that whenever the flux function is equal on the two roads, that is $f(\cdot) = f_l(\cdot) = f_r(\cdot)$, the flux across the junction, $f_{l,r}$ reduces to

$$f_{l,r} = \begin{cases} f(\rho_l) & \text{if } \rho_l \leq \sigma \text{ and } \rho_r \leq \sigma, \\ f(\rho_r) & \text{if } \rho_l \geq \sigma \text{ and } \rho_r \geq \sigma, \\ \min\{f(\rho_l), f(\rho_r)\} & \text{if } \rho_l \leq \sigma \text{ and } \rho_r \geq \sigma, \\ f(\sigma) & \text{otherwise.} \end{cases}$$

In particular, if $f(\rho) = f_l(\rho) = f_r(\rho)$ and $\rho_l = \rho_r$, we have $f_{l,r} = f(\rho_l) = f(\rho_r)$, meaning that the flux function is consistent.

We recall that the numerical scheme for updating the densities of traffic can be written on the form, here for road l:

$$\rho_{l,k}^{n+1} = \rho_{l,k}^n - \lambda^n [F_{l,k+1/2}^n - F_{l,k-1/2}^n], \tag{4.5}$$

where $F_{l,k\pm1/2}^n$ is calculated using some consistent numerical flux function. Here we have used the double index to indicate that this is the density on road l. After calculating the fluxes $f_l(\rho_l)$ and $f_r(\rho_r)$, we plug these values into the numerical scheme as as one of the numerical fluxes. Denoting the rightmost cell on road l as k_l and the leftmost cell on road r as k_r, we write the updated schemes for the boundary cells as

$$\begin{aligned} \rho_{l,k_l}^{n+1} &= \rho_{l,k_l}^n - \lambda^n [f_l - F_{l,k_l-1/2}^n], \\ \rho_{r,k_r}^{n+1} &= \rho_{r,k_r}^n - \lambda^n [F_{r,k_r+1/2}^n - f_r]. \end{aligned} \tag{4.6}$$

Here, the fluxes f_l and f_r denote the flux exiting road l and entering road r respectively. For the 1-to-1 junction these will be equal to each other as $f_l = f_r = f_{l,r}$.

As discussed in Section 2.2.1, we want our numerical scheme to be conservative. We saw that this was the case for a general finite-volume scheme on the form

$$\rho_j^{n+1} = \rho_j^n - \lambda(F_{j+1/2}^n - F_{j-1/2}^n), \tag{4.7}$$

because each of the fluxes $F_{j+1/2}^n$ appear with opposite signs when updating ρ_j^n and ρ_{j+1} respectively. Hence, when summing over all cells, the interior fluxes will cancel out. We saw that the fluxes f_l and f_r are equal, and since they appear with opposite signs in the updating schemes, they will cancel out when summing over all cells. The updating form (4.7) is only used for interior cells, and so the right flux of the rightmost interior cell and the left flux of leftmost interior cell only appear once each. Hence, when updating the boundary cells, we need to make sure that these fluxes are included with opposite signs so that the scheme remains conservative.

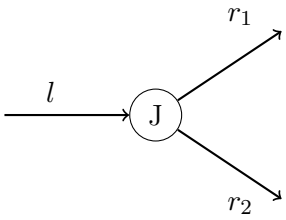


Figure 4.4: 1-to-2 junction

4.1.2 1-to-2 Junction

The 1-to-2 junction is a junction where a single incoming road l diverges into two outgoing roads, r_1 and r_2. A visual representation of this network can be seen in Figure 4.4. In this case, we need a distribution parameter α specifying how the traffic distributes among the two outgoing roads. This parameter needs to be specified, and could, i.e., be estimated using empirical data collected using real traffic. The distribution parameter serves the role of ensuring that Rule (A) is satisfied. Assuming the distribution parameter is known, there are two main rules one can use when estimating the maximum fluxes across the junction, so as to satisfy Rule (B): a FIFO (first-in-first-out) rule or a non-FIFO rule.

Denoting the densities on the roads as ρ_l, ρ_{r_1} and ρ_{r_2}, a FIFO rule leads to the fluxes

$$\begin{aligned} f_l := \min\Big\{ D_l(\rho_l), \frac{S_{r_1}(\rho_{r_1})}{\alpha}, \frac{S_{r_2}(\rho_{r_2})}{1-\alpha} \Big\}, \\ f_{r_1} := \alpha f_l, \\ f_{r_2} := (1-\alpha) f_l. \end{aligned} \tag{4.8}$$

Here, f_l denotes the flux exiting road l while f_{r_1} and f_{r_2} denote the fluxes entering roads r_1 and r_2, respectively. The FIFO rule models drivers having to

wait for drivers in front even if these drivers are heading to a different outgoing road. Thus, if one of the roads is fully congested, no flux will cross the junction, regardless of the density on the other outgoing road. For details about how the optimal solution can be found when a FIFO rule is applied, see, e.g., [24].

A different approach is to make use of a non-FIFO rule, as done in, e.g., [28], where some flow across the junction is allowed, even if one road is fully congested. In this case, the fluxes across the junction are defined as follows

$$\begin{aligned} f_{r_1} &:= \min\{\alpha D_l(\rho_l), S_{r_1}(\rho_{r_1})\}, \\ f_{r_2} &:= \min\{(1-\alpha) D_l(\rho_l), S_{r_2}(\rho_{r_2})\}, \\ f_l &:= f_{r_2} + f_{r_2}. \end{aligned} \tag{4.9}$$

We notice that the non-FIFO rule is analogous to solving the problem for two 1-to-1 junctions separately and then summing the fluxes to form the full solution. Since the f_{r_1} and f_{r_2} are found independent from each other, maximizing both of them separately leads to the overall maximum flux. In Section 4.1.1, we observed that the flux is maximized across a single 1-to-1 by design, and maximizing the flux on each of the 1-to-1 junctions separately thus also maximizes the overall flux, and hence rule (B) is satisfied. In this thesis, we will make use of the non-FIFO rule for the general junction.

Similarly to the previous section, we denote the rightmost node as l_k and the two leftmost nodes as r_{k_1} and r_{k_2}, respectively. The schemes for updating the ghost cell edge is written as

$$\begin{aligned} \rho^{n+1}_{l,k_l} &= \rho^n_{l,k_l} - \lambda^n [f_l - F^n_{l,k_l-1/2}], \\ \rho^{n+1}_{r_1,k_{r_1}} &= \rho^n_{r_1,k_{r_1}} - \lambda^n [F^n_{r_1,k_{r_1}+1/2} - f_{r_1}], \\ \rho^{n+1}_{r_2,k_{r_2}} &= \rho^n_{r_2,k_{r_2}} - \lambda^n [F^n_{r_2,k_{r_2}+1/2} - f_{r_2}]. \end{aligned} \tag{4.10}$$

As in Section 4.1.1, one must take care when updating the boundary cells that the scheme remains conservative.

4.1.3 2-to-1 Junction

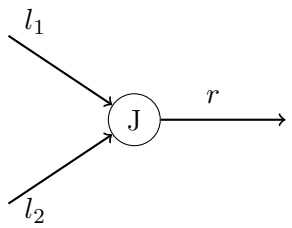


Figure 4.5: 2-to-1 junction

In a 2-to-1 junction, the traffic from two incoming roads merges together in one outgoing road; see Figure 4.5. We denote the incoming roads by l_1 and l_2

and the outgoing road by r. In this junction, there is no need for a distribution parameter, since both incoming roads are leading to the same outgoing road. Hence, rule (A) is trivially satisfied. However, if all of the traffic is not allowed to enter the junction, we will need an extra condition to determine from which road the traffic through the junction is coming from.

Commonly, one of the two roads have priority over the other either because of the priority-to-the-right rule where drivers yield for drivers to their right, or because of some other yield rule. However, even if one road has priority over the other, there will usually be some merging traffic, meaning that traffic from both roads enter the junction. In a microscopic traffic model, if drivers arrive at the junction at different times, there is no yielding. In a macroscopic this is not as easy. One needs to determine the amount of merging, which we assume does not only depend on the demand of the road with priority, but also on the density on that road. Usually, when there is a very high density, some drivers will slow down and allow some drivers from the other road pass through the junction in front of them. On the other hand, when density is low (and the speed is high) there will often be enough space for some merging. It seems that the amount of merging should depend on the density of the road with priority, and that the merging should have local maxima at minimal and maximal densities. The exact relation between density and amount of merging is not obvious.

Summing up, we need a *right-of-way* parameter P, potentially dependent on the densities on the roads, to determine where the traffic is coming from. Follow [24] and adapting the notation to suit ours, assuming that there exists a priority parameter $P \in (0, 1)$, we assign the rule

(C) Assuming that not all vehicles can enter the outgoing road and let C be the amount that can do it. Then PC vehicles come from the first incoming road and $(1 - P)C$ vehicles from the second.

If the right-of-way parameter P is known, we can use the usual demand and supply functions to find the flow of traffic by maximizing the fluxes. However, before investigating how the fluxes can be maximized, we first take a step back and look at how such a right-of-way parameter can be determined. The common approach from the literature has been to fix a constant parameter, potentially choosing it in a way to match experimental data, see, e.g., [28] or [24].

In this thesis, however, we assume that the extent of merging depends on the traffic densities on the roads and make use of an alternative method that, to the best of our knowledge, is a new way of modelling merging of traffic. We assume that the right-of-way parameter P is determined by a smooth function $P = h(\rho_{l_1})$ that depends on the traffic density of the road with priority. We start by stating some properties that it is reasonable for h to satisfy:

- We assume that h is bounded above by a constant, i.e., $h(\rho) \leq h_{\max} < 1 \, \forall \, \rho \in [0, 1]$. This ensures that all of the traffic entering road r will at

no point only come from road l_1.

- The extent of merging between the two roads will be greater when road l_1 is almost fully congested. This is motivated by the observation that when there is heavy traffic, some drivers on the priority road will stop (or momentary slow down) to let vehicles from the subordinate road enter the junction.
- Since the demand D_{l_1} is given as $\rho_{l_1} v(\rho_{l_1})$, fixing the demand but increasing the speed results in a lower density on the road. Even if the demand remains the same, we assume that since the density is low, more drivers from the subordinate road will be able to merge into the traffic from the priority road.

Combing these assumptions, we have that $h(\rho_{l_1})$ should be a bounded function that is increasing for ρ_{l_1} close to 0 and decreasing for ρ_{l_1} close to 1. Recall that a higher value of P results in less merging of traffic, which is why $h(\rho_{l_1})$ attains local minima at the end points. We further assume that h attains a local maximum $h_{\max}$ for some $\rho_m \in (0, 1)$, at which point there is little merging of traffic. Denoting the values of the two local minima as h^0 and h^1 respectively, any smooth interpolation between the three points may be a suitable choice for h. Finally, since road l_1 has priority over road l_2, one can make the assumption that $h(\rho) > 0.5$.

One natural choice for h is to make use of a cubic Hermite spline interpolation. We will here only give a brief introduction to the basic ideas. For more details, see, e.g., [29] or [30]. Hermite interpolation is a type of interpolation that uses information about both the derivative and the value at a number of points. In our case, we want to use the three points $(\rho_0, h^0), (\rho_m, h^{\max})$ and (ρ_1, h^1). We denote by m_0, m_1 and m_3 the tangents at each of these points.

For the tangent m_0 at the left end ($\rho_0 = 0$) we make use of a one-sided finite-difference approximation:

$$m_0 = \frac{h^{\max} - h^0}{\rho_m - \rho_0} = \frac{1}{\rho_m}(h^{\max} - h^0).$$

Since we want a local maximum to be at $(\rho_m, h^{\max})$, we set $m_1 = 0$. Finally for the right edge, we again make use of a one-sided finite-difference approximation:

$$m_2 = \frac{h^1 - h^{\max}}{\rho_1 - \rho_m} = \frac{h^1 - h^{\max}}{1 - \rho_m}.$$

For each of the two intervals $[0, \rho_m)$ and $[\rho_m, 1]$ we define the splines

$$\begin{aligned} p_1(\rho) = h_{00}(t_1)h^0 &+ h_{10}(t_1)(\rho_m - \rho_0)m_0 \\ &+ h_{01}(t_1)h^{\max} + h_{11}(t_1)(\rho_m - \rho_0)m_1 \\ &= h_{00}(t_1)h^0 + h_{10}(t_1)\rho_m m_0 + h_{01}(t_1)h^{\max}, \end{aligned} \quad (4.11)$$

and

$$\begin{aligned} p_2(\rho) = h_{00}(t_2)h^{\max} &+ h_{10}(t_2)(1-\rho_m)m_1 \\ &+ h_{01}(t_2)h^1 + h_{11}(t_2)(1-\rho_m)m_2 \\ &= h_{00}(t_2)h^{\max} + h_{01}(t_2)h^1 + h_{11}(t_2)(1-\rho_m)m_2, \end{aligned} \tag{4.12}$$

where

$$t_1 = \frac{\rho - \rho_0}{\rho_m - \rho_0} = \frac{\rho}{\rho_m} \qquad \text{and} \qquad t_2 = \frac{\rho - \rho_m}{\rho_1 - \rho_m} = \frac{\rho - \rho_m}{1 - \rho_m}. \tag{4.13}$$

The functions h_{00}, h_{10}, h_{01} and h_{11} are called Hermite basis functions, and are written, i.e., as

$$\begin{aligned} h_{00}(t) &= (1+2t)(1-t)^2, & h_{10}(t) &= t(1-t)^2, \\ h_{01}(t) &= t^2(3-2t), & h_{11}(t) &= t^2(t-1). \end{aligned} \tag{4.14}$$

A graphic visualization of these basis functions can be seen in Figure 4.6. One choice of the priority function is then

$$h(\rho; h^0, h^{\max}, h^1, \rho_m) = \begin{cases} p_1(\rho) & \text{if } \rho < \rho_m, \\ p_2(\rho) & \text{otherwise.} \end{cases} \tag{4.15}$$

Figure 4.7 shows how the priority function h looks like for some different choice

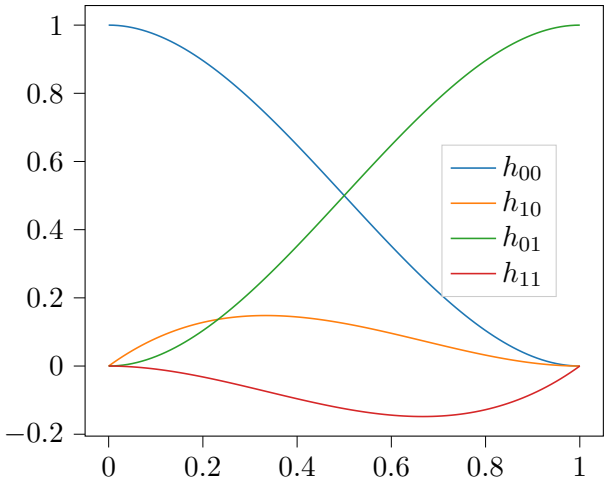


Figure 4.6: The four Hermite basis functions on [0,1] for constructing cubic Hermite splines.

of $h^0, h^{\max}, h^1$ and ρ_m. We note that one could define the function h in different ways. One could make more assumptions to obtain more interpolation points, or use different interpolation methods.

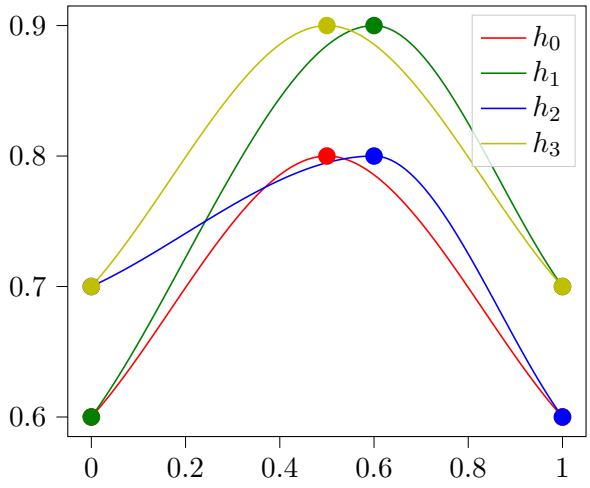


Figure 4.7: The priority function as defined in (4.15) for some choices of parameters.

Assuming that the right-of-way parameter P is determined by some feasible priority function h, we return to the goal of maximizing the flux through the junction while satisfying Rules (A) and (B), and satisfying Rule (C) as closely as possible. We recall that the incoming fluxes are bounded by their respective demand functions $D_{l_1}(\rho_{l_1})$ and $D_{l_2}(\rho_{l_2})$. We also know that the total amount of traffic entering road r is bounded by the supply function $S_r(\rho_r)$. Thus, any feasible solution (f_{l_1}, f_{l_2}, f_r) must have its two first components in the region

$$\Omega = \{(f_{l_1}, f_{l_2}) : 0 \leq f_i \leq D_{l_i}(\rho_{l_i}), i = 1, 2, \quad 0 \leq f_1 + f_2 \leq S_r(\rho_r)\}.$$

Knowing the two first components f_{l_1} and f_{l_2} one can determine the third component by conserving the mass and putting

$$f_r = f_{l_1} + f_{l_2}.$$

We recall that f_{l_i} denotes the flux exiting roads l_i, $i = 1, 2$ leading into the junction and that f_r denotes the flux entering outgoing road r. We will now consider the optimal solution in some different cases. These cases are namely the case where all of the traffic is allowed to enter the outgoing road, and the case where some of the traffic is not allowed to enter.

Demand-Limited Case

The demand-limited case is the case where

$$D_{l_1}(\rho_{l_1}) + D_{l_2}(\rho_{l_2}) < S_r(\rho_r),$$

and hence all of the traffic is allowed to enter. As mentioned, Rules (A) and (B) are the most important ones. Rule (C) is only applied when all of the

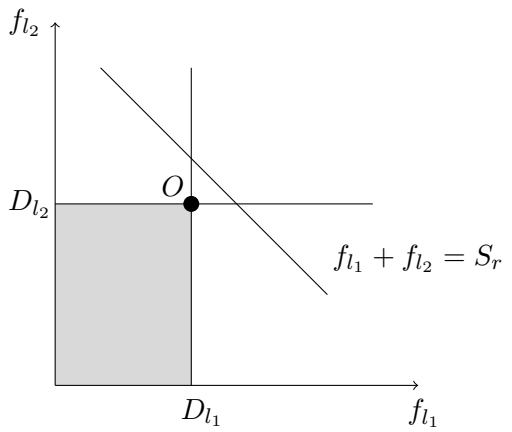


Figure 4.8: Optimal solution in the demand limited case denoted by O.

traffic is not allowed to enter the outgoing road. With only one outgoing road, Rule (A) is satisfied trivially. Further, according to Rule (B), the optimal solution should maximize the outgoing flux in the junction, meaning that it should lie as close to the line $f_{l_1} + f_{l_2} = S_r(\rho_r)$ as possible. In the demand-limited case, this line will be outside the feasible region Ω, and the optimal value will be chosen as the point in Ω minimizing the distance to the line. This point will be $(f_{l_1}^*, f_{l_2}^*) = (D_{l_1}, D_{l_2})$. A graphical visualization of the optimal value in the demand-limited case can be seen in Figure 4.8. This figure is inspired by figures in section 5.2 of [24].

Supply-Limited Case

The more interesting case is the supply-limited case. This is the case where

$$D_{l_1} + D_{l_2} > S_r(\rho_r).$$

As before, we want the optimal solution to lie on the line $f_{l_1} + f_{l_2} = S_r(\rho_r)$ so as to maximize the outgoing flux. In this case, there exists a non-empty segment

$$Q = \Omega \cap \{(f_{l_1}, f_{l_2}) : f_{l_1} + f_{l_2} = S_r(\rho_r)\}$$

on which the optimal solution will lie. The optimal solution $(f_{l_1}^*, f_{l_2}^*)$ will lie on Q, as Q is precisely the set of points in Ω maximizing the flux. As all of these points give the same amount of flux across the junction, they will all be equally good solutions according to Rule (B). Hence, to determine which of these points to choose, we also need to make use of Rule (C). As done in [24], we consider the space (f_{l_1}, f_{l_2}) and the line

$$f_{l_2} = \frac{1 - P}{P} f_{l_1}. \tag{r}$$

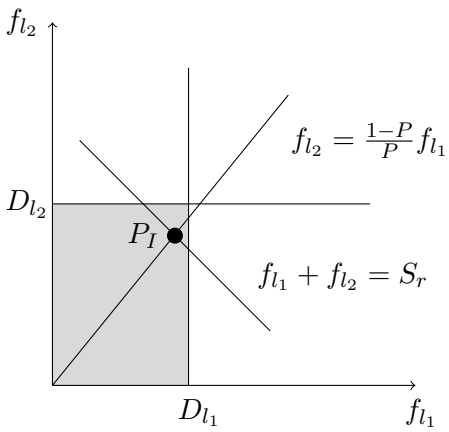


Figure 4.9: Optimal solution in the supply limited when the intersection between the priority line and the maximal flux line lie in the feasible region. Optimal value denoted by P_I.

It is clear that the line (r) is the set of points satisfying Rule (C): if C is the total outflux, Rule (C) dictates that $f_{l_1} = PC$ and $f_{l_2} = (1-P)C$, from which it follows that $f_{l_2}/f_{l_1} = (1-P)/P$.

We want the optimal solution to either satisfy (C) exactly, or to be the point on Q closest to a point satisfying (C). To this end, we denote the intersection between (r) and $f_{l_1} + f_{l_2} = S_r(\rho_r)$ by P_I. If $P_I \in \Omega$, we simply choose the optimal solution as the point P_I. If instead $P_I \notin \Omega$, we pick the optimal solution as the point on the segment Q that minimizes the distance to the point P_I. Figures 4.9 and 4.10 show graphical visualizations of the optimal solution in the supply-limited case when the intersection point lies inside and outside the feasible region, respectively. These figures are inspired by the figure in section 5.2 of the book [24].

The point of intersection P_I can be easily calculated, and is given by

$$P_I = (PS_r(\rho_r), (1-P)S_r(\rho_r)).$$

If we realize that at most one of the components of the point $(PS_r(\rho_r), (1-P)S_r(\rho_r))$ can be outside the feasible region, we can consider the two cases where $P_I \notin \Omega$ separately. If $PS_r(\rho_r) > D_{l_1}(\rho_{l_1})$, we know that the closest point will be $(D_{l_1}(\rho_{l_1}), S_r(\rho_r) - D_{l_1}(\rho_{l_1}))$. If instead $(1-P)S_r(\rho_r) > D_{l_2}(\rho_{l_2})$ we find the closest point as $(S_r(\rho_r) - D_{l_2}(\rho_{l_2}), D_{l_2}(\rho_{l_2}))$.

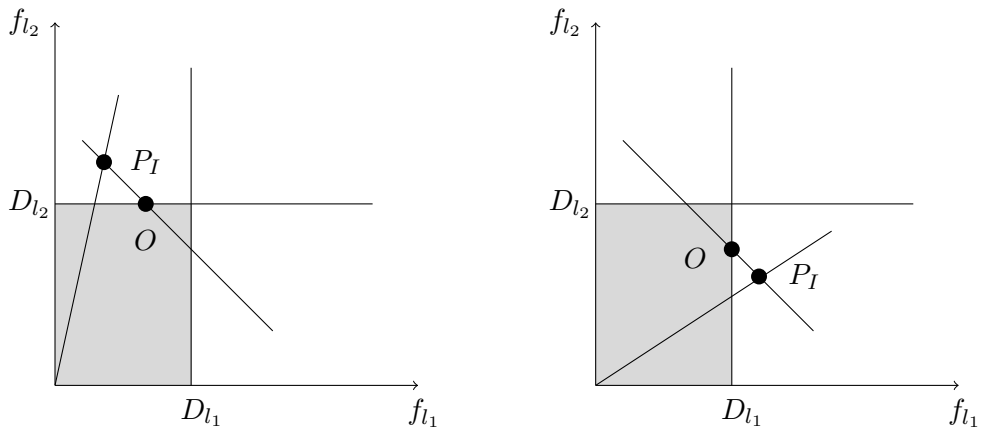


Figure 4.10: Optimal solution for the supply limited case where the intersection between the priority line and the maximal flux line lies outside the feasible region. In both figures, the optimal value is denoted by O.

Explicit Solution

In the simulation it is desirable to be able to write the exact solution in a simple closed form. In [28], the following definition of the fluxes is suggested

$$\begin{aligned} f_{l_1} &:= \min\big\{D_{l_1}(\rho_{l_1}), \max\{PS_r(\rho_r), S_r(\rho_r) - D_{l_2}(\rho_{l_2})\}\big\}, \\ f_{l_2} &:= \min\big\{D_{l_2}(\rho_{l_2}), \max\{(1-P)S_r(\rho_r), S_r(\rho_r) - D_{l_1}(\rho_{l_1})\}\big\}, \\ f_r &:= f_{l_1} + f_{l_2}, \end{aligned} \tag{4.16}$$

where $P \in (0,1)$ is the right-of-way parameter. We now check that the fluxes as defined by (4.16) overlap with the optimal values for the different cases.

First, we consider the demand-limited case; that is, the case where

$$D_{l_1}(\rho_{l_1}) + D_{l_2}(\rho_{l_2}) < S_r(\rho_r).$$

This implies that

$$S_r(\rho_r) - D_{l_1}(\rho_{l_1}) > D_{l_2}(\rho_{l_2}) \qquad \text{and} \qquad S_r(\rho_r) - D_{l_2}(\rho_{l_2}) > D_{l_1}(\rho_{l_1}).$$

Hence, we see that the calculated fluxes from (4.16) take the values

$$f_{l_1} = D_{l_1} \qquad \text{and} \qquad f_{l_2} = D_{l_2},$$

which is the same values we arrived at before.

We also check that the fluxes are correct in the supply-limited case, where

$$D_{l_1}(\rho_{l_1}) + D_{l_2}(\rho_{l_2}) > S_r(\rho_r).$$

First, we assume that the intersection point P_I is in the feasible region Q; that is, we have

$$PS_r(\rho_r) \leq D_{l_1}(\rho_{l_1}) \qquad \text{and} \qquad (1-P)S_r(\rho_r) \leq D_{l_2}(\rho_{l_2}).$$

These inequalities in turn imply that

$$\begin{aligned} S_r(\rho_r) - D_{l_2}(\rho_{l_2}) &\leq PS_r(\rho_r) \\ (1-P)S_r(\rho_r) = S_r(\rho_r) - PS_r(\rho_r) &\geq S_r(\rho_r) - D_{l_1}(\rho_{l_1}). \end{aligned}$$

Now, by combining these inequalities one can check that the fluxes calculated from (4.16) take the values

$$f_{l_1} = PS_r(\rho_r) \qquad \text{and} \qquad f_{l_2} = (1-P)S_r(\rho_r).$$

Finally, we check the case where the intersection point P_I lies outside the feasible region Ω. As discussed, this can occur when

$$PS_r(\rho_r) > D_1(\rho_{l_1}) \qquad \text{and} \qquad (1-P)S_r(\rho_r) < D_2(\rho_{l_2}), \tag{4.17}$$

or when

$$PS_r(\rho_r) < D_1(\rho_{l_1}) \qquad \text{and} \qquad (1-P)S_r(\rho_r) > D_2(\rho_{l_2}). \tag{4.18}$$

Assuming that (4.17) holds, we note that $S_r(\rho_r) - D_2(\rho_{l_2}) > PS_r(\rho_r)$, and since we already know that $S_r(\rho_r) - D_2(\rho_{l_2}) < D_1(\rho_{l_1})$, we conclude that

$$f_{l_1} = S_r(\rho_r) - D_2(\rho_{l_2}).$$

We can also check that

$$f_{l_1} = D_2(\rho_{l_2}).$$

By an almost identical procedure, one can check that the form (4.16) also holds true when (4.18) is true.

With the fluxes f_{l_1}, f_{l_2} and f_r calculated according to (4.16), we can write the updating of the boundaries as

$$\begin{aligned} \rho^{n+1}_{l_1,k_{l_1}} &= \rho^n_{l_1,k_{l_1}} - \lambda^n[f_{l_1} - F^n_{l_1,k_{l_1}-1/2}], \\ \rho^{n+1}_{l_2,k_{l_2}} &= \rho^n_{l_2,k_{l_2}} - \lambda^n[f_{l_2} - F^n_{l_2,k_{l_2}-1/2}], \\ \rho^{n+1}_{r,k_r} &= \rho^n_{r,k_r} - \lambda^n[F^n_{r,k_r+1/2} - f_r]. \end{aligned} \tag{4.19}$$

We now recall from Section 2.2.2 that a CFL condition needs to be imposed on the time step to ensure stability of the scheme. In (2.22), this condition was stated using the densities of the cells. When we have more than one incoming road, the fluxes leading into the outgoing road is a combination of fluxes coming from two different roads, and so the CFL condition in (2.22) might not be enough to ensure stability. Denoting the flux leading into road r by f_r, we

want to know the wave speed of a wave with the same flux coming from a cell on the outgoing road. To this end, we start by finding the two densities ρ^+ and ρ^- that satisfy

$$f_r = f(\rho^+) = f(\rho^-),$$

and

$$\rho^+, \rho^- \neq 0.5 \implies \rho^+ \neq \rho^-$$

We recall the expression for the flux as a function of the density as

$$f(\rho) = \gamma\rho(1-\rho).$$

Setting this flux equal to f_r gives

$$\rho^2 - \rho - \frac{f_r}{\gamma} = 0 \implies \rho = \frac{1}{2} \pm \frac{\sqrt{1-4\frac{f_r}{\gamma}}}{2}.$$

The wave speed is determined by $|f'(\rho)|$ and so by symmetry, the positive or negative roots both give the same speed. Inserting the negative root into the derivative of the flux gives

$$f'(\rho^-) = \gamma(1-1+\sqrt{1-4\frac{f_r}{\gamma}}) = \gamma\sqrt{1-4\frac{f_r}{\gamma}}.$$

This leads to a second CFL condition

$$\Delta t \leq \frac{\Delta x}{\gamma\sqrt{1-4\frac{f_r}{\gamma}}}. \tag{4.20}$$

To ensure stability, we require the time step to satisfy both (2.22) and (4.20).

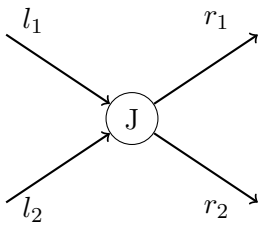


Figure 4.11: 2-to-2 junction

4.1.4 2-to-2 junction

We will now combine the ideas from the 1-to-2 junction and the 2-to-1 junction. In the 2-to-2 junction, we need to have both distribution and right-of-way parameters. We will continue with the new way of determining the extent of merging we introduced in the previous section. In addition, we will introduce

a method for handling yielding rules that to the best of our knowledge has not been considered before.

Similarly to the previous sections, we denote the two incoming roads by l_1 and l_2 and the outgoing roads by r_1 and r_2. A visualization of this network is shown in Figure 4.11. We introduce the distribution matrix

$$A = \begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} \\ \alpha_{2,1} & \alpha_{2,2} \end{bmatrix}$$

and the right-of-way parameters $P = (P_1, P_2)$. Since all of the flow needs to be distributed among the outgoing roads, we impose the condition of a unit rowsum in A,

$$\alpha_{i,1} + \alpha_{i,2} = 1 \text{ for } i \in \{1, 2\}.$$

Following the ideas from the previous section, we try to estimate the right-of-way parameters using some functions h_1 and h_2. Making use of the priority function h from the previous section, we define

$$h_1 := h(\alpha_{1,1}\rho_{l_1}) \qquad \text{and} \qquad h_2 := h(\alpha_{1,2}\rho_{l_1}). \tag{4.21}$$

With some way of determining the right-of-way parameters P, a first approach is to try and generalize the fluxes defined in [28] to a 2-to-2 junction as

$$\begin{aligned}
f_{l_1,r_1} &:= \min\big\{\alpha_{1,1}D_{l_1}(\rho_{l_1}), \max\{P_1 S_{r_1}(\rho_{r_1}), S_{r_1}(\rho_{r_1}) - \alpha_{2,1}D_{l_2}(\rho_{l_2})\}\big\}, \\
f_{l_2,r_1} &:= \min\big\{\alpha_{2,1}D_{l_1}(\rho_{l_1}), \max\{(1-P_1) S_{r_1}(\rho_{r_1}), S_{r_1}(\rho_{r_1}) - \alpha_{1,1}D_{l_2}(\rho_{l_2})\}\big\}, \\
f_{r_1} &:= f_{l_1,r_1} + f_{l_2,r_1}, \\
f_{l_1,r_2} &:= \min\big\{\alpha_{1,2}D_{l_1}(\rho_{l_1}), \max\{P_2 S_{r_2}(\rho_{r_2}), S_{r_2}(\rho_{r_2}) - \alpha_{2,2}D_{l_2}(\rho_{l_2})\}\big\}, \\
f_{l_2,r_2} &:= \min\big\{\alpha_{2,2}D_{l_2}(\rho_{l_2}), \max\{(1-P_2) S_{r_2}(\rho_{r_2}), S_{r_2}(\rho_{r_2}) - \alpha_{1,2}D_{l_1}(\rho_{l_1})\}\big\}, \\
f_{r_2} &:= f_{l_1,r_2} + f_{l_2,r_2}, \\
f_{l_1} &:= f_{l_1,r_1} + f_{l_1,r_2}, \\
f_{l_2} &:= f_{l_2,r_1} + f_{l_2,r_2}.
\end{aligned} \tag{4.22}$$

As before, f_{l_i,r_j} denotes the flux from incoming road l_i to outgoing road r_j, f_{l_i} denotes the total amount of traffic exiting incoming road l_1 and f_{r_j} denotes the total amount of traffic entering outgoing road r_j. We will simplify the notation slightly by defining $f_{i,j} := f_{l_i,r_j}$. To avoid confusion between incoming and outgoing roads, we will still keep f_{l_i} and f_{r_j} as before.

The definition of the fluxes (4.22) is equivalent to considering the 2-to-2 junction as two independent 2-to-1 junctions, and solving for each of them separately. This might seem like a reasonable approach, and indeed it is the idea we used for the 1-to-2 junction. However, the introduction of a second outgoing road complicates matters, and this naive approach might not work in all cases. The main effect being neglected is that of crossing traffic from the other connection. We will model this effect in a new way.

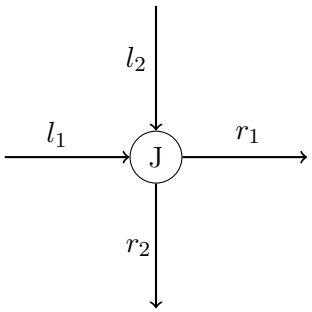


Figure 4.12: 2-to-2 junction with potential crossing traffic.

To understand the effect of crossing traffic, we consider a junction as in Figure 4.12, where we assume that all traffic from road l_1 goes to road r_1, and likewise, all traffic from road l_2 goes to road r_2. In this case, it is clear that the traffic from road l_1 crosses the traffic from road l_2. We now assume that road l_2 has duty to give way to traffic on road l_1. Thus, whenever there is heavy traffic on the connection (l_1, r_1), the traffic should be reduced on connection (l_2, r_2). However, this reduction is not captured if the fluxes are calculated according to (4.22).

We extend the model to also account for the effect of crossing connections. The idea is to first calculate the flux across the connection with higher priority, and then to use this flux to calculate some upper bound on the flux of the connection with lower priority. One way to do this is to define an upper bound $f_{2,2}^{\text{upper}} = g_{2,2}(f_{1,1}, \rho_{l_1})$ for some function $g_{2,2}$ that depends on the flow on the connection with higher priority $f_{1,1}$. Assuming for the moment that this upper bound is known, we may modify the fluxes in (4.22) as

$$\begin{aligned}
f_{1,1} &:= \min\{\alpha_{1,1} D_{l_1}(\rho_{l_1}), \max\{P_1 S_{r_1}(\rho_{r_1}), S_{r_1}(\rho_{r_1}) - \alpha_{2,1} D_{l_2}(\rho_{l_2})\}\}, \\
f_{2,1} &:= \min\{\alpha_{2,1} D_{l_2}(\rho_{l_2}), \max\{(1-P_1) S_{r_1}(\rho_{r_1}), S_{r_1}(\rho_{r_1}) - \alpha_{1,1} D_{l_1}(\rho_{l_1})\}\}, \\
f_{r_1} &:= f_{1,1} + f_{2,1}, \\
f_{1,2} &:= \min\{\alpha_{1,2} D_{l_1}(\rho_{l_1}), \max\{P_2 S_{r_2}(\rho_{r_2}), S_{r_2}(\rho_{r_2}) - \alpha_{2,2} D_{l_2}(\rho_{l_2}), S_{r_2}(\rho_{r_2}) - f_{2,2}^{\text{upper}}\}\}, \\
f_{2,2} &:= \min\{f_{2,2}^{\text{upper}}, \alpha_{2,2} D_{l_2}(\rho_{l_2}), \max\{(1-P_2) S_{r_2}(\rho_{r_2}), S_{r_2}(\rho_{r_2}) - \alpha_{1,2} D_{l_1}(\rho_{l_1})\}\}, \\
f_{r_2} &:= f_{1,2} + f_{2,2}, \\
f_{l_1} &:= f_{1,1} + f_{1,2}, \\
f_{l_2} &:= f_{2,1} + f_{2,2}.
\end{aligned} \tag{4.23}$$

The fluxes $(f_{1,1}, f_{2,1}, f_{r_1})$ and $(f_{1,2}, f_{2,2}, f_{r_2})$ are here calculated separately as two 2-to-1 junctions, with an extra upper bound on the flux $f_{1,2}$. From Section 4.1.3 we know that fluxes calculated in this way will satisfy Rules (A) and (B), while at the same time approximate Rule (C). We note that separating the flux f_{l_i} into the two components $f_{i,1}$ and $f_{i,2}$ is only allowed when using a non-FIFO rule. Otherwise we would not be able to maximize them separately.

To simplify the notation slightly, we can introduce

$$f_{2,2}^{\max} := \min\{f_{2,2}^{\text{upper}}, \alpha_{2,2} D_{l_2}(\rho_{l_2})\},$$

and we can simplify (4.23) as

$$\begin{aligned}
f_{1,1} &:= \min\{\alpha_{1,1} D_{l_1}(\rho_{l_1}), \max\{P_1 S_{r_1}(\rho_{r_1}), S_{r_1}(\rho_{r_1}) - \alpha_{2,1} D_{l_2}(\rho_{l_2})\}\},\\
f_{2,1} &:= \min\{\alpha_{2,1} D_{l_2}(\rho_{l_2}), \max\{(1-P_1) S_{r_1}(\rho_{r_1}), S_{r_1}(\rho_{r_1}) - \alpha_{1,1} D_{l_1}(\rho_{l_1})\}\},\\
f_{r_1} &:= f_{1,1} + f_{2,1},\\
f_{1,2} &:= \min\{\alpha_{1,2} D_{l_1}(\rho_{l_1}), \max\{P_2 S_{r_2}(\rho_{r_2}), S_{r_2}(\rho_{r_2}) - f_{2,2}^{\max}\}\},\\
f_{2,2} &:= \min\{f_{2,2}^{\max}, \max\{(1-P_2) S_{r_2}(\rho_{r_2}), S_{r_2}(\rho_{r_2}) - \alpha_{1,2} D_{l_1}(\rho_{l_1})\}\},\\
f_{r_2} &:= f_{1,2} + f_{2,2},\\
f_{l_1} &:= f_{1,1} + f_{1,2},\\
f_{l_2} &:= f_{2,1} + f_{2,2}.
\end{aligned} \tag{4.24}$$

We have yet to discuss how the upper bound $f_{2,2}^{\text{upper}}$ can be determined. Hence, with the aim of determining a suitable $g_{2,2}$, we first make some simple observations. First, if there is no traffic on the connection with right of way, all of the traffic should be allowed to cross. Thus the upper bound should simply be $f_{2,2}^{\text{upper}} = f_{l_2}(\sigma_{l_2})$, i.e., the upper limit should be the maximal allowed flux. On the other hand, if there is very heavy traffic, almost no flow should be allowed to cross the junction. In the extreme case, when $f_{1,1} = f_{l_1}(\sigma_{l_1})$, one can either put $f_{2,2}^{\text{upper}} = 0$, or $f_{2,2}^{\text{upper}} = \epsilon$ for some $0 < \epsilon \ll 1$ to allow a small flow even when the road is fully congested. In this thesis, we assume that $f_{2,2}^{\text{upper}} > 0$. Making use of these observations, one feasible function $g_{2,2}$ could be a linear interpolation between the end points, that is

$$g_{2,2}(f_{1,1}) = f_{l_2}(\sigma_{l_2}) - \frac{f_{1,1}}{f_{l_1}(\sigma_{l_1})} \cdot (f_{l_2}(\sigma_{l_2}) - \epsilon). \tag{4.25}$$

Here, $g_{2,2}$ is a strictly decreasing function of the flux $f_{1,1}$. It is possible to consider a more complex function $g_{2,2}$ by i.e., considering a dependency also on the density on road l_1. However, we will in this thesis stick to the simple 4.25. Furthermore, it is probably reasonable to let ϵ depend on the maximum flux, i.e., $\epsilon = \delta f_{l_2}(\sigma_{l_2})$, where $0 < \delta \ll 1$ is a small positive number.

Finally, we write the numerical scheme for the ghost cells as

$$\begin{aligned}
\rho_{l_1,k_{l_1}}^{n+1} &= \rho_{l_1,k_{l_1}}^{n} - \lambda^n [f_{l_1} - F_{l_1,k_{l_1}-1/2}^{n}],\\
\rho_{l_2,k_{l_2}}^{n+1} &= \rho_{l_2,k_{l_2}}^{n} - \lambda^n [f_{l_2} - F_{l_2,k_{l_2}-1/2}^{n}],\\
\rho_{r_1,k_{r_1}}^{n+1} &= \rho_{r_1,k_{r_1}}^{n} - \lambda^n [F_{r_1,k_{r_1}+1/2}^{n} - f_{r_1}],\\
\rho_{r_2,k_{r_2}}^{n+1} &= \rho_{r_2,k_{r_2}}^{n} - \lambda^n [F_{r_2,k_{r_2}+1/2}^{n} - f_{r_2}].
\end{aligned} \tag{4.26}$$

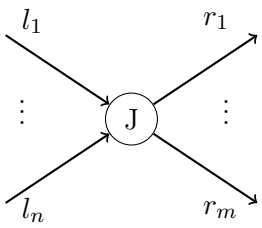


Figure 4.13: n-to-m junction

4.1.5 n-to-m Junction

After having investigated smaller junctions in detail, we now generalize our ideas to an n-to-m junction. To this end, we will need to generalize both the new way of handling merging traffic and the new way of handling yield rules to larger network. As before, we denote the incoming roads by $l_i, i = 1, \dots, n$ and the outgoing road by $r_j, j = 1, \dots, m$, and we do not assume that $n = m$. A visualization of this network can be seen in Figure 4.13. In the general junction, a matrix of distribution parameters, A, specifying how the traffic is distributed across the junction is needed. We write this matrix as

$$A = \begin{bmatrix} \alpha_{1,1} & \cdots & \alpha_{1,m} \\ \vdots & \ddots & \vdots \\ \alpha_{n,1} & \cdots & \alpha_{n,m} \end{bmatrix}.$$

In addition, we need right-of-way parameters, specifying what percentage of traffic is coming from the different roads. In the previous cases it was enough with a single P for each outgoing road, but when there are more than two incoming roads, this approach will no longer work. Instead, similarly to the distribution matrix A, we will define a priority matrix

$$P = \begin{bmatrix} \beta_{1,1} & \cdots & \beta_{1,m} \\ \vdots & \ddots & \vdots \\ \beta_{n,1} & \cdots & \beta_{n,m} \end{bmatrix},$$

where for each outgoing road j, the parameter $\beta_{i,j}$ denotes the percentage of traffic coming from road i. For the percentages to add up to 1, we impose the condition of unit column sums for P

$$\sum_{i=1}^{n} \beta_{i,j} = 1 \quad \forall j \in \{1, \dots, m\}.$$

We will come back to how these parameters can be determined from the densities of the other roads, generalizing the idea from Section 4.1.3. For the distribution matrix, the condition of unit rowsums still holds

$$\sum_{j=1}^{m} \alpha_{i,j} = 1 \quad \forall i \in \{1, \dots, n\}.$$

In addition to these parameters, we need to know which other connections with higher priorities that are being crossed for each pair of incoming road l_i and outgoing road r_i. This needs to be specified for each junction to be able to create a realistic model. For simplicity, we will in this section assume that the roads of the network are ordered by their priority. That is, we assume

$$\beta_{1,j} > \beta_{2,j} > \cdots > \beta_{n,j}, \quad \forall j \in \{1, \ldots, m\}.$$

This assumption is made to simplify the notation in this section.

We want to generalize our idea of having the right-of-way parameters depend on the densities of the other incoming roads. To this end, for each outgoing road r_j we introduce the function

$$h_{n,j} : ([0,1])^{n-1} \to ([0,1])^n,$$

and we choose the right-of-way parameters as

$$(\beta_{1,j}, \ldots, \beta_{n,j}) = h_{n,j}(\alpha_{1,j}\rho_{l_1}, \ldots, \alpha_{n-1,j}\rho_{l_{n-1}}).$$

Making use of the priority function h defined in Section 4.1.3, one possible choice for $h_{n,j}$ is

$$h_{n,j}(\alpha_{1,j}\rho_{l_1}, \ldots, \alpha_{n-1,j}\rho_{l_{n-1}}) = (\nu_{1,j}, \ldots, \nu_{n,j}), \tag{4.27}$$

where $\nu_{i,j}$ is defined recursively

$$\begin{aligned}
\nu_{1,j} &= h(\alpha_{1,j}\rho_{l_1}) \\
\nu_{i,j} &= \Big(1 - \sum_{l=1}^{i-1} \nu_{j,l}\Big) \cdot h(\alpha_{i,j}\rho_{l_i}), \qquad i = 2, \ldots, n-1 \\
\nu_{n,j} &= 1 - \sum_{i=1}^{n-1} \nu_{j,i}.
\end{aligned}$$

In some cases, there will be no flux between incoming road l_i and outgoing road r_j. This could, e.g., be a junction where a u-turn is illegal. In such a junction, we only need $n-1$ right-of-way parameters. If more crossings are illegal, even fewer right-of-way parameters are necessary. Denoting by $\mathcal{I}_{\text{i-c}}$ the illegal crossings, we can update function $h_{n,j}$ as follows

$$h_{n,j}(\alpha_{1,j}\rho_{l_1}, \ldots, \alpha_{n-1,j}\rho_{l_{n-1}}) = (\tilde{\nu}_{1,j}, \ldots, \tilde{\nu}_{n,j}). \tag{4.28}$$

with

$$\tilde{\nu}_{1,j} = \begin{cases} h(\alpha_{1,j}\rho_{l_1}) & \text{if } j \notin \mathcal{I}_{\text{i-c}}, \\ 0 & \text{if } j \in \mathcal{I}_{\text{i-c}}, \end{cases} \tag{4.29}$$

and

$$\tilde{\nu}_{i,j} = \begin{cases} (1 - \sum_{l=1}^{i-1} \tilde{\nu}_{l,j}) \cdot h(\alpha_{1,j}\rho_{l_1}) & \text{if } j \notin \mathcal{I}_{\text{i-c}}, \\ 0 & \text{if } j \in \mathcal{I}_{\text{i-c}}, \end{cases} \quad \text{if } i < i_{\text{final}}, \tag{4.30}$$

where $i_{\text{final}} = \max\{i \in \{1, \dots, n\} : i \notin \mathcal{I}_{\text{i-c}}\}$. Finally, we set

$$\tilde{\nu}_{i_{\text{final}},j} = 1 - \sum_{l=1}^{i_{\text{final}}} \tilde{\nu}_{i,j}.$$

Constructing the right-of-way parameters in such a way ensures that $\tilde{\nu}_{i,j} \in (0, 1)$ for all i, meaning that there will always be some flow from all of the roads (except for illegal connections). In addition, this construction ensures that $\sum_{i=1}^{n} \tilde{\nu}_{i,j} = 1$ for all $j \in \{1, \dots, m\}$, and so we may set $\beta_{i,j} = \tilde{\nu}_{i,j}$, ensuring that the conditions on the right-of-way parameters are satisfied.

Now, we need to introduce a way of calculating an upper bound on the flux from road l_i to road r_j based on the connections of higher priority that it crosses. In Section 4.1.4 we calculated an upper bound in the case of one crossing connection, but for a general connection, we may have up to $(n-1) \cdot (m-1)$ crossing connections with higher priority. To the best of our knowledge, this situation has not been investigated previously for a macroscopic traffic model. To develop a new way of calculating an upper bound, we start from a few simple observations.

- When there is no traffic on any of the crossing connections, the upper bound should simply be equal to the maximal possible flux.
- On the other hand, if one or more of the crossing connections are fully congested, almost no traffic should be allowed to enter the junction.
- If there is only traffic on one of the crossing connections, the upper bound should be equivalent to the one calculated in Section 4.1.4.

The main remaining question is how to calculate the upper bound when there is traffic on several of the crossing connections and none of the connections are fully congested. In this case, the contributions from the different crossing connections need to be combined in a way that makes sense physically.

Here, different ideas are possible. One approach is to simply sum the contributions from each of the crossing connections, resulting in the upper bound

$$f_{i,j}^{\text{upper}} = f_i(\sigma_i) - \min\Big\{1, \sum_{(l,k) \in \mathcal{C}_{i,j}^{\text{crossing}}} \frac{f_{l,k}}{f_l(\sigma_l)}\Big\} \left(f_i(\sigma_i) - \epsilon\right), \tag{4.31}$$

where $\mathcal{C}_{i,j}^{\text{crossing}}$ is the set of crossing connections with higher priorities. We recall from previous sections that $f_{i,j}$ was defined to be the flux of traffic between incoming road l_i and outgoing road r_j. At this point, we note that $f_{l,k}$ needs to be known for $(l,k) \in \mathcal{C}_{i,j}^{\text{crossing}}$ for it to be possible to calculate the upper bound.

A different approach is to take the maximum contribution from the crossing connections, giving the upper bound

$$f_{i,j}^{\text{upper}} = f_i(\sigma_i) - \max_{(l,k) \in \mathcal{C}_{i,j}^{\text{crossing}}} \left\{ \frac{f_{l,k}}{f_l(\sigma_l)} \right\} \left(f_i(\sigma_i) - \epsilon\right). \tag{4.32}$$

The two different approaches have different physical interpretations. The first case is the macroscopic analogy of the case where vehicles entering from the different crossing connections are all coming into the junction at different times. The second case is the macroscopic analogy of the case where vehicles entering from the different crossing connections are all coming into the junction at the same time. Hence, both of these approaches seem to model edge cases, and do not seem to be very realistic representations of how traffic actually behaves. In the real world, one would expect that some vehicles from different roads enter at the same time, while others enter at different times.

To model this in the macroscopic model, the calculation of the upper bound needs to be modified. We aim to find a parameter

$$\xi_{i,j} \in \left[\max_{(l,k)\in\mathcal{C}_{i,j}^{\text{crossing}}} \left\{ \frac{f_{l,k}}{f_l(\sigma_l)} \right\}, \min\Big\{1, \sum_{(l,k)\in\mathcal{C}_{i,j}^{\text{crossing}}} \frac{f_{l,k}}{f_l(\sigma_l)} \Big\} \right]$$

that in some sense model vehicles potentially arriving at different, or random, times.

Suppose for the moment that the parameter $\xi_{i,j}$ is known, and that it reflects that vehicles from different roads may arrive at random times. We then calculate the upper bound as

$$f_{i,j}^{\text{upper}} = \epsilon + \xi_{i,j}(f_i(\sigma_i) - \epsilon). \tag{4.33}$$

We will now discuss a way of calculating ξ that is physically meaningful. This discussion is rather involved, and is one the most important contributions of this work. We know that each fraction $f_{l,k}/f_l(\sigma_l)$ lies in the interval $[0, 1]$. Each of these fractions gives a measure of how congested this connection is. If $f_{l,k}/f_l(\sigma_l) = 1$, connection (l, k) is fully congested. If, on the other hand $f_{l,k}/f_l(\sigma_l) = 0$, there are no vehicles crossing connection (l, k).

Instead of considering each of the fractions by a number, we represent it by a stick, $s_{l,k}$, of length $d_{l,k} = f_{l,k}/f_l(\sigma_l)$. Now imagine that we place this stick somewhere on the unit interval. That is, we have for the left edge of the stick $s_{l,k}^{\text{left}} \in [0, 1 - d_{l,k}]$, and for the right edge of the stick $s_{l,k}^{\text{right}} \in [d_{l,k}, 1]$. Now, instead of trying to figure out how congested the junction is by making use of the fractions, we will solve the related problem of finding out how much of the interval $[0, 1]$ is covered by $m_{i,j}$ such sticks, where $m_{i,j} = |\mathcal{C}_{i,j}^{\text{crossing}}|$ is the cardinality of the set of crossing connections, or the number of connections that are being crossed. This calculation will be performed by letting the positions of each stick be randomly distributed on the unit interval, before taking the expectation of the total length covered by the sticks.

Before doing the calculating for a general number of these sticks, we will look at a simpler example to get a clearer understanding of the underlying idea. We consider a road network represented by Figure 4.14. This network consists of three incoming roads l_1, l_2 and l_3 and three outgoing roads r_1, r_2

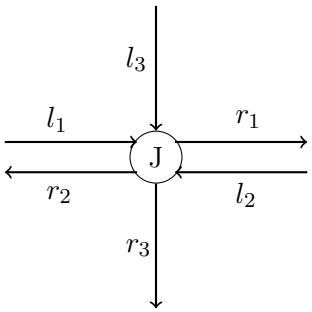


Figure 4.14: 3-to-3 junction with two lanes of being crossed.

and r_3. We assume for simplicity that all the traffic from road l_i enters road r_i and that the traffic from l_3 has duty to give way to the two other roads. In this case, we have two connections with higher priority being crossed, and we need to figure out how to combine their contributions. We assume that we know the amount of traffic going from road l_1 to road r_1 and the amount of traffic going from road l_2 to road r_2. Denoting the fluxes by $f_{1,1}$ and $f_{2,2}$, we calculate the percentage of the maximal fluxes that crosses the junction $f_{1,1}/f_1(\sigma_1)$ and $f_{2,2}/f_2(\sigma_2)$. We suppose that these fractions are calculated as i.e.,

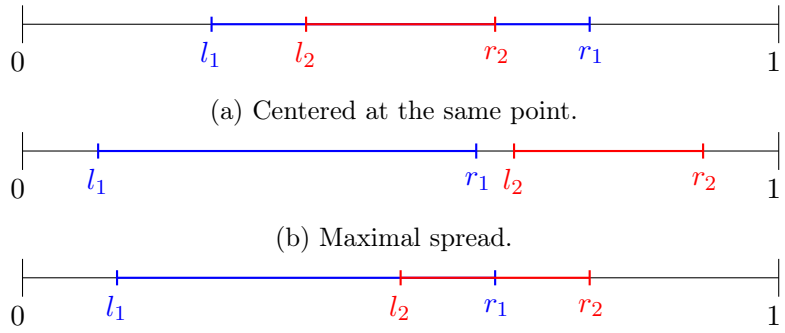
$$\frac{f_{1,1}}{f_1(\sigma_1)} = 0.5 \qquad \text{and} \qquad \frac{f_{2,2}}{f_2(\sigma_2)} = 0.25.$$

With the aim of finding the combined contribution of these two roads, we represent these fractions by two sticks with lengths $d_{1,1} = 0.5$ and $d_{2,2} = 0.25$ being placed somewhere on the unit interval. Figure 4.15 displays different ways two such sticks can be placed on the unit interval.

In Figure 4.15a, the two sticks are placed with their centers at the same point. The total length of the region covered by at least one of these sticks will in this case be equal to the length of the longest stick. This is analogous to choosing $\xi_{i,j}$ according to (4.32), where the maximal percentage of flux on a crossing connection was used. A physical interpretation of this case is that whenever a vehicle from road l_2 enters the junction, there is also a vehicle from road l_1 that is entering the junction.

In Figure 4.15b, the sticks are placed with as little overlap as possible. This is analogous to choosing $\xi_{i,j}$ according to (4.31), where the sum of the fluxes on the crossing connections was used. A physical interpretation of this case, is that whenever a vehicle from road l_2 enters the junction, there are no vehicles from l_1 entering the junction.

Finally, Figure 4.15c shows a case where there is some overlap between the two sticks, but they are not centered at the same point as in Figure 4.15a. A physical interpretation of this case is that for every vehicle entering the junction from road l_2, there is some probability $0 < p < 1$ that a vehicle also enters from road l_1. This case seems to be the closest representation of actual traffic, but we still need to determine exactly how much overlap between two



(a) Centered at the same point.

(b) Maximal spread.

(c) Some overlap.

Figure 4.15: The figure shows different ways two sticks of length 0.5 and 0.25 can be distributed on the unit interval. In all figures, l_1, l_2 denotes the left edges and r_1, r_2 denotes the right edges of the two sticks respectively. Figure 4.15a shows the case where both sticks are centered at 0.5, Figure 4.15b shows a case where there is no overlapping regions between the two sticks, and Figure 4.15c show a case where there is some overlap between the two sticks, but they are not centered at the same point.

(or more) such sticks is expected. We will now give a precise way to calculate the combined contribution from multiple connections being crossed.

We return to the general case of $m_{i,j}$ connections with higher priorities being crossed by connection (i, j). We collect all of these connection in the set $\mathcal{C}_{i,j}^{\text{crossing}}$. Assuming that the fraction $d_{l,k} = f_{l,k}/f_l(\sigma_l)$ for all $(l, k) \in \mathcal{C}_{i,j}^{\text{crossing}}$ is known for all of the connections, we represent the fractions by the sticks $s_{l,k} \, \forall \mathcal{C}_{i,j}^{\text{crossing}}$ with lengths $d_{l,k}$ and assume that they are placed somewhere on the unit interval. With the aim of finding the expected length of the portion of the unit interval covered by at least one of the sticks, we start by assuming that the left edge of each stick $s_{l,k}$ is uniformly distributed in the interval $[0, 1 - d_{l,k}]$. By distributing the sticks in this way, we aim to model that drivers can enter the junction at random times. Aiming to find the combined coverage of the sticks, we start by considering a point $x \in [0, 1]$ and try to find the probability that x is covered by a stick $s_{l,k}$ with length $d_{l,k}$ with a uniformly distributed left edge. We split the probability into two cases, namely $d_{l,k} < 0.5$ and $d_{l,k} \geq 0.5$.

Assume for the moment that $d_{l,k} < 0.5$. Then the probability of $x \in [0, 1]$ being covered by a stick $s_{l,k}$ with left end-point of stick chosen uniformly in $[0, 1 - d_{l,k}]$ is given by

$$p_1(x; d_{l,k}) = \begin{cases} \frac{x}{1-d_{l,k}} & \text{if } x < d_{l,k}, \\ \frac{d_{l,k}}{1-d_{l,k}} & \text{if } d_{l,k} < x < 1 - d_{l,k}, \\ \frac{1-x}{1-d_{l,k}} & \text{if } 1 - d_{l,k} < x < 1. \end{cases} \tag{4.34}$$

If we instead assume that $d_{l,k} \geq 0.5$, the situation is slightly different. When

the stick is larger than half of the interval, there is a region in the middle of the interval that will always be covered by the stick, no matter where it is placed. The length of this region will be equal to $2 \cdot (d_{l,k} - 0.5)$. We write the probability function in this case as

$$p_2(x; d_{l,k}) = \begin{cases} \frac{x}{1-d_{l,k}} & \text{if } x < 1 - d_{l,k}, \\ 1 & \text{if } 1 - d_{l,k} < x < d_{l,k}, \\ \frac{1-x}{1-d_{l,k}} & \text{if } d_{l,k} < x < 1. \end{cases} \tag{4.35}$$

Thus, for a general stick $s_{l,k}$ with length $d_{l,k}$ we write the probability of $x \in [0, 1]$ being covered by a stick $s_{l,k}$ as

$$p(x; d_{l,k}) = \begin{cases} p_1(x; d_{l,k}) & \text{if } d_{l,k} < 0.5, \\ p_2(x; d_{l,k}) & \text{otherwise.} \end{cases} \tag{4.36}$$

Now that we know the probability of a point $x \in [0, 1]$ being covered by a stick $s_{l,k}$, we can calculate the probability of $x \in [0, 1]$ being covered by *at least* one stick using the complement of it being covered by no sticks. Thus, given a collection of sticks $s \in S_{i,j}$, the probability of a point x not being covered by any sticks is equal to

$$P(\text{x not covered by any sticks}) = \prod_{s \in S_{i,j}} \big(1 - p(x; d)\big),$$

where d is the distance length of stick s. Using the complement of this probability, we obtain the probability of x being covered by at least one stick

$$P(\text{x covered by at least one stick}) = 1 - \prod_{s \in S_{i,j}} \big(1 - p(x; d)\big).$$

We now introduce the function $H(x, w)$ as the indicator for the event that at least one stick $s \in S_{i,j}$ covers x. The combined length of all regions being covered by at least one stick will then be

$$L(w) = \int_{x=0}^{1} H(x, w) dx. \tag{4.37}$$

Next, we must eliminate the randomness we have introduced to the model. One possible approach is to run the simulation a large number of times. For each run, at each junction and at each time step, the position of the sticks can be sampled from its probability distribution, followed by calculating the integral (4.37). Finally, one could average over all of the simulations to obtain an approximation of the expectation. The drawback of this approach is that it requires running the full simulation a large number of times.

Instead, what we will do is to take the expectation over each junction at each time step. Taking the expectation of the integral (4.37) gives

$$E[L(w)] = \int_{x=0}^{1} 1 - \prod_{s \in S_{i,j}} \big(1 - p(x;d)\big) dx. \tag{4.38}$$

When choosing $\xi_{i,j}$, we are interested in the region that is not covered by any of the sticks, which is analogous to the fraction of times no vehicles are occupying the junction. We therefore set the value of $\xi_{i,j}$ as

$$\begin{aligned} \xi_{i,j} = 1 - E[L(w)] &= 1 - \int_{x=0}^{1} 1 - \prod_{s \in S_{i,j}} \big(1 - p(x;d)\big) dx \\ &= \int_{x=0}^{1} \prod_{s \in S_{i,j}} \big(1 - p(x;d)\big) dx, \end{aligned} \tag{4.39}$$

and plug this value in when calculating the upper bound. In total we then have

$$f_{i,j}^{\text{upper}} = \epsilon + \Big(\int_{x=0}^{1} \prod_{s \in S_{i,j}} \big(1 - p(x;d)\big) dx \Big) \cdot \big(f_i(\sigma_i) - \epsilon \big). \tag{4.40}$$

This approach entails having to evaluate this integral, most likely numerically, at every time step for all connections in the network having crossing connections of higher priority. In Appendix A.2, this integral is calculated exactly for some cases. However, we have not calculated this integral in the general case, so some quadrature rule like the trapezoidal rule or Simpson's rule is needed to approximate the integral. We have included a brief discussion of the trapezoidal rule applied to this integral in Appendix A.2.3

Having established ways of determining both the right-of-way parameter for each edge (i,j), as well as the upper bound $f_{i,j}^{\text{upper}}$, we are ready to define the fluxes. We try to generalize the fluxes defined in (4.24), and will later check if this definition actually satisfies Rules (A) and (B), while approximating Rule (C). We define the fluxes as

$$\begin{aligned} f_{l_i,r_j} &= \min\Big\{ f_{i,j}^{\max}, \max\Big\{ \beta_{i,j} S_{r_j}, S_{r_j} - \sum_{k=1, k\neq i}^{n} f_{l,k}^{\max} \Big\} \Big\}, \\ f_{l_i} &= \sum_{j=1}^{m} f_{l_i,r_j}, \text{ for } i = 1, \dots, n, \\ f_{r_j} &= \sum_{i=1}^{n} f_{l_i,r_j}, \text{ for } j = 1, \dots, m, \end{aligned} \tag{4.41}$$

where $f_{i,j}^{\max} := \min\{f_{i,j}^{\text{upper}}, \alpha_{i,j} D_i(\rho_i)\}$. If connection (i,j) does not cross any connections having higher priority, the upper bound $f_{i,j}^{\text{upper}}$ will not be calculated, and we simply set $f_{i,j}^{\max} = \alpha_{i,j} D_i(\rho_i)$. It is important to note that the

upper bound $f_{i,j}^{\text{upper}}$ may depend on several of the other fluxes. However, there will be at the very least one connection that does not cross other connections. Thus, when calculating the fluxes, it is important to calculate them in the correct order, so that all necessary quantities are defined. Finally we write the numerical scheme for the edge nodes

$$\begin{aligned}\rho_{l_i,k_{l_i}}^{n+1} &= \rho_{l_i,k_{l_i}}^{n} - \lambda^n[f_{l_i} - F_{l_i,k_{l_i}-1/2}^{n}], \text{ for } i = 1,\dots,n,\\ \rho_{r_j,k_{r_j}}^{n+1} &= \rho_{r_j,k_{r_j}}^{n} - \lambda^n[F_{r_j,k_{r_j}+1/2}^{n} - f_{r_j}], \text{ for } j = 1,\dots,m.\end{aligned} \tag{4.42}$$

The flux definition given in (4.41) was given without proof that it actually satisfies Rules (A) and (B), and that it approximates Rule (C). To check whether this is the case or not, we first notice that if we assume that the upper bound $f_{i,j}^{\text{upper}}$ is known for all connections (l_i, r_j), the fluxes in (4.41) correspond to m different independent n-to-1 junctions. Thus, if we can show that the optimal value is attained for one such n-to-1 junction arbitrarily, then we know that it will also be satisfied by all the other junctions. Further, since we are using a FIFO rule, maximizing the flow on each n-to-1 junction is equivalent to maximizing the flow on the full n-to-m junction.

We consider the n-to-1 junction leading into outgoing road r_j, where j is chosen in $\{1,\dots,m\}$ arbitrarily. As before we can split into the supply- and demand-limited cases. If $S_{r_j}(\rho_{r_j}) > \sum_{i=1}^{n} f_{i,j}^{\max}$, the maximal flux will simply be $f_{i,j} = f_{i,j}^{\max}$ for $i = 1,\dots,n$. We note that this is exactly what we get by defining the fluxes $f_{i,j}$ according to (4.41).

Instead, suppose that we are in the supply-limited case, where $S_{r_j}(\rho_{r_j}) < \sum_{i=1}^{n} f_{i,j}^{\max}$. As in the 2-to-1 junction, we now need to determine where the traffic coming into outgoing road r_j is coming from, and we want to respect Rule (C) as far as possible. As for the 2-to-1 junction, we set the outgoing flux equal to the maximum value, i.e., $f_{r_j} = S_{r_j}(\rho_{r_j})$. We also introduce the intersection between the feasible region and the maximization of the flux by the convex set $K \subset \mathbb{R}^n$ given by

$$K = \Big\{(f_{1,j},\dots,f_{n,j}) : \sum_{i=1}^{n} f_{i,j} = S_{r_j}(\rho_{r_j}), 0 \le f_{i,j} \le f_{i,j}^{\max}, i = 1,\dots,n\Big\}.$$

Generalizing Rule (C), and letting C denote the amount of traffic entering road r_j, we want $\beta_{i,j}C$ to come from road l_i. The set of points satisfying this rule can be summarized by the line $r \in \mathbb{R}^n$ given by

$$r : \begin{cases} f_{n,j} = \frac{\beta_{n,j}}{\beta_{1,j}} f_{1,j}, \\ \quad\vdots \\ f_{n,j} = \frac{\beta_{n,j}}{\beta_{n-1,j}} f_{n-1,j}. \end{cases}$$

The unique optimal solution can then be determined as the point $(\hat{f}_{1,j},\cdots,\hat{f}_{n,j}) \in K$ minimizing the distance to the point P_I, where P_I is the intersection between

the line r and the hyperplane given by

$$\sum_{i=1}^{n} f_{i,j} = S_{r_j}(\rho_{r_j});$$

that is, the hyperplane where the flux is maximized. We can now write the optimization problem we are trying to solve, assuming for the moment that the upper bounds $f_{i,j}^{\max}$ are known for all connections.

For any outgoing road r_j, we calculate the fluxes leading into this road using the following steps. If

$$\sum_{i=1}^{n} f_{i,j}^{\max} \leq S_{r_j}(\rho_{r_j}),$$

the optimal solution is given by

$$f_{i,j} = f_{i,j}^{\max}, \forall i \in \{1, \ldots, n\}.$$

If, on the other hand,

$$\sum_{i=1}^{n} f_{i,j}^{\max} > S_{r_j}(\rho_{r_j}),$$

the optimal solution is determined by the unique point $(\hat{f}_{1,j}, \cdots, \hat{f}_{n,j}) \in K$ satisfying

$$(\hat{f}_{1,j}, \cdots, \hat{f}_{n,j}) = \arg\min_{P \in K} d(P, P_I), \tag{4.43}$$

where $d(\cdot, \cdot)$ is the usual Euclidean distance.

The intersection point P_I can be calculated as $P_I = (S_r\beta_{1,j}, \ldots, S_r\beta_{n,j})$. Clearly, this point lies on the hyperplane maximizing the flux, since we by construction have

$$\sum_{i=1}^{n} S_r\beta_{i,j} = S_r \sum_{i=1}^{n} \beta_{i,j} = S_r.$$

In addition, we see that the intersection point lies on the line r, since

$$\begin{aligned} & f_{i,j} = S_r\beta_{i,j} \text{ and } f_{n,j} = S_r\beta_{n,j} \\ \implies f_{n,j} &= \frac{\beta_{n,j}}{\beta_{i,j}} f_{i,j}, \quad \forall i \in \{1, \ldots, n-1\}. \end{aligned} \tag{4.44}$$

Appendix A.1 describes the calculation of the optimal solution of (4.43) to check whether the form (4.41) gives correct results or not. Even if the upper bounds on all of the fluxes are known, (4.41) leads to non-optimal fluxes in some cases. It is reasonable to assume that this is the case also for m-to-1 junctions with $m > 3$. However, when $m > 3$, the minimization problem (4.43) becomes increasingly complex, and finding the optimal solution becomes more difficult. Since such a minimization problem would have to be solved for

each junction at each time step, the accumulated time solving (4.43) becomes very high for larger networks and longer simulations. Therefore, an explicit, closed-form approximation such as (4.41) saves a lot of computational effort. In addition, we know that (4.41) solves the 2-to-1 junction exactly for all cases, and 3-to-1 junction for many cases. Therefore, we hope that it is a reasonable approximation even for more complicated junctions, and we will make use of it in the simulation because of its savings in computational effort.

In the analysis of the exact solution so far, we have assumed that the upper bounds on the fluxes were known, which will not be the case when starting to solve the junction problem. Even for the fluxes that do not cross other connections, the upper bounds appear in (4.41). Since these upper bounds have not been refined yet, some of them might be initialized as being too high. Since the upper bound of all other connection appear with a negative sign in the expression for the fluxes on higher priority connections, choosing them too high may make the fluxes on the higher priority roads become too small. This, in turn, might make the upper bounds that depend on the higher priority fluxes become too high, so that we get an incorrect distribution of the capacity among the incoming roads. Although it is clear that when the upper bounds are not known, the underlying maximization problem becomes much more complicated, we will not go into further details on this problem here. In our simulation, we still make use of (4.41) because of its low computational cost, keeping in mind that the calculated fluxes will not solve the maximization problem exactly in all cases.

4.2 Traffic Lights*

A common feature of junctions that we want to include in our model is traffic lights. A traffic light controls parts or all of the flow of traffic in a junction, and can either be red, yellow or green. A junction may consists of multiple traffic lights that depend on each other. Although such dependencies may in general be complicated, we only consider the case where at most one of two traffic lights may be green at all times. That is, before one of the lights can turn from red to green, it must first wait for the other light to turn red. The inclusions of traffic lights to a macroscopic traffic model is discussed in, e.g., [31], where the changing between green and red light are represented by a finite number of time dependent coefficients. As we will come back to, in order to capture the full dependence on the settings of the traffic lights when computing the gradient, we will model traffic lights differently.

Traffic lights are at any time be in one of three states; they will either be green, yellow or red. The behaviour of a traffic light can therefore be described by a list of times that specify how long each state is stayed in. We will assume that the time it takes for a light to change between green and red and from red to green is the same for all traffic lights in the simulation, and so the list of times only need to specify how long the traffic light should be green or red. We

will also assume that each traffic light follows a repeating cycle. We want each cycle to contain the same number of times for the green and red states. Hence, the simplest such cycle consists only of two times, the times specifying how long the light should be green or red before changing state. In this simplest case, every time the traffic light turns green, it will remain green for the same amount of time. More advanced cycles are possible, allowing the traffic light to be green (or red) for variable amounts of time. The cycle $[60, 40, 30, 50]$ could describe that a traffic light stays green for 60 seconds, before turning red for 40 seconds. After these 40 seconds, the light turns green for 30 seconds before finally turning red for 50 seconds. After these 50 seconds, the cycle repeats.

Different choices for times in the cycle of a traffic light leads to different flows of traffic. Also the delays in public transportation (i.e., busses) travelling across the junction will be impacted by how the traffic light is controlled. We want to determine the best possible cycle either minimizing the overall delay in public transportation, or optimizing the flow of traffic.

As described in Chapter 3, we will later in this thesis evaluate the gradient of such objective functions using automatic differentiation. This in turn means that every part of the simulation needs to be written using differentiable operations that are supported by the AD framework being used. Hence, traffic lights cannot be modelled just by checking the time to see which state the light is in. We instead model the traffic light using a smooth activation function that depends continuously on the cycle times of the traffic light. Then, the incoming flow to a junction can be multiplied by this activation function. The idea that whenever the light is red, there should be approximately zero flux coming into the junction, and so the activation function should be approximately zero. On the other hand, when the light is green, approximately all of the traffic should reach the junction, meaning that the activation function should be close to one.

The change from red to green or green to red is the phase where the traffic light is yellow. Right after the light changes, the drivers will not have enough time to react and so most drivers will keep going. On the other hand, when the light has been yellow for some time, more and more drivers will stop or at least slow down coming into the junction. Therefore it seems reasonable to model the traffic light by some function that is close to zero when the light is red and close to one when the light is green, with some smooth transition between the two states. A final assumption we make on the activation function is that it should be bounded in the interval $[0, 1]$.

Several functions match the desired behaviour of the activation function. One possibility is to define some points where we know what the activation function should be, and then to make use of a suitable smooth interpolating spline. Instead, we will approximate each of the state transitions by a logistic function, one of the most common sigmoid functions. The logistic function is

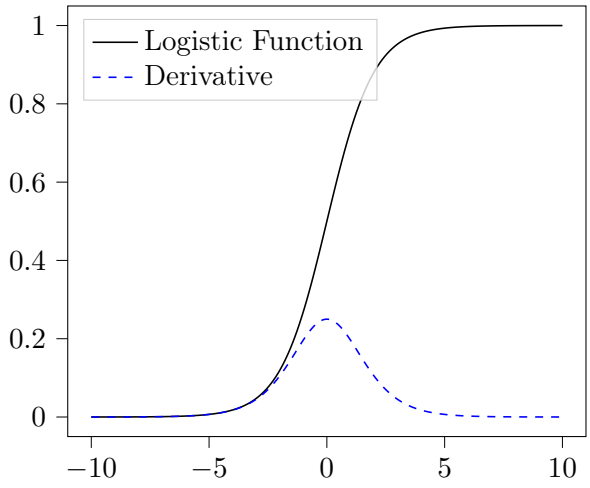


Figure 4.16: The logistic function (4.45) and its derivative.

a bounded differentiable function that is defined as

$$\sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{1+e^x}. \tag{4.45}$$

As it is a sigmoid function, its graph has a characteristic S-shaped curve. A visualization of the logistic function and its derivative can be seen in Figure 4.16.

As the logistic function transitions from being almost zero to being almost one with a smooth transition part, it seems to be a good choice to model a transition from red light to green light. Similarly, a transition from green light to red light can be modelled by the function $1 - \sigma(x)$. The times at which each transition happens is described by the cycle of the traffic light, and to be able to model these transitions at the correct times, we need to shift the logistic function. We first consider a single transition from red to green light. Supposing that the light changes after 20 seconds, we want to model this transition with a logistic function. We also want to be able to control how quickly the light changes from red to green, or more precisely how quickly drivers respond to a change of light from red to green. We assume for now that the transition time is 10 seconds. We combine these assumptions to create a modified logistic function $\tilde{\sigma}(x; t, \delta_t)$ that takes into account the time at which the light changes, as well as a parameter δ_t that controls the length of the transition time. Assuming that $\tilde{\sigma}(x; t, \delta_t)$ can be written on the form

$$\tilde{\sigma}(x; t, \delta_t) = \sigma\big(\alpha(\delta_t)(x - t) - \beta(\delta_t)\big),$$

we aim to determine suitable choices for $\alpha(\delta_t)$ and $\beta(\delta_t)$.

Since the logistic function is symmetric around its center, we know that the modified logistic function should take the value 0.5 at time $t + \delta_t/2$. That

is, we require

$$\tilde{\sigma}(t + \delta_t/2; t, \delta_t) = 0.5. \tag{4.46}$$

In addition, we want the modified logistic function to be almost zero at time t. That is,

$$\tilde{\sigma}(t; t, \delta_t) = \epsilon, \tag{4.47}$$

for some small ϵ, $0 < \epsilon \ll 1$.

Starting from (4.46), we use the definition of the logistic function to get

$$\begin{aligned} \tilde{\sigma}(t + \delta_t/2; t, \delta_t) = \sigma(\alpha(\delta_t)(t + \delta_t/2 - t) - \beta(\delta_t)) &= \frac{1}{2}, \\ \Rightarrow \frac{1}{1 + e^{-\alpha(\delta_t)\delta_t/2 + \beta(\delta_t)}} &= 2, \\ \Rightarrow 2 = 1 + e^{-\alpha(\delta_t)\delta_t/2 + \beta(\delta_t)}&, \\ \Rightarrow -\alpha(\delta_t)\delta_t/2 + \beta(\delta_t) &= 0, \\ \Rightarrow \beta(\delta_t) &= \alpha(\delta_t)\delta_t/2. \end{aligned} \tag{4.48}$$

Similarly, starting from (4.47) we have

$$\begin{aligned} \tilde{\sigma}(t; t, \delta_t) = \sigma(\alpha(\delta_t)(t - t) - \beta(\delta_t)) &= \epsilon, \\ \Rightarrow \frac{1}{1 + e^{\beta(\delta_t)}} &= \frac{1}{\epsilon}, \\ \Rightarrow \frac{1}{\epsilon} = 1 + e^{\beta(\delta_t)}&, \\ \Rightarrow \beta(\delta_t) = \ln(\frac{1}{\epsilon} - 1)&. \end{aligned} \tag{4.49}$$

If we choose $\epsilon = \frac{1}{e^5+1}$, we get $\beta(\delta_t) = 5$, that is $\beta(\delta_t)$ is a constant independent on how long the transition period is. Using $\beta(\delta_t) = 5$ we calculate $\alpha(\delta_t)$ as

$$5 = \alpha(\delta_t)\delta_t/2, \quad \Rightarrow \quad \alpha(\delta_t) = \frac{10}{\delta_t}.$$

Using these values for $\alpha(\delta_t)$ and $\beta(\delta_t)$ we write the modified logistic function as

$$\tilde{\sigma}(x; t, \delta_t) = \sigma\Big(\frac{10}{\delta_t}(x - t) - 5\Big). \tag{4.50}$$

Unless stated otherwise, we will make use of $\delta_t = 10$, and we write

$$\tilde{\sigma}(x; t) = \sigma\big((x - t) - 5\big).$$

We can make use of this function to model a transition between red to green light at time t_1 as $\tilde{\sigma}(x; t_1)$ or a transition from green to red light at time t_2 as $1 - \tilde{\sigma}(x; t_2)$. The natural question at this point is how these transitions can be combined to model a transition from red light to green light at time t_1, followed by a transition back from green light to red light at time t_2, where we assume

$t_2 > t_1 + 10$ so that the traffic light has time to finish the first transition before turning back to red. We try to model this by the simple activation function

$$\gamma(x, t_1, t_2) = \tilde{\sigma}(x; t_1) - \tilde{\sigma}(x; t_2),$$

and check to see if all of the desired properties are satisfied. We first observe that $\tilde{\sigma}(x; t)$ is a strictly increasing function, which implies that

$$\tilde{\sigma}(x; t_1) > \tilde{\sigma}(x; t_2), \quad \forall x \in \mathbb{R}.$$

This, in turn, implies that

$$0 < \gamma(x, t_1, t_2) < \tilde{\sigma}(x; t_1) < 1, \quad \forall x \in \mathbb{R}.$$

Further, since $\tilde{\sigma}(x; t_i)$ changes rapidly only close to t_i, when t_1 and t_2 are sufficiently far apart, the change in $\gamma(x, t_1, t_2)$ close to t_i will be dominated by $\tilde{\sigma}(x; t_i)$. Thus, if $\tilde{\sigma}(x; t_i)$ accurately models the transition at time t_i, then $\gamma(x, t_1, t_2)$ will also accurately model the same transition.

The same idea can be extended to include more than two transitions by defining the activation function as a sum of logistic functions, where each logistic function models one transition. A sequence of transitions determined by the cycle of the traffic light can in this way be modelled by an activation function defined as a sum of logistic functions. We define $s_{\text{start}} \in \{0, 1\}$, where $s_{\text{start}} = 0$ if the traffic light starts as red at time $t = 0$ and $s_{\text{start}} = 1$ if the traffic light starts as green at time $t = 1$. Given a set of times the traffic light switches from red to green T_p and a set of time the traffic light switches from green to red T_m as well as a starting state s_{start}, the total activation function can be defined as

$$\gamma(x; s_{\text{start}}, T_p, T_m) = s_{\text{start}} + \sum_{t_i \in T_p} \tilde{\sigma}(x; t_i) - \sum_{t_j \in T_m} \tilde{\sigma}(x; t_j). \tag{4.51}$$

At this point, it is important to remark that the same transition cannot occur two times in a row: if the traffic light is already green, it cannot make a switch to being green. Hence, for T_m and T_p to be feasible transition times, assuming the times are ordered as $T_m = \{t_1^m, t_2^m, \dots\}$ s.t. $t_1^m < t_2^m < \dots$ and $T_p = \{t_1^p, t_2^p, \dots\}$ s.t. $t_1^p < t_2^p < \dots$, they need to satisfy

$$\begin{aligned} &\forall i \in \mathbb{N},\, 1 \leq i < |T_m|\, \exists j \in \mathbb{N} \text{ s.t. } t_i^m < t_j^p < t_{i+1}^m; \quad t_j^p \in T_p \text{ and } t_i^m, t_{i+1}^m \in T_m, \\ &\forall j \in \mathbb{N},\, 1 \leq j \leq < T_p|\, \exists i \in \mathbb{N} \text{ s.t. } t_j^p < t_i^m < t_{j+1}^p; \quad t_j^p, t_{j+1}^p \in T_p \text{ and } t_i^m \in T_m. \end{aligned} \tag{4.52}$$

This condition states that before and after every transition from red to green light (except potentially the first or last), there must be transitions going from green to red light. As we said, the cycle of a traffic light will be given as a set of times specifying how long the current states should be stayed in, and not a set of times specifying when the transition happens. Supposing that this cycle

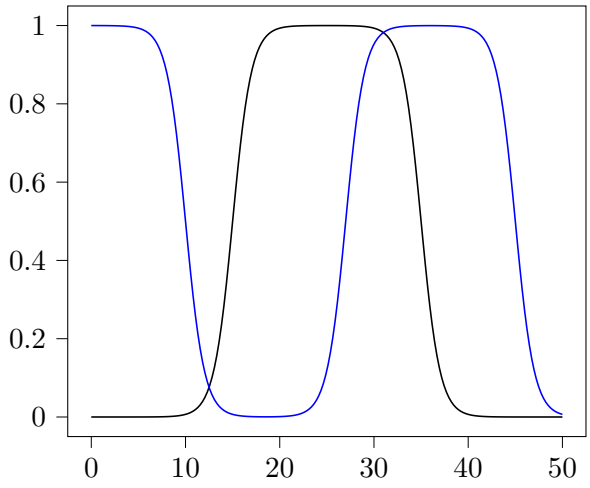


Figure 4.17: Potential activation functions of two different traffic lights.

of times is given by the set T_c ordered as $T_c = \{t_1^c, t_2^c, \dots\}$, we may define the total activation function as

$$\begin{aligned}\gamma(x; s_{\text{start}}, T_c) = s_{\text{start}} + \sum_{t_i \in T_c} ((i + s_{\text{start}}) \bmod 2) \cdot \tilde{\sigma}(x; t_i + \sum_{j<i} t_j) \\ - (1 - ((i + s_{\text{start}}) \bmod 2)) \cdot \tilde{\sigma}(x; t_i + \sum_{j<i} t_j),\end{aligned} \tag{4.53}$$

where mod is the modulus operator, meaning that $a \bmod 2$ is equal to one if a is odd and equal to zero if a is even. Formulated in this way, we immediately guarantee that two of the same transitions cannot happen consequently. Modelling a traffic light using a smooth activation function has not been done before to the best of our knowledge. A visualization of how such activation functions could look like can be seen in Figure 4.17.

As we briefly mentioned in the start of this section, we will also model the simple interaction between two traffic lights so that only one of the two can be green at the same time. We will from this point onwards denote the two traffic lights by traffic light a and traffic light b. Assuming that the period of time where both traffic lights are red is constant, this interaction between traffic lights can be modelled using only one cycle. Now, instead of describing how long a single traffic light remains either green or red, this cycle can describe how long each of the two traffic lights stays green at a time. In order to avoid collisions, after one of the lights turns red, there is a brief period where both traffic lights are red, before the other light can turn green. Assuming that this time period remains constant, the time one of the traffic lights stays red can be determined from the time period the other traffic light stays green and the transition time from green to red. Although, it suffices with one cycle, we still

need two activation functions, one for light a and one for light b. We once again let T_c denote the set of times of the cycle of the coupled traffic light. Then, aiming to write the activation functions in a similar way as in (4.53), we make use of T_c and t_{delay}, the time both lights are red, to formulate the two new sets T_c^a and T_c^b. The set $T_c^k, k = a, b$ will be constructed so that if element t_i^k specifies the time light k should remain green before switching to red, then the element t_{i+1}^k will specify the time light k should remain red before switching back to green. As before, we need a starting state $s_{\text{start}} \in \{0, 1\}$ that specifies what light is green from the beginning. If $s_{\text{start}} = 0$, then light a is green from the start, and if $s_{\text{start}} = 1$, then light b is green from the start. Denoting the time period where both lights are red by t_{delay} we can construct the two sets T_c^a and T_c^b by

$$\begin{aligned} T_c^a &= \{\tilde{t}_i\}, \text{ s.t. } \begin{cases} \tilde{t}_i = t_c^i & \text{if } i \bmod 2 = 1, \\ \tilde{t}_i = t_c^i + t_{\text{delay}} & \text{if } i \bmod 2 = 0, \end{cases} \\ T_c^b &= \{\tilde{t}_i\}, \text{ s.t. } \begin{cases} \tilde{t}_i = t_c^i + t_{\text{delay}} & \text{if } i \bmod 2 = 0, \\ \tilde{t}_i = t_c^i & \text{if } i \bmod 2 = 1, \end{cases} \end{aligned} \tag{4.54}$$

if $s_{\text{start}} = 0$, or by

$$\begin{aligned} T_c^a &= \{\tilde{t}_i\}, \text{ s.t. } \begin{cases} \tilde{t}_i = t_c^i & \text{if } i \bmod 2 = 0, \\ \tilde{t}_i = t_c^i + t_{\text{delay}} & \text{if } i \bmod 2 = 1, \end{cases} \\ T_c^b &= \{\tilde{t}_i\}, \text{ s.t. } \begin{cases} \tilde{t}_i = t_c^i + t_{\text{delay}} & \text{if } i \bmod 2 = 1, \\ \tilde{t}_i = t_c^i & \text{if } i \bmod 2 = 0, \end{cases} \end{aligned} \tag{4.55}$$

if $s_{\text{start}} = 1$. That is, we construct T_c^a and T_c^b by adding the delay times to every second element of the ordered set T_c, starting from either the first or second element. Using the newly constructed sets, we define the two activation functions for the two lights by

$$\begin{aligned} \gamma_a(x; s_{\text{start}}, T_c, t_{\text{delay}}) = 1 - s_{\text{start}} + \sum_{t_i \in T_c} &((i + 1 - s_{\text{start}}) \bmod 2) \cdot \tilde{\sigma}(x; t_i) \\ &- (1 - ((i + 1 - s_{\text{start}}) \bmod 2)) \cdot \tilde{\sigma}(x; t_i), \\ \gamma_b(x; s_{\text{start}}, T_c, t_{\text{delay}}) = s_{\text{start}} + \sum_{t_i \in T_c} &((i + s_{\text{start}}) \bmod 2) \cdot \tilde{\sigma}(x; t_i) \\ &- (1 - ((i + s_{\text{start}}) \bmod 2)) \cdot \tilde{\sigma}(x; t_i), \end{aligned} \tag{4.56}$$

A plot of how two such activation functions could look like can be seen in Figure 4.18.

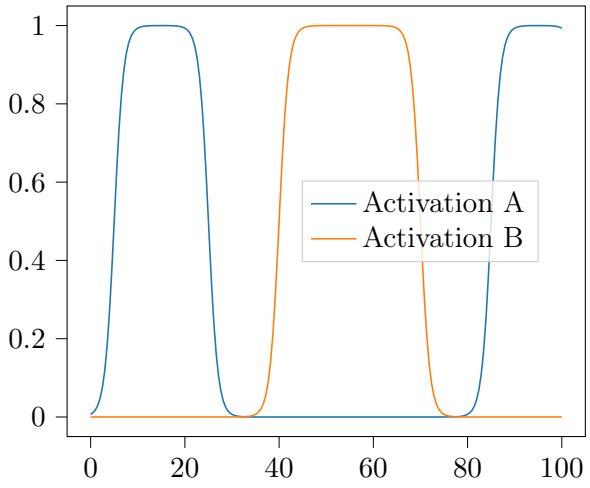


Figure 4.18: A visualization of how the two activation functions of two traffic lights coupled together can look like following definition (4.56).

4.3 Roundabouts

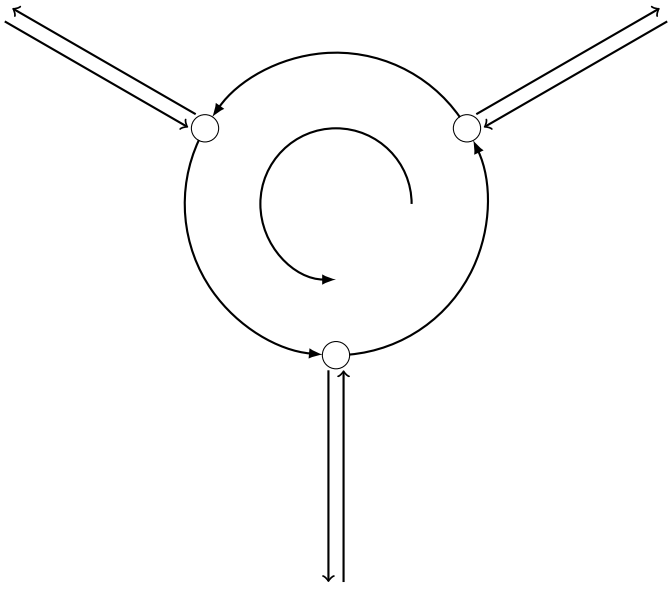


Figure 4.19: A roundabout modelled as a concatenation of junctions. The figure is inspired by the figure in [32].

In a general road network, roads are not only connected in junctions, but also meet in roundabouts. Similarly to what was done in [32], we will model these roundabouts as a concatenation of a special type of 2-to-2 junctions. Similar macroscopic roundabout models were investigated in, e.g., [33]. A visualiz-

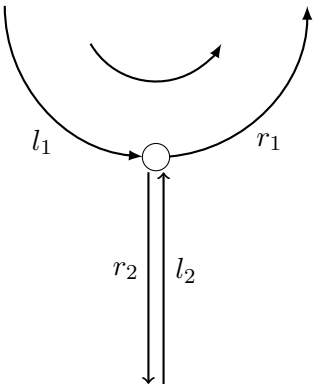


Figure 4.20: Example of a roundabout junction.

ation of such a roundabout can be seen in Figure 4.19. The junctions of these roundabouts will differ slightly from the junction introduced in Section 4.1.4. One of the major differences is that we will make use of a FIFO-rule for determining how the flux is distributed across the junction. In addition, we will allow the roundabouts to have incoming or outgoing roads that are not a part of the road network that we are modelling. In particular, this means that we do not care about the densities on these roads, only on the incoming and outgoing fluxes from these roads. This extension means that the junctions of the roundabout need to handle both the case where incoming and outgoing roads are a part of the road network, as well as the case where the incoming and outgoing roads are not a part of the road network.

First we will consider the junctions where we do not care about what happens to outgoing and incoming roads. Following a notation similar to that of section 4.1, we denote the four roads of the junction by l_1, l_2, r_1 and r_2. Here, l_1 represent the main incoming lane of the roundabout and r_1 represent the main outgoing lane of the roundabout. In contrast, l_2 and r_2 represent the external lanes leading in to and out of the roundabout. A visualization of this roundabout junction can be seen in Figure 4.20.

By the geometry of the junctions of the roundabout, there will be no crossing connections in contrast to the general 2-to-2 junction considered in Section 4.1.4, and hence there is no need to calculate an upper bound. We assume that the lanes l_2 and r_2 are not a part of the overall model, and so we do not keep track of the densities on these roads. Instead, we need to know the inflow to the incoming road $f_{l_2}^{\text{in}}$ at all times, the maximal possible inflow $f_{l_2}^{\max}$, and the maximal outflow of the outgoing road. For simplicity, we will not enforce an upper bound on the maximal outflow, meaning that all traffic wanting to enter road r_2 is allowed to. We assume that the inflow to the road $f_{l_2}^{\text{in}}$ is known as a function of time, but we also need to keep track of the amount of traffic that arrives at the roundabout, but isn't allowed to enter. Therefore we will introduce a queue $q_{l_2}(t)$ of potentially infinite length to model the amount of

traffic that is waiting to enter the roundabout. This queue length is described by the ordinary differential equation (ODE)

$$(q_{l_2})_t = f_{l_2}^{\text{in}} - f_{l_2}, \tag{4.57}$$

where f_{l_2} denotes the flux leaving road l_2, and q_{l_2} the queue length. We will come back to how the flux f_{l_2} can be calculated later in this section. We will also introduce a demand function for this road by

$$D_{l_2}(q, f^{\text{in}}) = \begin{cases} f_{l_2}^{\max} & \text{if } q > 0, \\ \min\{f^{\text{in}}, f_{l_2}^{\max}\} & \text{if } q = 0. \end{cases} \tag{4.58}$$

When the time step Δt is known, the queue will be updated numerically at time t^{n+1} as

$$q_{l_2}^{n+1} = q_{l_2}^n + \Delta t(f_{l_2}^{\text{in}} - f_{l_2}).$$

Some care needs to be taken to avoid this queue length becoming negative. If i.e., the queue length is only slightly larger the zero, and $f_{l_2} > f_{l_2}^{\text{in}}$, a too large time step could lead to the queue length $q_{l_2}^{n+1}$ becoming negative. Since the flux f_{l_2} is bounded from above by the demand function, a simple way to avoid the queue length is to modify the demand function as follows

$$D_{l_2}(q, f^{\text{in}}; \Delta t) = \begin{cases} \min\{f_{l_2}^{\max}, f^{\text{in}} + \frac{q}{\Delta t}\} & \text{if } q > 0, \\ \min\{f^{\text{in}}, f_{l_2}^{\max}\} & \text{if } q = 0, \end{cases} \tag{4.59}$$

or equivalently

$$D_{l_2}(q, f^{\text{in}}; \Delta t) = \min\{f_{l_2}^{\max}, f^{\text{in}} + \frac{q}{\Delta t}\}. \tag{4.60}$$

This modification guarantees that the queue length does not become negative.

Similarly to as in Section 4.1, the distribution of traffic across the junction will be determined by conservation of mass and a maximization of the outgoing flux, denoted by f_{r_1}. In contrast to Sections 4.1.4 and 4.1.5, instead of considering each connection l_i to r_j, we will immediately solve for the incoming and outgoing fluxes $f_{l_1}, f_{l_2}, f_{r_1}$ and f_{r_2}. The difference in approach comes from the fact that we are using a FIFO rule instead of a non-FIFO rule as we did before. We will also need a distribution parameter $\alpha \in [0, 1]$ that says how much of the incoming traffic from road l_1 that enters road r_2. We assume that u-turns are not allowed, and so any traffic from road l_2 is not allowed to enter road r_2. Hence, since all the traffic entering road r_2, we have using a FIFO rule that $f_{r_2} = \alpha f_{l_1}$. We calculate the outgoing flux on road r_1 as

$$f_{r_1} = \min\{(1 - \alpha)D_{l_1}(\rho_{l_1}) + D_{l_2}(q, f^{\text{in}}), S_{r_1}(\rho_{r_1})\}, \tag{4.61}$$

where S and D denotes the usual supply and demand functions introduced in Section 4.1. This choice maximizes the outgoing flux, leading to a solution satisfying Rule (B). Finally, we need to define some priority parameter $P \in$

$(0, 1)$ to determine what percentage of traffic is allowed to enter the junction. This parameter will be chosen following the discussion in Section 4.1.3. Since we assume no upper bound on the flux entering road r_2, and we are using a FIFO rule, we calculate the fluxes as discussed for the 2-to-1 junction in Section 4.1.3 with the fluxes defined as

$$\begin{aligned} f_{l_1} &:= \frac{1}{1-\alpha} \min\{(1-\alpha)D_{l_1}(\rho_{l_1}), \max\{PS_{r_1}(\rho_{r_1}), S_{r_1}(\rho_{r_1}) - D_{l_2}(q, f^{\text{in}})\}, \\ f_{l_2} &:= \min\{D_{l_2}(q, f^{\text{in}}), \max\{(1-P)S_{r_1}(\rho_{r_1}), S_{r_1}(\rho_{r_1}) - (1-\alpha)D_{l_1}(\rho_{l_1})\}, \\ f_{r_1} &:= \min\{(1-\alpha)D_{l_1}(\rho_{l_1}) + D_{l_2}(q, f^{\text{in}}), S_{r_1}(\rho_{r_1})\}. \end{aligned} \tag{4.62}$$

The factor $1 - \alpha$ comes from the fact that we are really calculating the flux from road l_1 to road l_1, $f_{1,1}$, and then making use of the relation

$$f_{1,1} = (1-\alpha)f_{l_1}.$$

Since the fluxes are calculated in the same way as in Section 4.1.3, we argue in the same way for why the calculated fluxes will satisfy Rules (A) and (B), while at the same time approximate Rule (C).

Now we consider the junctions containing roads that are a part of the network. In this case, we also need to consider the supply of the secondary outgoing road and we cannot simply do the calculation as before and put $f_{r_2} = \alpha f_{l_1}$, since this might violate the supply constraint of the secondary outgoing road. A simple way to fix this, is to update the demand function of incoming road l_1 as

$$\tilde{D}_{l_1}(\rho_{l_1}) = \min\{D_{l_1}(\rho_{l_1}), \frac{S_{r_2}(\rho_{r_2})}{\alpha}\}$$

We now define the fluxes as

$$\begin{aligned} f_{l_1} &:= \frac{1}{1-\alpha} \min\{(1-\alpha)\tilde{D}_{l_1}(\rho_{l_1}), \max\{PS_{r_1}(\rho_{r_1}), S_{r_1}(\rho_{r_1}) - D_{l_2}(q, f^{\text{in}})\}, \\ f_{l_2} &:= \min\{D_{l_2}(q, f^{\text{in}}), \max\{(1-P)S_{r_1}(\rho_{r_1}), S_{r_1}(\rho_{r_1}) - (1-\alpha)\tilde{D}_{l_1}(\rho_{l_1})\}, \\ f_{r_1} &:= \min\{(1-\alpha)\tilde{D}_{l_1}(\rho_{l_1}) + D_{l_2}(q, f^{\text{in}}), S_{r_1}(\rho_{r_1})\}, \\ f_{r_2} &:= \alpha f_{l_1}. \end{aligned} \tag{4.63}$$

We see immediately from these definitions that

$$f_{l_1} \leq \tilde{D}_{l_1}(\rho_{l_1}) \leq \frac{S_{r_2}(\rho_{r_2})}{\alpha},$$

which in turn implies that $f_{r_2} \leq S_{r_2}(\rho_{r_2})$, so the upper limit on the flux entering road r_2 is not violated. Since the fluxes $f_{1,1}$, $f_{1,2}$ and f_{l_1} are connected by the FIFO rule, maximizing one of them, while respecting the upper bounds on the others, will result in an overall maximum. Thus, the fluxes as defined in (4.63) will be the unique point satisfying the three rules (A), (B) and (C).

4.4 Public Transportation: Buses

As discussed in the introduction, this work does not only consider the flow of traffic, but also aims to reduce delay time in public transportation represented by busses. The introduction of buses to the LWR model was considered in detail by Lebacque et al [34], and a similar model was considered by Gasser et al. [35]. In this thesis, we model each bus as a particle travelling at the velocity of the traffic around it. Further, each bus has a predetermined route, specifying which roads it will drive on, as well as a number of stops at which it will drop off or pick up passengers. Although actual busses might not always stop at these bus stops, and when they stop, they halt for a variable amount of time, we will in our model assume that all busses stops at all bus stops, and that the stopping time is always 30 seconds.

In addition to having a predetermined route, each bus also has a predetermined schedule. That is, the times at which a given bus should arrive at a given stop is known. Although the schedule is usually given in minutes, we will in our simulation assume a schedule given in seconds. This means that we can calculate the delay in seconds, as opposed to only measuring the delay in minutes. Even though we said that the stopping time should be equal to 30 seconds, when a bus arrives more than 30 seconds prior to its scheduled arrival time, it will wait until the scheduled time before departing.

At each road in the simulation, the density of traffic on each of the cells is updated using a finite-volume scheme as described in Section 2.2.1. When deriving the model in Section 2.1.1, we assumed the simple relation between speed and density stated in (2.8). Thus, given the density of traffic on a cell, we can easily calculate the corresponding speed of traffic on that cell. At each time step of the simulation, we know the position of the bus, and therefore we can also calculate its speed, using the speeds of the closest cells. We can then update the position using the calculated speed. The length the bus drives during the time step is denoted by $L = v(\rho) \cdot \Delta t$, assuming that ρ is the density of the cell the bus is on.

As stated in the beginning of this section, the bus will always stop whenever it reaches a bus stop. In addition, it will also stop if it reaches a junction where the traffic light is turned red. The calculated length travelled in the current time step L, might sometimes be longer than the length to a bus stop, or the length to a junction where the bus cannot cross. Hence, to ensure that the bus actually stops where it should, we will keep track of the length to the next bus stop as well as the length to the next junction with a traffic light. We will now discuss how we handle the stopping of the bus.

We start by consider the case where a bus is approaching a bus stop. One possibility is to simply compute the length the bus should travel L using the speed of the neighbouring traffic. Then, the actual length the bus travels can be determined by taking the minimum of L and the length to the next bus stop. Although this method would work, actual busses will have an acceleration

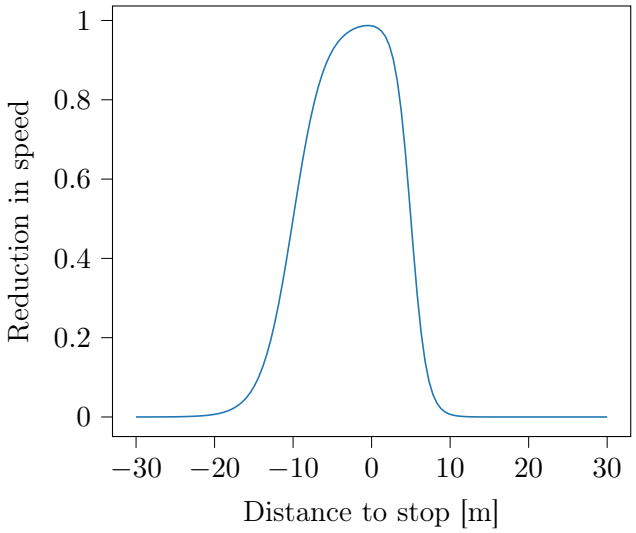


Figure 4.21: Reduction of the speed of the bus as a function of the distance to the closest bus stop.

period before and after each stop, which we will try to account for. The idea we will use is to multiply the speed of the bus by some factor between 0 and 1. This factor should depend on the distance to the next stop, in such a way that when the bus is close to the stop, there should be a greater reduction in the speed. Following the ideas from Section 4.2, we will compute the factor by a smooth activation function. We let this activation function be defined as the difference between two logistic functions. We want the peak of the activation function to be centered at zero, and we want the activation function to be close to zero away from the center. Thus, we need to scale and shift the two activation functions. The total activation function can be written as

$$\sigma_{\text{bus}}(d) = \sigma(\alpha_1 \cdot d + 5) - \sigma(\alpha_2 \cdot d - 5), \tag{4.64}$$

where d denotes the distance to the bus stop. The shifting parameters ± 5 are determined following a similar discussion as in Section 4.2. The scaling factors α_1 and α_2 determine the slope, and can be chosen independently. We visualize the idea in Figure 4.21, where the slope parameters have been chosen as $\alpha_1 = \frac{1}{2}$ and $\alpha_2 = 1$.

As stated, whenever the bus is far enough away from a bus stop, it will travel with the same speed as the traffic around it. When the bus approaches a bus stop, it will start to slow down, which will impact the traffic around it. Since a vehicle cannot drive through the bus, if there is no bus turnout, any vehicles coming from behind will need to overtake the bus. We will assume that this overtaking results in a reduction in the flow of traffic near the bus. How much this reduction is, depends on the specifics of the road and the specifics

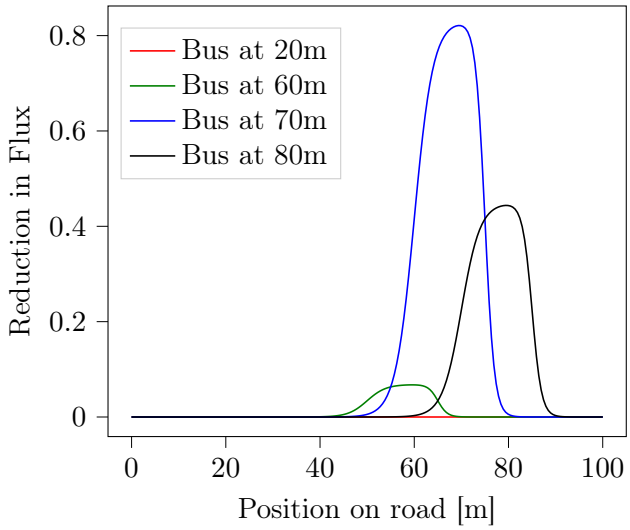


Figure 4.22: Reduction of the flux across the interfaces of the road as a function of the length of the roads for different positions of the bus. The full length of the road is 100 metres, and the bus stop is positioned at length 75 metres.

of a potential road going in the opposite direction.

We will make use of (4.64) to model how the traffic is affected, and will start by making some simple observations. First, when the bus is far away from any stops, it will travel at the speed of the traffic around it, and so the traffic should not be affected by the bus. Secondly, even when the bus is stopped at a bus stop, only the traffic close to the bus will be affected by the bus. Making use of these observations, one way of estimating the degree to which the speed of traffic should be slowed by could be as a product of two activation functions where one of the activation functions is a function of the distance between the bus and a stop, and the other activation function is a function of the distance from the position on the road to the position of the bus. Both of these activation functions could be defined as in (4.64), but with different distances as arguments.

The only place where the spikes of these activation functions overlap is when the bus is near a bus stop and we are considering the slowing of traffic close to the bus. We visualize this idea in Figure 4.22. The figure shows an example of a bus stopping at a bus stop on a road. The road has a full length of 100 metres, and the bus stop is placed at length 75 metres. We show the reduction in flux on the road for different positions of the bus. We first note that the interfaces far away from the bus stop is never affected. We also note that the spike in reduction is centered at the bus' position in all cases, but that the amplitude of the spike is greater when the bus is closer to the bus stop.

Now, we modify this flux reduction by making a third observation. Assume

for the moment that we are considering a single-lane road. If there are no roads going in the opposite direction, it will be quite hard for drivers to overtake the bus. If, on the other hand, there is a road going in the opposite direction, the drivers could potentially make use also of this road to overtake the bus. We assume that if there is no traffic in the opposing lane, the reduction in traffic is halved. We further assume that if the opposing lane is fully congested, the reduction in traffic is unchanged. We make use of a linear interpolation between these edge cases to obtain the actual reduction in traffic. Suppose we are considering only the flux across the interface at position $x = 10\text{m}$. Suppose further that we have computed a reduction of 50% without considering the opposing lane, and that the density of traffic at the same point on the opposing lane is equal to ρ_{opposing}. Then, we multiply the reduction by the factor $\frac{1}{2}(1 + \rho_{\text{opposing}})$ to obtain the actual reduction.

Next, we consider the case where the road we are considering has multiple lanes. We denote the number of lanes by n. Also in this case we expect to have a smaller reduction in the traffic. When the number of lanes is greater than one, we assume that the opposing lanes are not used to overtake the bus. Further since the bus is only positioned in one of the lanes, we expected the reduction in traffic to decrease with the number of lanes. We again consider the interface at position $x = 10\text{m}$ with average density of $\rho_{\text{interface}}$. In this case we now update the flux reduction with the factor $\frac{1}{2}(1 + \rho_{\text{interface}})$. In this way, when there is a large number of lanes, one bus stopping only marginally impacts the overall flux. A video showing a bus arriving at a bus stop and how the neighbouring traffic is impacted can be found at `torjeihelset.github.io/master_project`. The video compares four different cases with different number of lanes with and without lanes going in the opposite direction.

Finally, we move on the second case where the bus might not travel the full distance L, namely the case where it approaches a junction with a traffic light. As discussed in Section 4.2, we model the traffic light using a smooth activation function with range $[0, 1]$. For any road connection at any time, this activation will give a number on the percentage of the influx that can cross the junction. However, since the bus is modelled as a single particle, it will either cross the junction or not, and so the activation function cannot be used directly as it was when determining the amount of traffic crossing the junction. Instead, one can choose some threshold on the activation function, say 0.5, and let the bus cross the junction whenever the activation function is higher than the chosen threshold. On the other hand, if the activation is lower than the chosen threshold, the bus should wait at the junction.

One possible issue with this approach, is that the decision on whether the bus enters the outgoing road or not only depends only the incoming road and any traffic lights connecting the two roads. A scenario where this might cause an issue, is when the outgoing road is fully congested, there is some flow on the incoming road and the traffic light is green. In this case, if the bus is on the end of the incoming road, it will travel with some positive speed. If the

calculated length travelled is longer than the distance to the junction, the bus will cross the junction since the light is green. This will in turn mean that the bus will travel some distance on a fully congested road, which should not happen. One method for combating this effect is to also take into account the calculated flux from the incoming road onto the outgoing road to determine whether the bus should enter the outgoing road or not. If the calculated flux is too small, the bus should stop at the junction.

As discussed in Chapter 3, we want to calculate the gradient of some objective function with respect to the parameters of the model by ensuring that all parts of the numerical simulation that will be used in the objective function can be written as a sequence of differentiable operations. To be able to capture the full dependence of the delay of the busses with respect to the cycles of the traffic lights, it is important to ensure that all operations related to the busses are indeed differentiable. In particular, implementing a boolean check to see if the activation function of a traffic light is greater than a threshold or not will not be a differentiable operation, and hence all the dependence on the parameters will not be captured. Hence, we need some other way of modelling the bus stopping at a junction when the light is red.

We once again consider the smooth activation functions made up of logistic functions. As before, we start by making some simple observations that we want to satisfy. First, when the bus is far away away from the junction, the reduction in speed due to the junction should be negligible. Only when the bus is close to the junction should its speed potentially be changed. The reduction in speed due to the junction should also in some way depend on the activation of the traffic light controlling the junction the bus is entering. To account for these two phenomena, we will consider two different activation functions. One activation is related to the distance of the bus to the junction, and the other is related to the traffic light. Then, we will take the maximum of these two functions to get the reduction of the bus.

First we consider the activation that relates to the bus' distance to the junction. We want the speed to be unchanged when the bus is far away from the junction. Therefore this activation function should be close to one when the bus is far away. When the bus is close to the junction, we want the speed to be determined by the activation of the traffic light. Therefore, this activation function should be close to zero so that the traffic lights activation function should dominate. We visualize the idea in Figure 4.23, where we again consider a road of length 100 metres. When the position of the bus approaches the end of the road, the activation function decreases.

Now we consider the second part, that is the bus' dependency of the traffic light. The simplest possibility here is to simply make use of the activation of the traffic light that we have discussed how to calculate in Section 4.2. We then take the maximum of these activation functions and multiply the speed of the bus by the maximum value. Since neither of the activation functions will ever reach exactly zero, the activation function depending on the bus' distance to

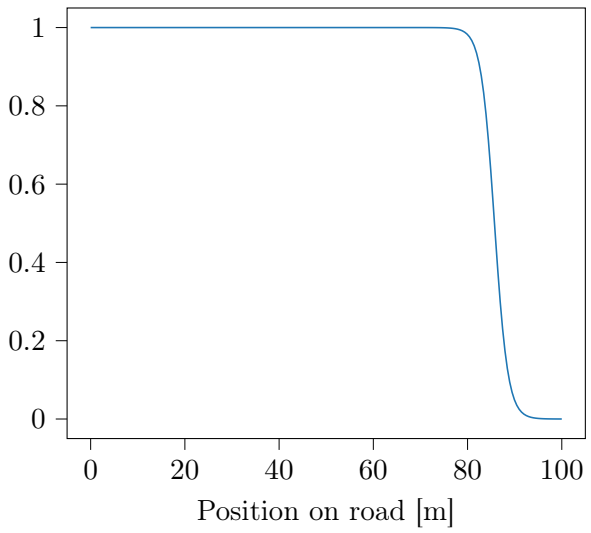


Figure 4.23: Activation function of the reduction in speed depending on the closeness of the bus to the next junction. When the activation function is close to one, there is no reduction, and when the activation function is close to zero the speed is heavily reduced.

the junction need to start decreasing sufficiently far away from the junction to ensure that the bus only crosses when the traffic light is actually green.

4.5 Variable Speed Limits**

Another natural extension to the model is to allow the speed limit to vary in time. We will model the speed limits as a piecewise constant function that may only change at at finite number of control points, $0 \leq \mu_1 < \cdots < \mu_{N_{speed}} \leq T$. The distance $\mu_{k+1} - \mu_k$ may be different for each j, but since it is not very realistic that the speed limit will vary very rapidly, this distance should not be taken too small.

One important feature of speed limits is that the speed does not change instantly for the whole road. When a sign displaying the speed limit changes, the roads in front have no way of knowing that the speed limit changes until they reach a new speed limit sign. Therefore, the new speed limit will effectively move to the right with the same speed as the first vehicle arriving at the sign. Thus, some time after the speed limit changes, the speed limit at some point on the road will be a function of both time and space. However, for the purposes of the discussions herein, we will assume that the change is instant over the whole road.

It is important to ensure that the times of the jumps in speed are included as control points in the simulation. That is, if t_k is a control point, the time

step of the simulation should be controlled as $\Delta t = \min\{\Delta t_{CFL}, t_k - t\}$ if t was the previous time of the simulation.

Chapter 5

Optimal Control Problem

We recall that the goal of this work is to find the optimal control of traffic lights and the best choices of speed limits to either improve the flow of traffic in a road network or to reduce the delay time of public transport such as busses. In this chapter, we will discuss some possible objective functions that can be used to either quantify the flow of traffic or the delay of public transport. Then, we will discuss some optimization methods that can be used to find the optimal control parameters.

5.1 Objective Functions

The first step to find an optimal set of control parameters is to quantify what we mean by a good solution. If our aim is to improve the general flow of traffic, we first need to define some way of measuring the flow. On the other hand, if we instead want to minimize the delay time, we need some way of combining the delays from different bus stops for different busses. The optimal control problem has been considered from different viewpoints using a multitude of different objective functionals. We refer the interested reader to, e.g., [28, 36–39], among others. We will in the following subsections introduce different objective functionals that may be used to measure the optimality of a given set of control parameters.

5.1.1 Optimizing the General Flow of Traffic

We start by considering objective functions that can be used to quantify the quality of the flow in the network. This is a complex problem where different ideas are possible. One natural idea is to maximize the total throughput from the system. That is, we can try to maximize the number of cars exiting the system. Letting R_{out} denote the set of all roads not connected to the road network at their right end, we can measure the total throughput using the fluxes exiting these roads and integrate over the simulation period. Assuming that $f_{r_i}(t)$ denotes the exiting flux from road $r_i \in R_{\text{out}}$ as a function of time,

we write the objective function

$$J(\{f_{r_i}\}_{r_i \in R_{\text{out}}}) = -\int_0^T \sum_{r_i \in R_{\text{out}}} f_{r_i}(t)\, dt, \tag{5.1}$$

where we have made the assumption that the simulation period starts at time $t = 0$ and ends at $t = T$. We add a minus sign in front to formulate a minimization problem.

Our finite-volume method will produce approximate exiting fluxes f_{r_i} at a finite number of points in time, $0 = t_0 < t_1 < \cdots < t_{N_t - 1} < t_{N_t} = T$, separated by time steps $\Delta t^n = t_{n+1} - t_n$ that may vary with n. Using the approximate exiting flux $f_{r_i}^n$ calculated at each time point t_n, we use the trapezoidal rule to compute a numerical approximation to (5.1)

$$\int_0^T \sum_{r_i \in R_{\text{out}}} f_{r_i}(t)\, dt \approx \sum_{n=0}^{N_t - 1} \left(\sum_{r_i \in R_{\text{out}}} \frac{f_{r_i}^{n+1} + f_{r_i}^n}{2} \right) \Delta t_n. \tag{5.2}$$

Since our numerical simulation is deterministic, the exiting fluxes depend only on the initial data and the control parameters. Ordering the control parameters (speed limits and cycle times for the traffic lights) as one long vector p, we write the numerical objective function as

$$J(p) = -\sum_{n=0}^{N_t - 1} \left(\sum_{r_i \in R_{\text{out}}} \frac{f_{r_i}^{n+1} + f_{r_i}^n}{2} \right) \Delta t_n. \tag{5.3}$$

A different way to quantify the quality of the traffic flow is to consider the travel times of the cars in the system. Even when restricting the focus only to the travel times of the cars, different viewpoints are possible. One can either try to minimize the longest travel time, or one can try to minimize the average travel time. Finally, one can try to minimize the combined travel time of all the cars in the system. If the exact densities were known for all cars and for all roads, we could write this objective function, in a similar way as in [28],

$$J(\{\rho_i(x,t), q_i(t)\}_{i \in R}) = \sum_{r_i \in R} \int_0^T \left(q_i(t) + \int_0^L \rho_i(x,t) dx \right) dt, \tag{5.4}$$

where $\rho_i(x,t)$ denotes the density of cars on road i as a function of time and space, $q_i(t)$ denotes the length of the queue of traffic leading into road i, and R denotes the set of all roads in the network. As we do not know the exact functions $\rho_i(x,t)$ and $q_i(t)$, but rather the approximate evaluations $\rho_{i,j}^n$ and q_i^n, we can define the numerical objective function as

$$J(p) = \sum_{r_i \in R} \sum_{n=1}^{N} \frac{\Delta x \cdot \Delta t^n}{2} \left(\sum_{j=1}^{N_{r_i}} \rho_{r_i,j}^n + \rho_{r_i,j}^{n+1} \right), \tag{5.5}$$

where again p denotes the vector containing all of the control parameters of the model.

5.1.2 Minimizing the Delay of Busses

Instead of trying to improve the general flow through the network, a different goal is to improve the performance of buses by reducing their delays. Although more complicated objective functions are possible, the most natural approach is to sum all the delay times for each bus, before taking the sum of all busses. When simulating the evolution of traffic for a certain amount of time, one potential problem with this approach may arise. Assume that at least one of the busses stops between two bus stops at the end of the simulation with one choice of control parameters. Assume further that with a different choice of control parameters, the bus travels faster through the network and also reaches the second stop. In this case, it might be that the total delay time of the bus increases, since one extra delay is added, even though the bus travels faster. Even though the delay of the bus at the first stop is reduced, the sum of the delays at both stops may be greater than the previous delay at the first stop.

There exist several ways to fix this issue. Instead of deciding a simulation time before starting the simulation, one can simulate until all the busses have exited the network. Although this fixes the problem, it would be better to know how long the simulation is beforehand. If for some reason one bus stops completely in the network, the simulation might not terminate in a reasonable amount of time. Hence, we consider the different approach where we take the average of the delays, instead of just summing them. In most cases, this is likely a good approach, but there might still be some special cases where it leads to some inaccurate results. Suppose that a bus stops somewhere between two stops. Suppose further that we find a set of control parameters that reduces the delay of the bus at the first case. In this case, it might be that this new set of control parameters leads to a greater delay between the two stops so that the average delay time would be greater if the simulation was run for longer. Nevertheless, since this effect seems to play a role only for exotic scenarios, we choose to ignore it.

The numerical arrival times of all the busses depend only on the initial data and the control parameters. Supposing that there are N_{buses} buses in the simulation, each stopping at N^i_{stops}, $i = 1, \dots, N_{\text{busses}}$ bus stops. Fixing the initial data, we write the objective function as

$$J(p) = \frac{1}{\sum_{i=1}^{N_{\text{buses}}} N^i_{\text{stops}}(p)} \sum_{i=1}^{N_{\text{buses}}} \sum_{j=1}^{N^i_{\text{stops}}(p)} t^{\text{delay}}_{i,j}(p), \tag{5.6}$$

where p as before is a vector containing all the control parameters, and $t^{\text{delay}}_{i,j}(p)$ is the delay of bus i at the j^{th} stop it reaches. We note that N^i_{stops} depends on the choice of control parameters p.

5.1.3 Combining Optimal Flow with Minimal Delay

The problem with choosing a objective function that either only measures the efficiency of the traffic flow or only the delay of the busses, is that optimizing one measure one does not necessarily translate in a good solution for the other. It could be that when reducing the delay of the busses, one or more roads experience heavy traffic. These could be roads that, e.g., none of the busses travel on. A natural solution approach to avoid this is to combine the objective functions (5.3) and (5.6) to form the new objective function

$$\begin{aligned} J(p) =& \frac{\gamma^{\text{delay}}}{\sum_{i=1}^{N_{\text{busses}}} N_{\text{stops}}^i} \sum_{i=1}^{N_{\text{busses}}} \sum_{j=1}^{N_{\text{stops}}^i} t_{i,j}^{\text{delay}}(p) \\ & - \gamma^{\text{flow}} \sum_{n=0}^{N_t} \left(\sum_{r_i \in R_{\text{out}}} \frac{f_{r_i}^{n+1} + f_{r_i}^n}{2} \right) \Delta t_n, \end{aligned} \tag{5.7}$$

where γ^{delay} and γ^{flow} are parameters that can be used to decide which of the two parts of the objective function should dominate. It is even possible to weigh different roads and different busses differently. We can define the objective function

$$\begin{aligned} J(p) =& \frac{1}{\sum_{i=1}^{N_{\text{busses}}} N_{\text{stops}}^i} \sum_{i=1}^{N_{\text{busses}}} \gamma_i^{\text{delay}} \sum_{j=1}^{N_{\text{stops}}^i} t_{i,j}^{\text{delay}}(p) \\ & - \sum_{n=0}^{N_t} \left(\sum_{r_i \in R_{\text{out}}} \gamma_i^{\text{flow}} \frac{f_{r_i}^{n+1} + f_{r_i}^n}{2} \right) \Delta t_n, \end{aligned} \tag{5.8}$$

where γ_i^{delay} is the weight parameter for bus i and γ_i^{flow} is the weight parameter for road r_i. If for some reason, we wante to weigh the individual bus stops differently, this would also be possible by adding more weight parameters to the objective function. Defining $\gamma_{i,j}^{\text{delay}}$ as the weight parameter for the j^{th} stop of bus i, we can define the objective function

$$\begin{aligned} J(p) =& \frac{1}{\sum_{i=1}^{N_{\text{busses}}} N_{\text{stops}}^i} \sum_{i=1}^{N_{\text{busses}}} \sum_{j=1}^{N_{\text{stops}}^i} \gamma_{i,j}^{\text{delay}} t_{i,j}^{\text{delay}}(p) \\ & - \sum_{n=0}^{N_t} \left(\sum_{r_i \in R_{\text{out}}} \gamma_i^{\text{flow}} \frac{f_{r_i}^{n+1} + f_{r_i}^n}{2} \right) \Delta t_n. \end{aligned} \tag{5.9}$$

A different way to combine the two optimization goals, is to keep the same objective functions as before, but instead impose constraints on the value of the other objective function. We could, e.g., try to minimize the bus delay using (5.6), while at the same time ensuring that the total throughput from the system always exceeds some upper bound. We will discuss how this can be done in practice in the next section.

5.2 Determining Optimal Solution

Assuming that we have some objective function $J(p)$ we want to minimize, we need to determine some way of finding the optimal control parameters p, i.e., solve

$$\arg\min_{p} J(p), \tag{5.10}$$

using some optimization method. In most cases, the control parameters in p cannot take an arbitrary value, but are instead restricted to some interval. For each component of the vector we have

$$p_i \in [l_i, u_i],$$

for some lower bound l_i and some upper bound u_i. The speed limit of a road cannot be chosen arbitrarily high, and will depend on the specifics of the road, and the rest of the road network it is a part of.

Each evaluation of $J(p)$ (and its gradient) entails having to run the full simulation, which is computationally costly. To reduce the number of times $J(p)$ needs to be evaluated, we want to make use of a gradient-based optimization method. As described in Chapter 3, we compute the gradient $\nabla J(p)$ using automatic differentiation. In this section we will consider different gradient-based optimization methods, as well as a way of minimizing one objective function $J_1(p)$, while ensuring that another objective function $J_2(p)$ satisfies certain constraints.

5.2.1 Gradient Descent

Maybe the simplest gradient-based optimization method is the gradient descent or the steepest descent method, see, e.g., [40]. This is an iterative procedure that starts from an initial guess p^0 and then aims to find a local minimum, p^* of the objective function $J(p)$. The basic idea behind the gradient descent algorithm is that at each iteration, the new iterate should be found by moving in the direction of steepest descent; that is, moving in the direction $-\nabla J(p^n)$. As shown in, e.g., [40], this is the direction where the function decreases fastest. We can write the gradient descent algorithm starting from p^0 as

$$p^{n+1} = p^n - \alpha^n \nabla J(p^n), \tag{5.11}$$

where α^n denotes the step length. Following Chapter 3, we know how to compute $\nabla J(p^n)$ for each iterate p^n. What remains is a way of determining the step length α^n. At each iteration, we would ideally choose α^n to substantially reduce the objective function. On the other hand, we do not want to spend too much effort in finding the step length. The ideal step length can be found by solving a new optimization problem, the so-called line-search problem

$$\alpha^n = \arg\min_{\alpha>0} J\big(p^n - \alpha \nabla J(p^n)\big), \tag{5.12}$$

but as our goal was to reduce the number of times $J(p)$ is evaluated, we do not want to have to solve (5.12) at each iteration. Instead, we want to perform an *inexact* line search to achieve *sufficient* reduction in the objective function to get convergence.

Before giving the details of the inexact line-search method, we start by considering an example showing that the gradient descent method might not converge for an arbitrary choice of step lengths. The only condition we impose on the step length is that it should at least achieve some reduction in the objective value. That is, we impose the condition that

$$J(p^n - \alpha^n \nabla J(p^n)) < J(p^n). \tag{5.13}$$

As explained in, e.g., [40], it is not enough to only impose (5.13) to guarantee convergence. Hence, we introduce a line search condition that guarantees a *sufficient decrease* in the objective function by imposing the condition

$$J(p^n - \alpha^n \nabla J(p^n)) \leq J(p^n) + c_1 \alpha (\nabla J(p^n))^T \nabla J(p^n), \tag{5.14}$$

for some constant $c_1 \in (0, 1)$. This condition is commonly referred to as the *Armijo condition*. We also saw that the steepest descent method might not converge when α^n is chosen too small. However, as shown in [40], (5.14) is satisfied for all sufficiently small α. Hence, we need a second condition for the inexact line search. To ensure that α^n is not chosen too small, we impose the second condition

$$(\nabla J(p^n - \alpha^n \nabla J(p^n)))^T \nabla J(p^n) \geq c_2 (\nabla J(p^n))^T \nabla J(p^n), \tag{5.15}$$

for constant $c_2 \in (c_1, 1)$. This condition (5.15) is commonly referred to as the *curvature condition*. The combination of (5.14) and (5.15) is known as the *Wolfe conditions*. When the step length α^n satisfies the Wolfe conditions, convergence of the method can be guaranteed under some quite general assumptions [40].

Although we have stated that we need both conditions (5.14) and (5.15) to guarantee convergence, we now introduce a method of choosing a suitable step length using only condition (5.14). This method is known as a *backtracking* approach. Staring from an initial guess α_0^n, we repeatedly reduce the step length until (5.14) is satisfied. We summarize the basic idea in Algorithm 1, similar to the one in [40].

Algorithm 1 Steepest Descent Backtracking

1: Choose $\alpha_0^n, \rho \in (0, 1), c \in (0, 1)$
2: Set $\alpha^n = \alpha_0^n$
3: **while** $J(p^n - \alpha^n \nabla J(p^n)) > J(p^n) - c\alpha^n (\nabla J(p^n))^T \nabla J(p^n)$ **do**
4: Set $\alpha^n = \rho \cdot \alpha^n$

In addition to making use of a backtracking procedure, we choose the initial guess for the step length at each step, α_0^n in a specific way. Supposing that there

is at least one nonzero component of the gradient vector $\nabla J(p^n)$, we find the largest element in absolute value by

$$p^n_{\max} = \arg\max_i |(\nabla J(p^n))_i| .$$

We then choose the initial guess for the step length as

$$\alpha^n_0 = \frac{\Delta p}{p^n_{\max}}, \tag{5.16}$$

where Δp is some constant for controlling the maximum updating the components of p. Unless otherwise stated, we will set $\Delta p = 20$, meaning that we try to update at least one of the control parameters by a value of 20.

As mentioned in the introduction of this section, each component of p^n is restricted to certain intervals. Although the steepest descent method is in its most basic form an unconstrained optimization method, we can modify it to ensure that the bounds on p are satisfied. At each iteration, we must ensure that no components of the new iterate p^{n+1} are outside their feasible interval. One possibility is to modify the iteration step as

$$p^{n+1} = P[p^n - \alpha^n \nabla J(p^n))], \tag{5.17}$$

where $P[\cdot]$ is a projection operator defined for each component as

$$P[p]_i = \text{median}\{p_i, l_i, u_i\}.$$

As we shall see, this projection will be equivalent to an inexact line-search method with a different search direction. Suppose that we want to write (5.17) on the form

$$p^{n+1} = p^n + \alpha^n p^n_{\text{search}}, \tag{5.18}$$

we need to determine the components of the search direction p^n_{search}. As the projection operator $P[\cdot]$ projects all components of p^{n+1} onto the feasible region, we know that the inequalities

$$l_i \leq p^{n+1}_i \leq u_i$$

are satisfied for all components p^{n+1}_i. Inserting the expression (5.18) for p^{n+1} and rewriting the inequalities gives the new inequalities

$$\frac{u_i - p^n_i}{\alpha^n} \leq (p^n_{\text{search}})_i \leq \frac{l_i - p^n_i}{.}\alpha^n. \tag{5.19}$$

Using the inequalities (5.19) gives us a way of defining the new search direction. We put

$$(p^n_{\text{search}})_i = \text{median}\Big\{-(\nabla J(p^n))_i, \frac{u_i - p^n_i}{\alpha^n}, \frac{l_i - p^n_i}{\alpha^n}\Big\}, \tag{5.20}$$

so that the all components of p^n_{search} are equal to the corresponding components from the gradient wherever possible, and equal to the upper or lower bounds

from (5.19) for the other components. With this definition, the two iterations (5.17) and (5.18) will be equal.

We have yet to discuss how to determine the step lengths for this slightly different line search. As for the steepest descent, we will make use of a backtracking algorithm to ensure that the step length is chosen to guarantee convergence. We replace the search direction $\nabla J(p^n)$ by the new search direction $p^n_{\text{search}}(\alpha^n)$ in Algorithm 1 and summarize the new approach in Algorithm 2.

Algorithm 2 General Backtracking

1: Choose $\alpha_0^n, \rho \in (0,1), c \in (0,1)$
2: Set $\alpha^n = \alpha_0^n$
3: **while** $J(p^n + \alpha^n p^n_{\text{search}}) > J(p^n) + c\alpha^n (\nabla J(p^n))^T p^n_{\text{search}}$ **do**
4: Set $\alpha^n = \rho \cdot \alpha^n$

We still need a suitable way of choosing the initial guess for the step length α_0^n. We still make use of (5.16), but define $p^n_{\max}$ in a slightly different way. We put

$$p^n_{\max} = \underset{i \in I_{\text{interior}}}{\arg\max} \left| (\nabla J(p^n))_i \right|,$$

where I_{interior} is a set of indexes where one of the three following conditions need to hold true:

(i) $l_i < (p^n)_i < u_i$
(ii) $l_i = (p^n)_i$ and $(\nabla J(p^n))_i < 0$
(iii) $u_i = (p^n)_i$ and $(\nabla J(p^n))_i > 0$

The first point holds true for all indices i where the component $(p^n)_i$ is in the interior of the feasible region. The second point holds true for all indices i where $(p^n)_i$ is at its lower bound, but the associated component of the gradient is negative, so that $(p^{n+1})_i$ will be larger than its lower bound. The final point is similar to the second point, but is related to the upper bound of the components.

5.2.2 Constrained Optimization

As discussed in Section 5.1.3, one way of determining the best control parameters is to optimize one objective function, while at the same time ensuring that condition on a different objective function is satisfied. We could, e.g., try to reduce the delay of busses, while at the same time ensuring that the througput of the network is greater than a certain lower bound. In Section 5.2.1, we explained how the steepest descent method could be modified to account for bounds on the control parameters. However, the same method cannot be used when we also have an extra constraint on another objective function. We will here explain briefly how such constraints can be handled by making use of Lagrange multipliers. This subsection and the notation we use in it will closely follow chapter 12 of [40].

A general formulation of the type of constrained problems we are looking at is

$$\arg\min_{p} J_1(p) \quad \text{subject to} \quad J_2^{\text{bound}} - J_2(p) \geq 0, \tag{5.21}$$

where J_1 is the objective function we are minimizing with respect to, while ensuring that the second objective function J_2 is less than some upper bound J_2^{bound}. Adding the lower and upper bounds on the control parameters, the problem can be written as

$$\arg\min_{p} J_1(p) \quad \text{subject to} \quad l \leq p \leq u \text{ and } J_2^{\text{bound}} - J_2(p) \geq 0. \tag{5.22}$$

Following [40], we can define the *feasible set* Ω

$$\Omega = \{p : l_i \leq p_i \leq u_i, {}_2^{\text{bound}} - J_2(p) \geq 0\}, \tag{5.23}$$

and rewrite (5.22) as

$$\arg\min_{p \in \Omega} J_1(p). \tag{5.24}$$

We discuss briefly how such a constrained problem can be solved. We will as in [40] introduce the *Lagrangian function* and state some conditions an optimal solution needs to satisfy. We start by considering a more general problem on the form

$$\min_{x} f(x) \text{ s.t. } \begin{cases} c_i(x) = 0, & i \in \mathcal{E}, \\ c_i(x) \geq 0, & i \in \mathcal{I}, \end{cases} \tag{5.25}$$

where $\mathcal{E}$ and $\mathcal{I}$ are two finite index sets. We also generalize the feasible set Ω as

$$\Omega = \{x | c_i(x) = 0, i \in \mathcal{E}; c_i(x) \geq 0, i \in \mathcal{I}\},$$

where again c_i for $i \in \mathcal{E}$ is the set of equality constraints, and c_i for $i \in \mathcal{I}$ is the set of inequality constraints.

A notion that will be useful later is the notion of the *active set*. The following definition from [40] defines what is meant by active set $\mathcal{A}(x)$.

Definition 5.2.1 (Active set)**.** *The* active set $\mathcal{A}(x)$ *at any feasible x consists of the equality constraint indices from $\mathcal{E}$ together with the indices of the inequality constraints i for which $c_i(x) = 0$; that is,*

$$\mathcal{A}(x) = \mathcal{E} \cup \{i \in \mathcal{I} | c_i(x) = 0\}.$$

We also introduce a condition that will be useful later. The following definition from [40] defines what is meant by the *linear independence constraint qualification* or the (LICQ).

Definition 5.2.2 (LICQ)**.** *Given the point x and the active set $\mathcal{A}(x)$ defined in Definition 5.2.1, we say that the* linear independence constraint qualification (LICQ) *holds if the set of active constraint gradients $\{\nabla c_i(x), i \in \mathcal{A}(x)\}$ is linearly independent.*

To be able to recognize local solutions to the constrained problem, we introduce the Lagrangian function associated to (5.25) as

$$\mathcal{L}(x, \lambda, \mu) = f(x) - \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i c_i(x), \tag{5.26}$$

where λ_i are so-called *Lagrange multipliers*. Each of these multipliers are associated to a constraint of the problem.

We are now ready to state a first-order condition that any optimal solutions need to satisfy. The following theorem from [40] defines necessary conditions for a point to be a local solution.

Theorem 5.2.1 (First-Order Necessary Conditions)**.** *Suppose that x^* is a local minimum of* (5.25)*, that the functions f and c_i are continously differentiable, and that the LICQ holds at x^*. Then there is a Lagrange multiplier vector λ^*, with components $\lambda_i^*, i \in \mathcal{E} \cup \mathcal{I}$, such that the following conditions are satisfied at (x^*, λ^*)*

$$\begin{aligned} \nabla_x \mathcal{L}(x^*, \lambda^*) &= 0, \\ c_i(x^*) &= 0, \quad \forall i \in \mathcal{E} \\ c_i(x^*) &\geq 0, \quad \forall i \in \mathcal{I} \\ \lambda_i^* &\geq 0, \quad \forall i \in \mathcal{I} \\ \lambda_i^* c_i(x^*) &= 0, \quad \forall i \in \mathcal{E} \cup \mathcal{I}. \end{aligned} \tag{5.27}$$

There exist many methods for solving problems of the type (5.25). Two of the most common such methods are sequential quadratic programming (SQP) or interior-point methods. We will not go into much details of these methods, but instead refer the reader to, e.g., [40].

Chapter 6

Implementation

In this chapter, we discuss how the model and the method for solving the optimal control problem was implemented. A complete set of source codes can be freely downloaded from GitHub: `github.com/TorjeiHelset/master_project`. The homepage of this repository contains a few animations of the numerical experiments we consider in Chapter 7. The homepage can be found at `torjeihelset.github.io/master_project`.

6.1 Implementation of the Model

All of the code used for the simulation was developed from scratch using the PyTorch framework in Python [41]. The reason PyTorch and Python was chosen over other potentially faster frameworks and languages, is that it supports automatic differentiation or AD, as well as the ease of implementation that comes with Python. Although there already exist finite-volume solvers in Python, it is challenging to adapt these to take into account all the different components of our model. In addition, since our solution approach requires gradient computation using AD as discussed in Chapter 3, we have implemented a finite-volume solver from scratch. PyTorch is a Python framework that supports AD, and thus it seems like a good framework for building our simulator. To ensure that the correct derivative is approximated, some care must be taken to ensure that all operations performed are support by the PyTorch framework.

The code is split into different classes, each class representing one of the components of the model. The overall road network can be broken down into smaller parts; it consists of a number of roads, a number of junctions connecting these roads, as well as a number of roundabouts. Each junction has a number of incoming and outgoing roads as well as up to several traffic lights of the two types discussed in Section 4.2. The roundabouts are modelled as a concatenation of a special type of 2x2 junction. However, since these junctions are slightly different than the others in the simulation, a separate class will be created to model them. Further, since the support roads of a roundabout are

not actually a part of the road network being simulated, these roads are also described by their own class. Finally, the simulation includes a number of busses travelling on scheduled routes. Following this division of components, we create a class for each of the different components. The subsequent sections present a simplified explanation of the various classes and methods that are a part of the simulation. In the repository`github.com/TorjeiHelset/master_project`, all of these classes can be found as separate python files inside the `scr` folder.

6.1.1 The Road Class

The `Road` class handles all of the logic and parameters related to a single road. The roads considered in this simulation all have a speed limit, or a list of speed limits that can change at specified times. Each of the speed limits is one of the parameters we are optimizing the objective function with respect to, and so to compute the overall gradient, any operations including these speed limits need to be tracked. When making use of the backward mode AD in PyTorch, one can specify that operations should be tracked by declaring the speed limit parameters with `requires_grad=True`. Since we allow the speed limits to change at discrete points in time, we also need a set of times where the speed limits changes. In addition, each road has a certain length and a number of internal cells. While the lengths of each road in the network may vary, we construct the roads in such a way that the cell size Δx is uniform across all roads. This is done to simplify the updating of the flux crossing the junctions where different roads meet, but with a small change in the code, it would be possible to allow each cell to have different lengths.

As discussed, each road is unidirectional, and so to model a two-directional road, one needs to make use of two `Road` objects. As mentioned, when modelling a multi-lane road, instead of considering the lanes separately and trying to model the flow between lanes, we will consider it as a regular unidirectional road only with higher maximum density. Thus, each `Road` object also needs to have some maximum density so that the maximum density of a two-lane road is twice that of a single lane road.

For each `Road` object, both the initial density as well as the boundary conditions need to be specified. If the road is connected to a junction at both ends, the boundary conditions will be handled by these junctions. On the other hand, if the road has one or two ends not connected to a junction, incoming and/or outgoing boundary conditions are needed. These boundary conditions can either be constant, or given by some function depending on the time.

Finally, since the simulation supports different finite-volume schemes for updating the densities, one needs to specify which scheme should be used on each road. In general, it is possible to make use of different schemes for the different roads, but in this work, we will make use of the same scheme for all roads. We summarize the member variables of the road class in Table 6.1.

The `RoadNetwork` class handles the time stepping, but at each time step

Table 6.1: Member variables of the `Road` class.

v_{limits} :	List of speed limits
v_{times} :	List of times where the speed limit changes
L :	Length of the road
N :	Number of internal cells
f_{init} :	Initial density distribution
f_{boundary} :	Boundary condition
scheme :	String specifying which finite-volume scheme should be used
$\rho_{\max}$:	Maximum density
id :	Id used to differentiate between `Road` objects

the internal updating is handled by the road class. Therefore, the road class contains some methods that are being called by the `RoadNetwork` class.

Methods

In addition to the member variables listed in Table 6.1, each `Road` object also has a set of member functions that handle the logic. As described in Section 2.2.1, the time step is chosen according to a CFL condition (2.22) that depends on the maximum wave speed over the entire simulation. Therefore, each road has a method that calculates the maximum speed of the waves emanating from the cell interfaces.

Assuming that the time step Δt has been calculated so as to satisfy the CFL condition (2.22), the densities at the next time need to be calculated. The internal cells, i.e., the cells not connected to junctions or boundaries of the road network are updated using a finite-volume scheme. To this end, the `Road` class has a method that updates the densities on the internal cells using the specified finite-volume scheme. The finite-volume updating methods are not an internal method of the `Road` class, but will instead be called from this class.

The cells connected to the junctions also need to be updated. To do this, it is first necessary to calculate the fluxes entering and exiting the road. When the road is connected to a roundabout or a junction, this flux will be calculated inside the roundabout or junction class. If, on the other hand, one of the edges are not connected to the network, the flux entering or exiting the road needs to be determined using a boundary condition. If the left edge of a road is not connected to the network, the incoming flux needs to be specified as a function of time. For the right edge, we assume that all traffic that reaches the boundary is allowed to exit, and so there is no need to specify an extra boundary condition. To check whether a road is connected at its edges, two binary variables are introduced and initialized to be zero. Upon the initialization of

the junctions or roundabouts, all relevant edges are set to one.

As we have seen, we want our scheme to be conservative. For cells away from the boundary where a finite-volume updating scheme of the form

$$\rho_j^{n+1} = \rho_j^n - \lambda(F_{j+1/2}^n - F_{j-1/2}^n),$$

is applied both to the cell itself as well as it two neighbouring cells, the scheme is conservative since the fluxes $F_{j+1/2}^n$ and $F_{j-1/2}^n$ will cancel out. When updating the boundary cells, where the finite-volume scheme used for interior cells is not used, we need to take care to ensure that all fluxes used for updating cells appear with an opposite sign, so that the scheme remains conservative.

As discussed, each road may change its speed limit at a finite number of points in time. When initializing a `Road` object, the different speed limits and their associated time of change need to be specified. The `Road` class has a member variable that corresponds to the current speed limit. To capture the change in speed limit, a method for updating the speed limit is needed. This method only depends on the time of the simulation t, and compares this time with the times at which the speed changes to return the current speed limit.

Finally, since the updating of each bus is determined by a speed that in turn depends on the density at a specific point on a specific road, the `Road` class also has a method to calculate the speed at a certain length on the road. This method takes in a length, and first finds the two cells with centers closest to this length. It then calculates a weighted average of the speeds of the two cells. If, for instance, the length corresponds to a cell interface, the speed will be calculated as the average of the speeds of the two neighbouring cells. If instead the length corresponds to the midpoint of a cell, the calculated speed will be equal to the speed on this cell.

6.1.2 The Junction Class

The `Junction` class contains all of the logic related to each junction in the network. Each `Junction` contains an arbitrary number of incoming and outgoing roads as well as an arbitrary number of traffic lights and coupled traffic lights. The theory behind the general n-to-m junction is discussed in Section 4.1.5. The `Junction` class combines this theory with potentially multiple traffic lights that modify the incoming fluxes to the junction depending on the state of the traffic lights. As described in Section 4.1.5, the general junction need a distribution matrix A as well as a priority ordering of the connections leading into outgoing road r_j for each of the outgoing roads. The distribution matrix A takes the form

$$A = \begin{bmatrix} \alpha_{1,1} & \cdots & \alpha_{1,m} \\ \vdots & \ddots & \vdots \\ \alpha_{n,1} & \cdots & \alpha_{n,m} \end{bmatrix},$$

and when the `Junction` object is being initialized, it is checked that the condition

$$\sum_{j=1}^{m} \alpha_{i,j} = 1 \quad \forall i = 1, \cdots, n.$$

Instead of specifying the right-of-way parameters, these will be specified using the densities on the roads. Nevertheless, for each outgoing road r_j, a priority ordering of the incoming roads is needed. This priority ordering is summarized in a matrix that could take the following form when $n = m = 3$

$$\begin{bmatrix} 1 & 0 & 2 \\ 2 & 1 & 1 \\ 0 & 2 & 3 \end{bmatrix}.$$

If element $(1, 1)$ is equal to one, this means that incoming road 1 has priority over the other incoming roads leading into outgoing road 1. Element $(1, 2)$ being equal to zero, means that cars coming from incoming road 1 cannot enter outgoing road 2.

Finally, when creating a `Junction` object, the connections of higher priority being crossed need to be specified for each connection between incoming road l_i and outgoing road r_j. If $n = m = 3$, the matrix specifying the crossing connections could, e.g., take the form

$$\begin{bmatrix} \cdot & \cdot & (2,2) \\ \cdot & \cdot & \cdot \\ \cdot & (1,1) & [(1,1),(2,2)]. \end{bmatrix}$$

Element $(1, 3)$ being equal to $(2, 2)$, means that traffic from incoming road 1 wanting to go to outgoing road 3 crosses the traffic from incoming road 2 to outgoing road 2. The traffic on the connection being crossed will then be used to calculate an upper bound on the traffic from incoming road 1 to outgoing road 3 as described in Section 4.1.5. When initializing the junction object, all of the road objects that appear in the junction need to specified. In addition, an ordering of which of the roads are entering and which of the objects are leaving is needed. Finally, the distribution matrix, the priority ordering and the crossing connections are stored as member variables.

Methods

As for the road class, a number of internal methods are needed in order to handle the logic related to each junction. The most import method is the method that calculates the incoming and outgoing fluxes to the junction at each time step of the simulation. As each `Junction` object contains references to all of the `Road` objects that are connected to the junction, to calculate the crossing fluxes no extra arguments are needed.

As we saw in Section 4.1, the calculation of fluxes across a general junction may be complicated.

Table 6.2: Member variables of the `Junction` class.

A :	Distribution matrix
priorities :	Priority ordering matrix
crossings :	Matrix containing crossing connections
roads :	List of references to `Road` objects
entering :	List of indexes of incoming roads
leaving :	List of indexes of exiting roads
trafficlights :	List of references to `TrafficLight` objects
coupled :	List of references to `CoupledTrafficLight` objects

Table 6.3: Member variables of the `TrafficLight` class.

s_{start} :	Starting state of the traffic light
entering :	List of indexes of incoming roads affected by the light
leaving :	List of indexes of exiting roads affected by the light
cycle :	List of times specifying the behaviour of the light

6.1.3 The Traffic Light Class

The `TrafficLight` class is a part of a junction, and contains a subset of all of the roads that are connected to the junction. This class does not actually keep a reference to the roads themselves, but rather their indices in the full list of roads that is kept in the `Junction` class. The simple traffic light contains a list of the incoming roads and outgoing roads. This traffic light is used to determine how much of the traffic from the specified subset of incoming roads is allowed to enter the specified subset of the outgoing roads at the evaluation times of the simulation. The traffic from each incoming road of a junction can either depend on no traffic lights or potentially several traffic lights. Although possible, each connection between incoming road l_i and outgoing road r_j should not depend on multiple traffic lights.

Each traffic light changes states between red and green according to a prescribed cycle. The cycle consists of a list of times at which the current state is stayed at before it changes. For each cycle to be equal, the number of times should be even.

Methods

The main method of the traffic light is the activation function that returns a number in the interval $[0, 1]$ corresponding to the percentage that is allowed to cross the connections that are depending on the traffic light. Upon the initialization of the full road network, this activation function will constructed

Table 6.4: Member variables of the CoupledTrafficLight class.

s_{start} :	Starting state of the two traffic light
a-entering :	List of indexes of incoming roads affected by the light a
a-leaving :	List of indexes of exiting roads affected by the light a
b-entering :	List of indexes of incoming roads affected by the light b
b-leaving :	List of indexes of exiting roads affected by the light b
cycle :	List of times specifying the behaviour of the light
t_{delay} :	Time period both traffic lights are red

using the cycle of the traffic light as explained in Section 4.2.

6.1.4 The Coupled Traffic Light Class

The `CoupledTrafficLight` class is mostly the same as the `TrafficLight` class, with the main difference being that it models two traffic lights coupled together, instead of one independent traffic light. When one light is green, the other is red and likewise (except for the period where both traffic lights are red).

Methods

As for the single traffic light, the main method of the `CoupledTrafficLight` class is its activation function that returns a number in the interval $[0, 1]$ corresponding to the percentage that is allowed to cross the connections that are depending on the traffic light. The difference is that instead of containing only one such activation function, the `CoupledTrafficLight` contains two activation functions. Upon the initialization of the full road network, these activation function will be constructed using the cycle of the traffic light following Section 4.2.

6.1.5 The Bus Class

Busses travelling through the road network are modelled using the `Bus` class. At all points in time, each bus has a certain position. Instead of tracking these positions in 2d space, we map the position of a bus to a corresponding length of a 1d route. The total length of this route is equal to the sum of all the roads the bus travels on.

Each bus travels along some predetermined route. As each `Road` object also has a unique id, this route is given by a list of id's, each relating to a specific `Road` object. In addition, we need to know the locations of the bus stops along the route. The stops are given as a list of tuples, where each tuple contains the id of a `Road` object and the length along this `Road` where the stop is located. We note that we could alternatively give the list of stops simply as a list of

Table 6.5: Member variables of the `Bus` class.

ids :	List of ids of `Road` objects
stops :	List of tuples where the first element is the id of a `Road` object and the second element is the position on this road where a stop is placed
times :	List of times at which the bus should reach a stop
t_{start} :	The time the bus reaches the network

lengths along the total route, but to avoid having to map between length on a route to specific `Road` objects, we also list the ids.

To estimate the delays of the busses, we also need to know at which times the bus should arrive at the different stops. Thus, each `Bus` object will contain a list of times, each element referring to the expected arrival time at a stop.

Finally, we may not want all busses to start at time $t = 0$. Therefore, we also specify a starting time t_{start} at which time the bus will enter the network. The necessary member variables of the `Bus` class are listed in Table 6.5.

Methods

At every time step of the simulation, the positions of all `Bus` objects must be updated. The speed the bus is travelling at is computed using the `Road` object it is currently in, and will not be computed by an internal method of the `Bus` class, but rather by a method of the `RoadNetwork` class. Although the `Bus` class contains minor methods for mapping between roads and length on the route, the most important is the method for updating the position. As mentioned, the speed of the traffic close to the bus will be calculated elsewhere, before being passed as an argument. However, as discussed at the end of Section 4.4, the speed might need to be updated if the bus is close to a bus stop or a junction. The modification of the speed, as well as the actual updating of the position will be handled by the main method of the `Bus` class.

6.1.6 The Roundabout Class

As a roundabout is more or less just a concatenation of simple junctions, it would be possible to add roundabouts to the network just by modifying the junction class slightly and then adding the junctions that make up the roundabout one by one. Nevertheless, since we use different rules for the junctions of the roundabouts (see Section 4.3), we make a special class. All roundabout junctions we consider are 2-to-2 junctions following a FIFO-rule. Hence, the implementation of the roundabout junctions can be much less general than the `Junction` class needed to be. Further, we only allow a `Junction` object to contain `Road` objects that actually are a part of the simulation. To the contrary,

Table 6.6: Member variables of the `Roundabout` class.

mainline-in :	Mainline incoming `Road` object
mainline-out :	Mainline outgoing `Road` object
second-in :	Secondary incoming `Road`/`RoundaboutRoad` object
second-out :	Secondary outcoming `Road`/`RoundaboutRoad` object
α :	Distribution parameter

the roundabout junctions may contain secondary incoming and outgoing roads where the densities are not being tracked. The only thing needed is some way of knowning the influx/outflux of these secondary roads.

The flux distribution of roundabouts can be written in an explicit form, and to calculate it, the `Roundabout` class contains a reference to the incoming and outgoing mainline roads, as well as the secondary incoming and outgoing road. The secondary roads may not be a part of the network we are considering, and so we create a small `RoundaboutRoad` class to model these roads. This class only contains a function describing the incoming traffic and a queue length modelling the cars waiting to enter the roundabout. The final member variable of the `Roundabout` class is a distribution parameter α describing the distribution of traffic. We summarize the member variables in Table 6.6.

Methods

The junctions of a roundabout are a simple type of junctions. The most import method of the `Roundabout` class is a method that calculate the flux across the roundabout junction following Section 4.3. Since the optimal distribution of fluxes can written in a more explicit way for roundabout junctions, the internal method of the `Roundabout` class will call fewer external functions.

6.1.7 The Road Network Class

The `RoadNetwork` class handles the logic related to the full simulation. This class contains references to all components that are part of the network: a list of references to all `Road` objects, a list of references to all `Junction` objects and a list of references to all `Roundabout` objects. Since the traffic lights only impact the junction they are a part of, this class does not need to keep references to the `TrafficLight` objects. The `RoadNetwork` class also contains references to all `Bus` objects of the network, as well as the total time to be simulated T. A summary of the necessary member variables is listed in Table 6.7.

Methods

The main method of the `RoadNetwork` is the method that simulates the network until the final time is reached. The pseudo-code for this method can be seen

Table 6.7: Member variables of the `RoadNetwork` class.

roads :	All `Road` objects in the network
junctions :	All `Junction` objects in the network
roundabouts :	All Roundabouts of the network
busses :	All busses of the network
T :	Simulation period

Algorithm 3. The method starts by initializing $t = 0$ before determining the time step Δt. Then it goes through all of the busses in the system and calculates the length each of them should travel. Here, the impact of the bus on the traffic is calculated as discussed at the end of Section 4.4. Then, the fluxes across each of the junctions and across each of the roundabout junctions are computed as discussed in Sections 4.1 and 4.3. Finally, the densities on the internal cells of each `Road` object are updated and the boundary cells are updated either using the incoming fluxes from junctions or by using suitable boundary elements.

Algorithm 3 Solve Conservation Law

```
1: Set t = 0
2: while t < T do:
3:     t_c = calculate next control point
4:     while t < t_c do
5:         Δt = min{t_c − t, get max dt from CFL}
6:         for bus in network do
7:             Update position of the bus
8:             Update Δt using the slowing of traffic due to the bus
9:         for junction in network do
10:            Calculate flux through junction
11:            Update Δt using the fluxes through the junction
12:        for roundabout in network do
13:            Calculate fluxes through each junction of the roundabout
14:            Update Δt using the calculated fluxes
15:        for Road in network do
16:            Solve cons-law internally on road using FV scheme
17:            Calculate fluxes on edges not connected to junctions or roundabouts
18:            Update boundary cells using incoming/outgoing fluxes
```

6.1.8 Remarks on the Code Implementation

Our implementation of a road network model consists of a number of classes associated to the different components of the road network we are modelling.

When modelling a specific road network, there is thus a large number of parts the user has to specify. The user has to, e.g., accurately specify all yielding rules for each junction separately. In addition, the exact relations between all roads and junctions need to be specified. While this implementation requires some work to model a specific road network, it also makes the simulation very flexible. There are no limitations, except for computational or memory related limitations, on the number of roads, junctions or roundabouts that can be modelled. In addition, for each junction, there are no upper limit on the number of incoming or outgoing roads. There are also no limitations to the number of bus lines the simulation is able to handle, except for computational or memory related ones. Further, due to its modular implementation, it is easy to extend or modify the model. If one wants to make use of other yielding rules for the distribution of traffic through a junction, this can be done by changing one method called inside the `Junction` class.

6.2 Implementation of the Optimization Method

Now that we have an overview of how the different components of the model were implemented, we will discuss how the solution approach to the optimal control problem was implemented. In all of our numerical experiments, we used the gradient descent algorithm as discussed in Section 5.2.1. The basic steps of this optimization method can be seen in Algorithm 4. One step entails

Algorithm 4 Optimize Control Parameters

1: T, p^0, p_{lower}, p_{lower} given
2: Set $p = p^0$
3: **while** stopping criteria not reached **do**:
4: Initialize `RoadNetwork` object from parameters p
5: Simulate until time T
6: Calculate objective function and gradient
7: Update gradient according to (5.20)
8: **while** line search criteria not reached **do**
9: Update step length
10: Set $p = p + \alpha\vec{p}$

computing the gradient of an objective function with respect to the control parameters, which is done using the AD capabilities that come with PyTorch. However, as we will see in the next subsection, we have some freedom as to how we want to compute the gradient.

6.2.1 AD Implementation

As discussed in Section 3.2, there are two different modes that can be used to compute a gradient using AD. In our case, all of the objective functions

considered in Section 5.1 are functionals; that is, given the input $p \in \mathbb{R}^n$, $n \geq 1$ they all return scalar output. According to the discussion at the end of Section 3.2 the computational effort of computing the gradient in the forward mode scaled with the number of input variables n, and the computational effort of computing the gradient in the backward mode scaled with the number of out variables, which is one in our case. Hence, when $p > 1$, it is most likely more efficient to compute the gradient in backward mode.

The drawback we experienced when using the backward mode, is that the memory usage grows with the time horizon of the simulation. Hence, the longest possible simulation horizon is restricted by the RAM of the computer running the simulation. In contrast, although less efficient, computing the gradient using the forward mode did not pose the same memory issues.

Still wanting to benefit from the efficient backward mode, we try to find a way to get rid of the memory issues. The objective function being used for most of the numerical experiments is (5.6), meaning that we try to reduce the average delay of busses. The idea we use to reduce the overall memory usage is to try to free the memory at certain parts of the simulation. Natural such points when considering the objective function (5.6) are every time a new bus stop is reached. Following this idea, we start from $t = 0$ and simulate until a bus reaches a bus stop. When a stop is reached, the delay is calculated, and the gradient is computed using the backward mode. Then all evaluation traces are reset to free the memory, before simulating until a new stop is reached. At this point, we exploit that since the objective function contains the sum of all bus stops, the total gradient is equal to the sum of the gradients computed using only one delay. Thus, we proceed until the end of the simulation is reached, incrementing the gradient every time a new bus stop is reached.

We note that the final gradient will no longer be exact. Every time the evaluation trace is reset, some of the dependence on the control parameters is forgotten. Nevertheless, we hope that this approximate gradient is good enough that a good choice of control parameters can still be found. We will check whether this is the case or not in Chapter 7.

Chapter 7

Numerical Experiments

In this chapter we will consider a number of different numerical experiments that show the different components of the model. We will start by simple networks that are well suited to show parts of the model, before increasing the complexity in order to investigate how the model behaves for larger networks. We will investigate the smoothness of the objective function (5.6) for some networks and check the convergence rate of the solution of the model. In addition, we will search for the optimal set of control parameters of different networks using several of the objective functions from Section 5.1.

7.1 Single Bus on a Single Road

We start by considering the simple case of one bus travelling on a unidirectional road. A visualization of the network can be seen in Figure 7.1. The length of the road is 200 metres, with a speed limit that is bounded between 20 km/h and 50 km/h. The high-resolution scheme (2.24) with the Rusanov flux (2.23) was used for updating internal cells. The spatial length of each cell is chosen as $\Delta x = \frac{50}{7} \approx 7.143$. The initial density on the road was chosen to be constant at 40% of the maximum density. The incoming flux to the road is chosen to be equivalent to the flux between two cells on this road if the density on the cells is equal to 40% of the maximum density if the speed limit was 40 km/h. That is, the incoming flux is calculated as

$$f_{\text{in}} = \frac{v}{L}\rho(1-\rho) = \frac{40/3.6}{50}0.4(1-0.4) = \frac{4}{75} \approx 0.053.$$

We assume that all traffic arriving at the end of the road is allowed to exit; that is, we do not impose any conditions on the outgoing flow of traffic. We

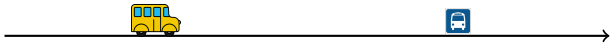


Figure 7.1: Road network with a single road, a single bus and a single bus stop.

Table 7.1: Convergence history of the optimization method starting from $v_{\max} = 40$km/t and $v_{\max} = 20$km/t using objective function (5.6).

Iteration	$v_{\max}$	$J(p)$
0	40	26.272
1	**50**	**21.018**
0	20	52.545
1	40	26.272
2	**50**	**21.018**

also set the maximal density to be equal to 1, meaning that the road only has one lane.

To ensure that there is a delay that can be minimized, we assume that the bus is already delayed when it enters the road. This is done by specifying in the schedule for the bus that the stop should be reached at time $t = 0$, while at the same time starting the simulation at time $t = 0$ with the bus positioned at left edge of road l. We let the bus stop be positioned at length 150 metres. The only control parameter we can choose in this case is the speed limit, which we will assume to be constant in time. Using the objective function (5.6), the best speed limit is the one that makes the bus arrive at the bus stop as quickly as possible.

We now want to check what choice for the speed limit we obtain at the end of the optimization procedure. We search for the optimal solution using the modified gradient descent described in Section 5.2.1. We start the optimization procedure from speed limits 20 km/h and 40 km/h to check if the same optimum is reached, and if so, how quickly the optimum is reached.

The results from the optimization can be seen in Figures 7.2 to Figure 7.4. Figure 7.2 shows the path taken to reach the optimal speed limit as a function of the number of iterations. For both starting points, the speed limit 50 km/h is found. When starting with the speed limit equal to 40 km/h, the optimum was reached after only one iteration, but when starting from 20 km/h, two iterations was necessary. In Figure 7.3 the distance travelled by the bus is visualized as a function of time for both of the two starting points, and for the optimal speed limit $v_{\max} = 50$ km/h. Finally, Figure 7.4 shows the reduction of the bus delay as a function of the number of iterations from the two starting points. We also summarize the results from the iteration in Table 7.1, where we include exact numbers for the delay of the bus. A video comparing the flow of traffic with speed limit $v_{\max} = 20$km/h and $v_{\max} = 50$km/h can be seen at `torjeihelset.github.io/master_project`. Snapshots of some of the times of the simulation are included in Figure 7.5.

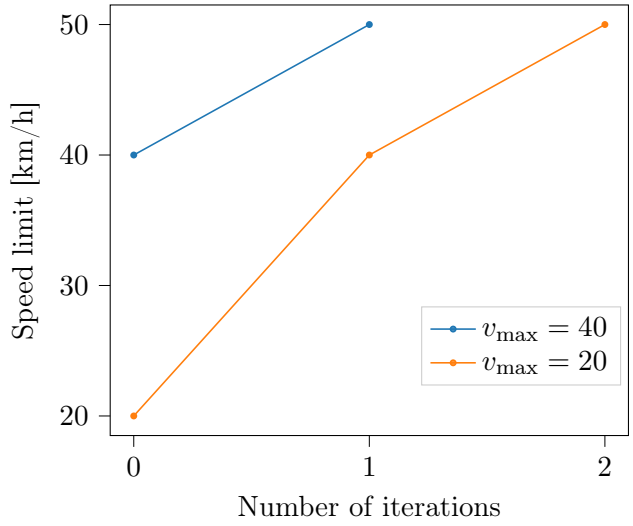


Figure 7.2: The best speed limit as a function of the number of iterations.

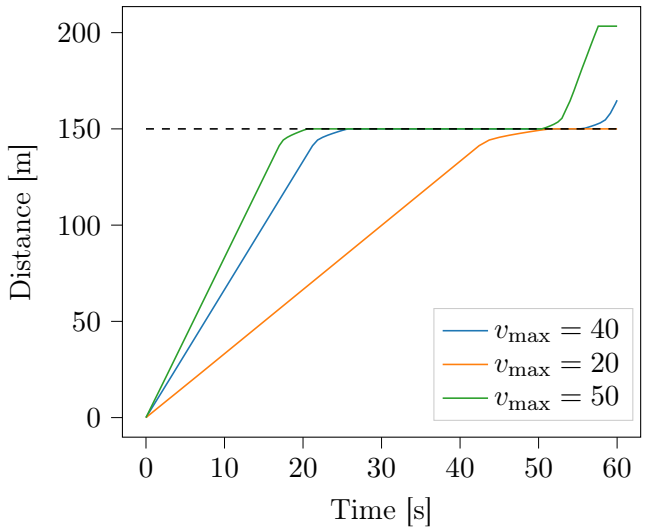


Figure 7.3: Distance travelled by the bus with different choices of speed limit. The dotted line marks the distance to the bus stop.

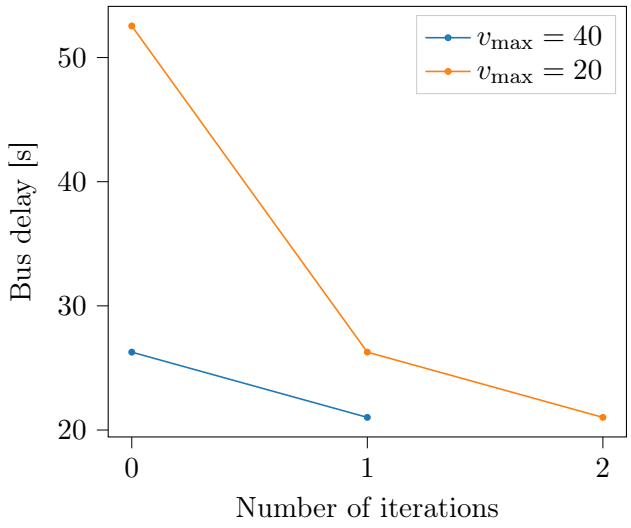


Figure 7.4: Objective values as a function of the number of iterations.

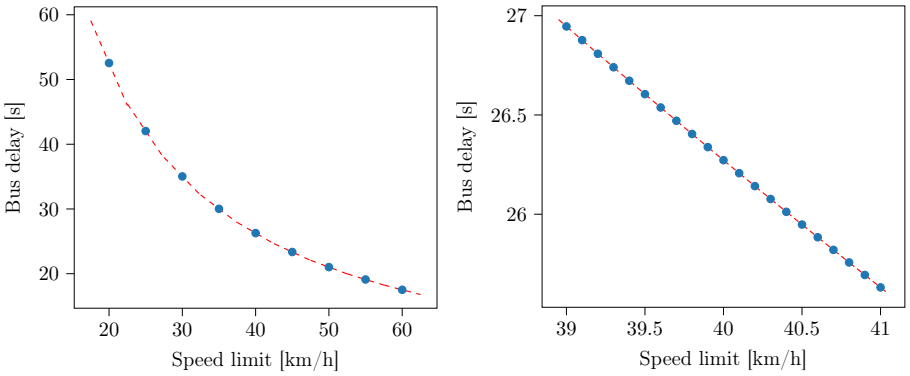
(a) t = 7

(b) t = 17

(c) t = 25

(d) t = 69

Figure 7.5: Snapshots of the simulation for the optimal speed limit of the single lane network. The amount of traffic is visualized with a red color scale. White means that there is no traffic, and a darker red color means that the road is congested. The green square shows the position of the bus with optimal speed limit. The blue square shows the position of the bus with non-optimal speed limit.



(a) Increment of 5 km/h.

(b) Increment of 0.1 km/h.

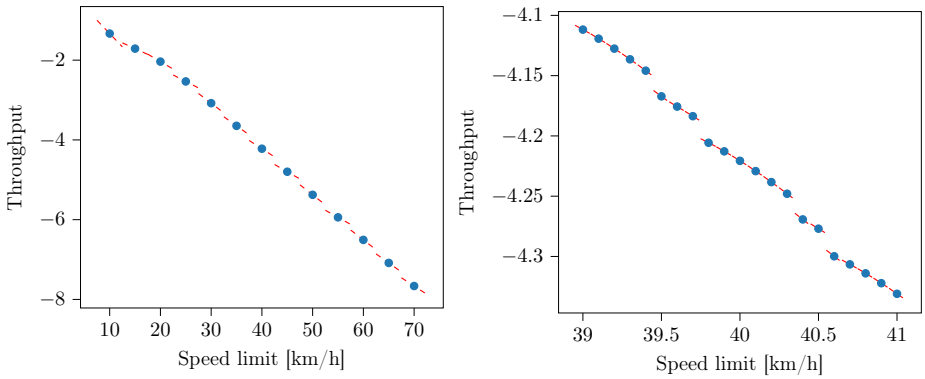
Figure 7.6: Bus delays for different speed limits. Increments of 5 km/h and 0.1 km/h respectively.

Properties of the objective function

We want to get an idea of the properties of the objective function of this network as well as the computed gradient. We vary the speed limit around 40 km/h to check how the objective function varies. We do this for the three objective functions (5.6), (5.3) and (5.5). We visualize the results in figures where the objective values at the evaluation points are marked by blue dots, and the tangent lines are shown as red dashed lines. The slopes of the tangent lines at each evaluation point are computed using automatic differentiation. Figure 7.6 show how the delay of the bus varies with the speed limit when varying the speed limit with increments of 5 km/h and 0.1 km/h respectively. From this figures, it looks like the objective function (5.6) is a strictly decreasing function for this network. This also matches the optimal speed limit from Table 7.1, where the best speed limit was its upper bound. The minimum of a strictly decreasing function is at its right endpoint, which is what we observed when determining the best speed limit.

Figure 7.7 shows how the objective function (5.3) varies with the speed limit with increments of 5 km/h and 0.1 km/h respectively. When varying the speed limit with increments of 5 km/h the objective function again looks like a decreasing function. However, when decreasing the increments between evaluation points, we observe that the objective function appears to have some jumps that were not visible when the evaluation points were separated by 5 km/h. Although the objective function in this case is more irregular, it still looks to be a decreasing function that is well suited for a gradient-based optimization method.

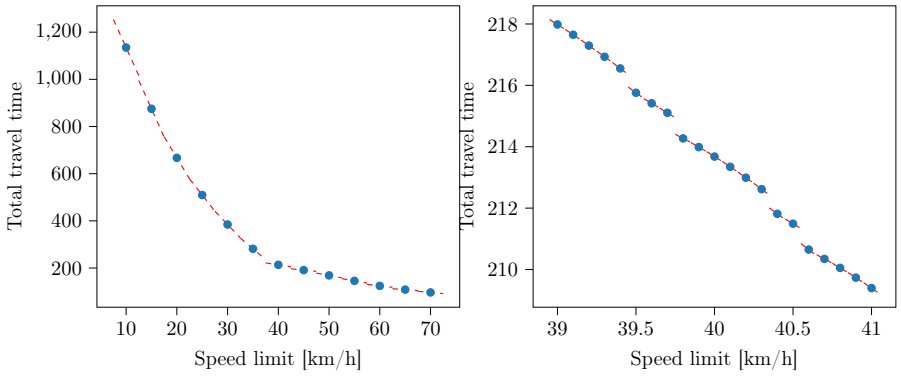
Finally, we consider the objective function (5.5). We see from Figure 7.8 that the objective function again looks like a decreasing function, with a change in slope somewhere between 30 and 40 km/h. When decreasing the increments



(a) Increment of 5 km/h.

(b) Increment of 0.1 km/h.

Figure 7.7: Total throughput from single lane for different speed limits. Increments of 5 km/h and 0.1 km/h respectively.



(a) Increment of 5 km/h.

(b) Increment of 0.1 km/h.

Figure 7.8: Total travel time of single lane for different speed limits. Increments of 5 km/h and 0.1 km/h respectively.

between evaluation points, we again observe some irregularity of the objective function. Nevertheless, since the function still looks to be a decreasing function, also in this case a gradient-based optimization method should be able to reach the global optimum. As we shall come back to in Section 7.2, we believe that this more irregular behaviour is mainly a side effect of insufficient numerical accuracy.

7.2 Single Bus Crossing a Junction

We now consider a slightly more interesting case of two unidirectional roads connected by a junction with a traffic light. Denoting the two roads by l and r as in Figure 7.9, we assume that a single bus starts on road l and stops halfway on road r before exiting the network. The length of each of the roads is 100 metres, both having speed limits that are bounded between 20 km/h and 50 km/h (that may be set independently of each other). The high-resolution scheme (2.24) using the Rusanov flux (2.23) was used for updating the densities on the internal cells. The spatial length of each cell is chosen as $\Delta x = \frac{50}{7} \approx 7.143$ for each of the cells on the road. The initial density on the road was chosen as 40% of the maximum density. The incoming flux to the road is chosen to be equivalent to the flux between two cells on this road if the density on the cells is equal to 40% of the maximum density if the speed limit was 40km/h. That is, the incoming flux is calculated as

$$f_{\text{in}} = \frac{v}{L}\rho(1-\rho) = \frac{40/3.6}{50}0.4(1-0.4) = \frac{4}{75} \approx 0.053.$$

We also set the maximal density to be equal to 1, meaning that the road only has one lane.

As before, we assume that the bus is already delayed when it enters the network. This is done by setting the scheduled arrival time at the stop at time $t = 0$, and starting the bus from the left edge of road l. Assuming that the traffic light can be described by a cycle consisting only of the time it stays red and the time it stays green, and that the traffic light starts out being red, we have four control parameters that we may choose. We can choose the constant speed limits on each of the two roads, and we may also choose for how long the traffic light stays red and green respectively. We assume further that the light can stay in one state for at most 200 seconds, and that it at must stay in each of states for at least 10 seconds. Since the junction only connects two roads, there is no need for any distribution or priority parameters.

We now try to find the control parameters minimizing the objective function (5.6). We start the optimization from the two different starting points listed in Table 7.2 and use the modified steepest descent method discussed in Section 5.2.1 to obtain locally optimal solutions. Figure 7.10 shows how the bus travels much faster through the two roads with the optimized parameters compared with the two starting parameters. In Figure 7.11, the reduction

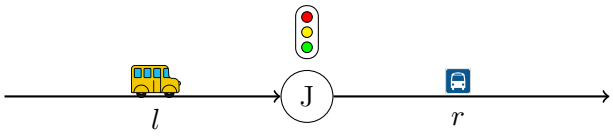


Figure 7.9: Road network of two road connected by a traffic light with bus stopping on second road.

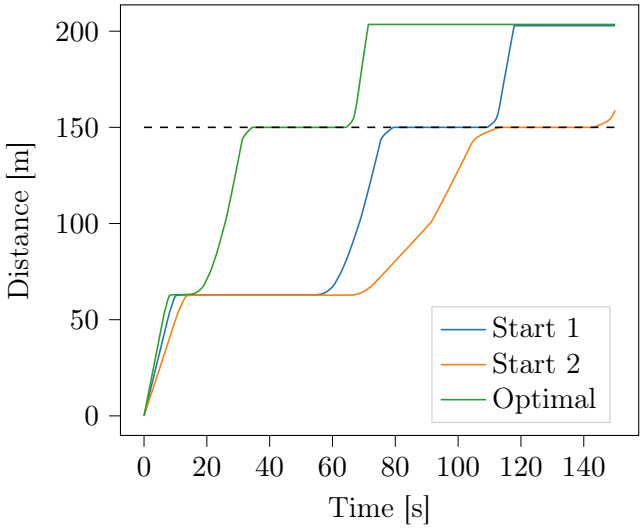


Figure 7.10: Distance travelled by the bus with different choices of speed limits for the two roads. The dotted line marks the distance to the bus stop.

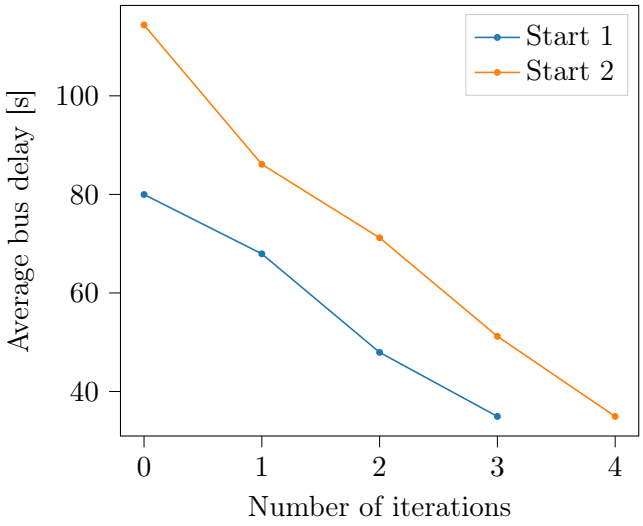


Figure 7.11: Objective values as a function of the number of iterations.

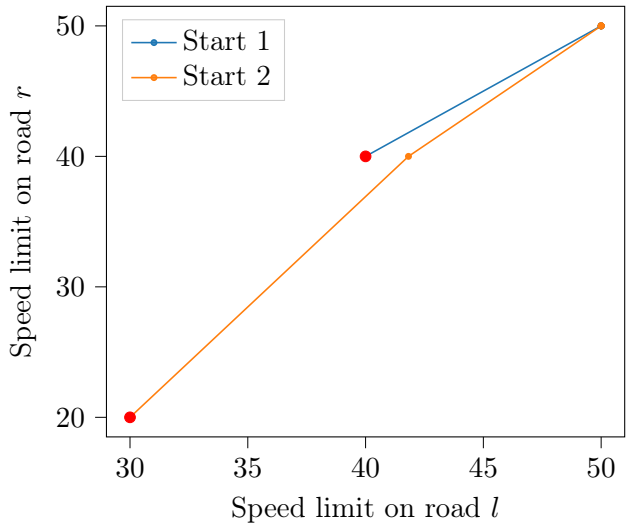


Figure 7.12: The path taken towards the optimal speed limits from two different starting points.

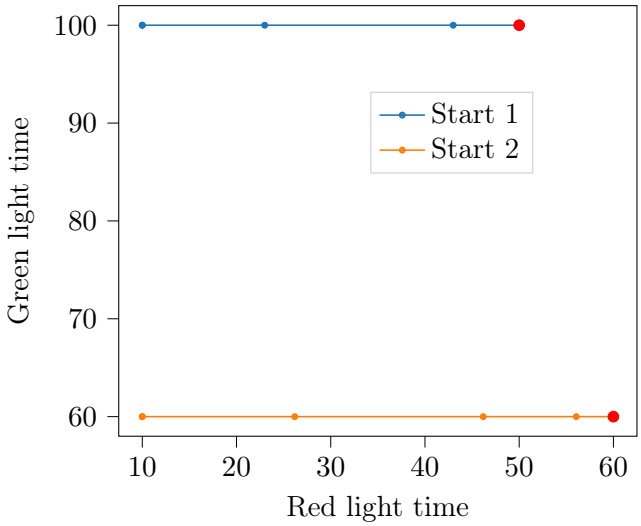


Figure 7.13: The path taken towards the optimal traffic light cycle from two different starting points.

Table 7.2: Starting positions for the case with a single bus crossing a junction.

Start Id	l_1	r_1	t_1	t_2	s_{start}
1	40	40	50	100	0
2	30	20	60	60	0

in the objective value as a function of the number of iterations is displayed for the two staring points. We see that the same objective value was reached for the two starting points after three and four iterations, respectively. The path taken towards the optimal speed limits is shown in Figure 7.12, with the starting points marked by a red circle. Although it seems that only one and two iterations were taken, this is due to the fact that the boundary values were reached after one and two iterations. The path taken towards the optimal control parameters for the traffic light is shown in Figure 7.13 for the two starting points, again with the starting points marked by red circles. We see from the figure that only the parameter for the time the light should remain red is updated, and that this parameters reached the boundary at the end of the optimization algorithm.

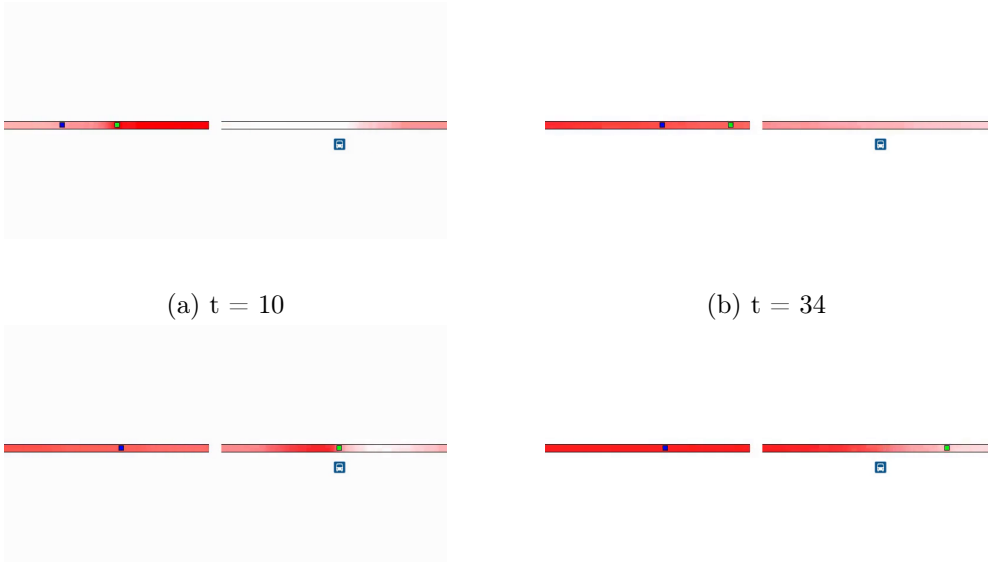
We recall that the traffic light is initialized to be red, meaning that reducing the time it stays red will mean that the bus can cross the junction earlier. The optimal choice of control parameters is then to put the speed limits as high as possible, and to reduce the time the traffic light stays red. It makes sense that the other parameter is not updated, as the bus does not care about how long the light stays green after it has crossed the junction. The full convergence history is listed in Table 7.3, where the optimal solutions found from the two starting points are marked in bold text. We observe that both speed limits, as well as the time period the light should remain red are equal for the two optima. Although the second time period t_2 is chosen differently, we see from the objective values that these are equivalent solutions. A video showing the difference in the flow of traffic when using optimal control parameters and when using the control parameters of the starting point 1 can be found at `torjeihelset.github.io/master_project`. We include some snapshots of this video in Figure 7.14.

Properties of the objective function

As for the single lane, we want to investigate the properties of the different objective functions. The results can be seen in Section C.1. Figure C.1 and Figure C.2 show how the delay of the bus is affected by the settings of the traffic light. We observe that only the first time of the traffic light impacts the delay time. This matches well with the optimization results in Table 7.3, which show that only one of the two times of the traffic lights was updated. Similar to as for the single lane network, when reducing the increments between

Table 7.3: Convergence history of the optimization method from the two starting points listed in Table 7.2 using objective function (5.6).

Start	**Iteration**	l_1	r_1	t_1	t_2	$J(p)$
1	0	40.0	40.0	50.0	100.0	79.9758
1	1	50.0	50.0	43.0	100.0	67.9417
1	2	50.0	50.0	23.0	100.0	47.9417
1	**3**	**50.0**	**50.0**	**10.0**	**100.0**	**34.9422**
2	0	30.0	20.0	60.0	60.0	114.3715
2	1	41.8	40.0	56.1	60.0	86.1101
2	2	50.0	50.0	46.2	60.0	71.1978
2	3	50.0	50.0	26.2	60.0	51.1978
2	**4**	**50.0**	**50.0**	**10.0**	**60.0**	**34.9422**



(a) t = 10

(b) t = 34

(c) t = 49

(d) t = 95

Figure 7.14: Snapshots of the simulation for the optimal control parameters of the single junction network. The amount of traffic is visualized with a red color scale. White means that there is no traffic, and a darker red color means that the road is congested. The green square shows the position of the bus with optimal control parameters. The blue square shows the position of the bus with non-optimal control parameters.

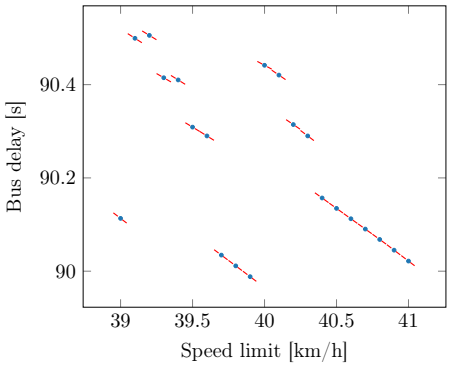
evaluation points some irregularity of objective function is revealed. Despite these irregularities, the objective function looks to be an increasing function with a global minimum at its left endpoint. Figure C.3 and Figure C.4 show how the delay of the bus is affected by the speed limits of the two roads. For the speed limit on the second road r, the objective function looks to be a decreasing function, although slightly irregular. For the first road however, when decreasing the increment between evaluation, the objective function looks more irregular and no longer is monotone, concave or convex. Although this behaviour likely makes optimization more difficult, we remark that we are updating several control parameters at the same time, and so even if there is some irregularity related to one of the parameters, it might not be a major problem for the overall optimization procedure. We also remark that both the speed limit on the second road as well as the settings of the traffic light affect the bus delay to a higher degree, and so these parameters might dominate the optimization procedure.

One of the cases where the objective function looked highly irregular was when varying the speed limit on road l with increments of 0.1 km/h. We want to get a better understanding of the cause of these irregularities. Therefore, we perform the simulation with the same values for the speed limit on road l, but this time with a much finer spatial grid, and with a much smaller time step. The comparison can be seen in Figure 7.15, where we also include the original graph. For the finer grid, we use a spatial length of $\Delta x = \frac{50}{84} \approx 0.595$ metres. Figure 7.15b shows that when reducing the spatial length, the difference between maximum and minimum values decrease. The irregularities of the objective function still remain however. We try to instead reduce the time steps of the simulation. This is done by modifying the CFL condition (2.22) by using a factor 1/20, instead of 1/2. That is, we use

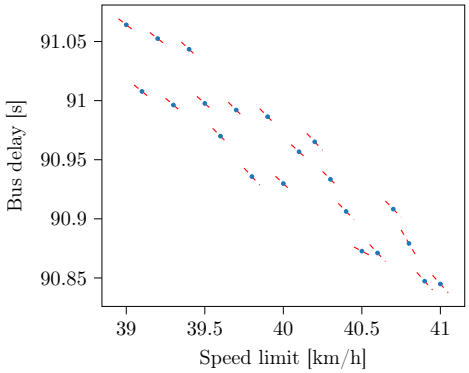
$$\nu^n = \max_j |f'(\rho_j^n)| \frac{\Delta t}{\Delta x} \leq \frac{1}{20}$$

to choose the time step. For this comparison we make use of the original spatial length $\Delta x = \frac{50}{7} \approx 7.143$ metres. The results can be seen in Figure 7.15c. We see that also by decreasing the time steps, the difference between the maximum and minimum value decreases. More interestingly, most of the irregularity has been eliminated, and the objective function now more closely resembles a monotone function. From this comparison, it seems that at least for this network and this objective function, the irregularity of the objective function is mostly due to numerical errors which can be eliminated by reducing the time step Δt. We note that the position of the bus is updated only at the time steps, and since the objective functional depends directly on the position of the bus, it seem reasonable that by reducing the time step, the irregularities of the objective functional is reduced. The irregularities of the objective function that we see is likely close to the numerical accuracy of the simulation.

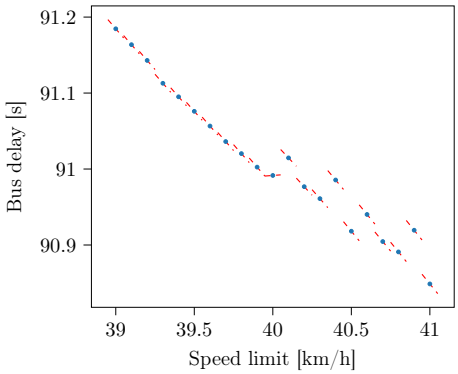
We also investigate the two other objective functions (5.3) and (5.5). The



(a) Original.



(b) Reduced spatial length.



(c) Reduced time steps

Figure 7.15: Comparison of the bus delay for different speeds on road l with increments of 0.1 km/h when using a finer spatial grid and when using smaller time steps.

rest of the figures in Section C.1 show the properties of these objective functions. The results are similar to the ones for (5.6). When decreasing the increments between evaluation points, more irregularity of the objective functions is visible. We again believe that these irregularities are mainly due to insufficient numerical accuracy. Although for this network it looks like the objective functions are monotone when varying the most of the control parameters, Figure C.9 indicates that there might a minimum for the second time of the traffic light somewhere between 50 and 50 seconds when considering the objective funciton (5.5).

Table 7.4: Initial parameters describing the roads of the 2-to-2 network.

Road	L [m]	ρ_{init}	$\rho_{\max}$	$v_{\min}$ [km/h]	$v_{\max}$ [km/h]
l_1	100	0.4	1	20	50
l_2	100	0.4	1	20	50
r_1	100	0.4	1	20	50
r_2	100	0.4	1	20	50

Table 7.5: Influx of traffic to 2-to-2 network. The influx is calculated as $f_{\text{in}} = \gamma\rho(1-\rho)$, with $\gamma = \frac{v_{\text{in}}}{L}$.

Variable	ρ	v_{in} [km/t]	L [m]
F_1	0.4	40	50
F_2	0.4	40	50

7.3 Two Busses Travelling Through a 2-to-2 Junction

In this section we consider a slightly more complicated network, which consists of four roads connected in a 2-to-2 junction. Following the notation in Figure 7.16, we denote the two incoming roads by l_1 and l_2, and the two outgoing roads by r_1 and r_2. We assume that all of the traffic from road l_1 enters road r_1 and that all the traffic from road l_2 enters road r_2. All four roads have lengths of 100 metres and maximum densities of 1, meaning that they all model single-lane unidirectional roads; see Table 7.4 for full parameter configuration. The influx of traffic to the two roads is listed in Table 7.5. The junction is controlled by a coupled traffic light defined in such a way that when the traffic from road l_1 is allowed to cross the junction, no traffic from road l_2 is allowed to enter the junction. After the light switches from green to red, there is a five second period where no traffic crosses the junction. As before we choose the spatial length of each cell to be $\Delta x = \frac{50}{7} \approx 7.143$ for all roads, and we update the internal densities using the high-resolution scheme (2.24) with the Rusanov flux (2.23).

Since we in this case have two outgoing roads, we need a distribution matrix. We let this matrix be defined as

$$A = \begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{bmatrix},$$

so that all the traffic coming from road l_1 enters road r_1, and all the traffic from road l_2 enters road r_2. Because of the coupled traffic light, the traffic from road l_1 will never meet traffic from l_2 in the junction, and so there is no need for right-of-way parameters. Nor is there any need to consider yielding rules.

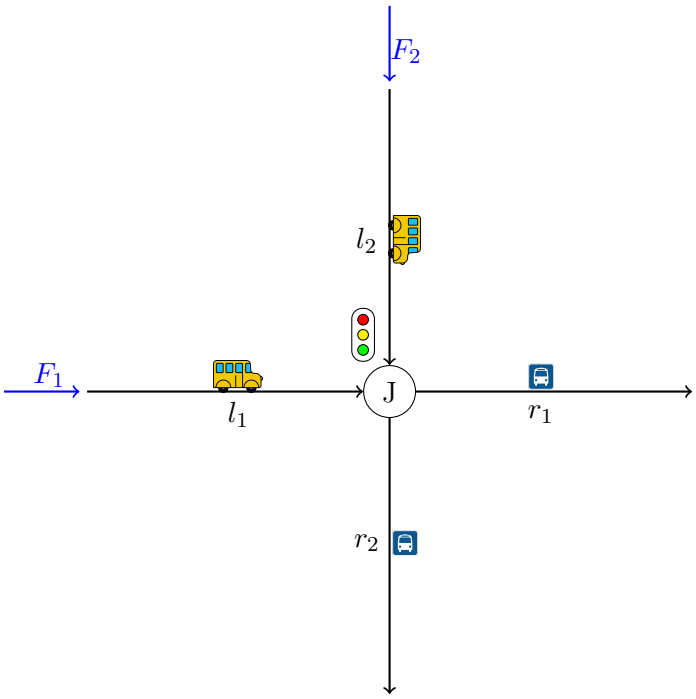


Figure 7.16: Graphical visualization of a road network consisting of four roads connected in a 2-to-2 junction with a coupled traffic light. Two busses are travelling through the network and there are two bus stops on roads r_1 and r_2.

We consider two busses travelling from l_1 to r_1 and from l_2 to r_2, respectively. The two busses stop halfway on road r_1 and r_2, and we assume that both busses are delayed before entering the junction, by setting the expected arrival times at the stops as $t = 0$ for both busses. Since the traffic through the junction is controlled by a single coupled traffic light, it suffices with a single cycle of times, as explained in Section 4.2. In this experiment, we assume that the cycle only consists of only one time for each of the two states. We also assume that we can set the speed limits on the four roads independently of each other, resulting in a total of six control parameters that we need to determine. Assuming that all speed limits of the roads are bounded between 20 km/t and 50 km/t, and the times the states of the traffic light are stayed in are between 10 seconds and 200 seconds, we make use of the objective function (5.6) to determine the optimal control parameters from the two starting points listed in Table 7.6. In this table, the variables t_1 and t_2 refer to the time periods traffic from road l_1 and l_2 has green light at a time, respectively. The variable s_{start} specifies what road starts out with having green light. If $s_{\text{start}} = 1$, it means that the road l_1 starts out by having green light, while $s_{\text{start}} = 0$ means that the road l_2 starts out by having green light.

Figure 7.17 shows the reduction in the average delay time as a function of the number of iterations taken from the two starting points. From this figure,

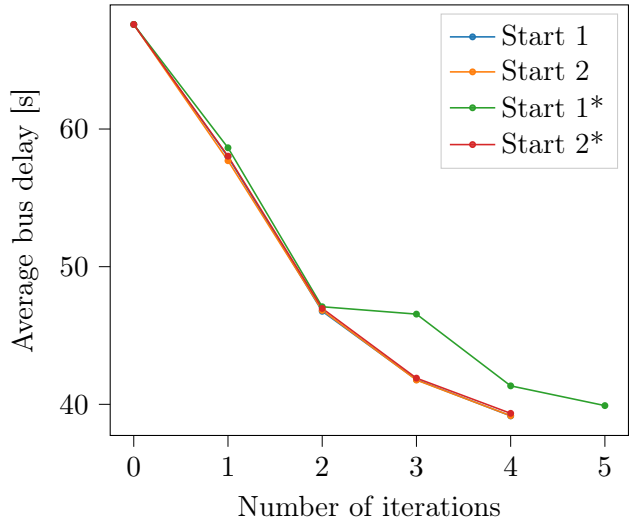


Figure 7.17: The objective function as a function of the number of iterations.

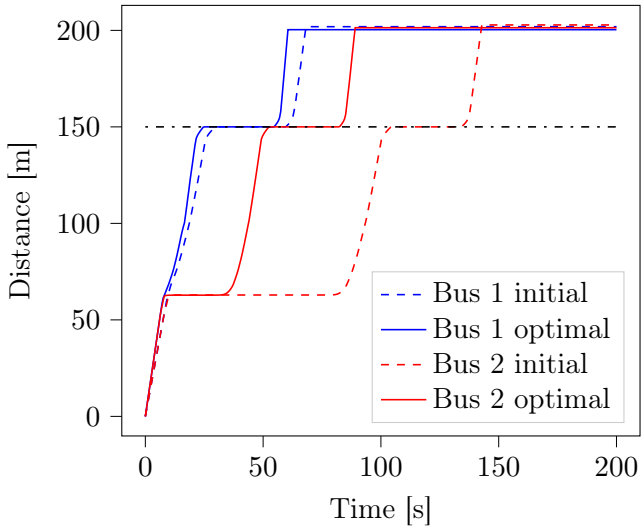


Figure 7.18: The distance travelled by the two busses of the network. The lengths travelled with initial choice of control parameters is marked by dotted lines, and with optimal parameters by full lines. The distance to the bus stop is marked by a black dotted line.

Table 7.6: Starting points for the optimization of the control parameters of a two-to-two junction.

Start Id	l_1	l_2	r_1	r_2	t_1	t_2	s_{start}
1	40	40	40	40	60	60	1
2	40	40	40	40	60	60	0

we see that the optimal value is found after four iterations. In Figure 7.18 we see the distances travelled by the two busses both for the initial choice of control parameters given as starting point 1 in Table 7.6 and for the optimal choice of control parameters. From this figure, we see that bus one travels marginally faster for the optimal choice of control parameters, while bus two arrives at the stop much faster. This can be explained by the fact that for both choices of control parameters, the traffic is green for bus one, allowing this bus to cross the junction immediately, while the light is red for bus two at the beginning of the simulation. Changing the light earlier from green to red makes it so that the light is still green for the first bus upon arrival, while the second bus has to wait less, resulting in a reduced delay time.

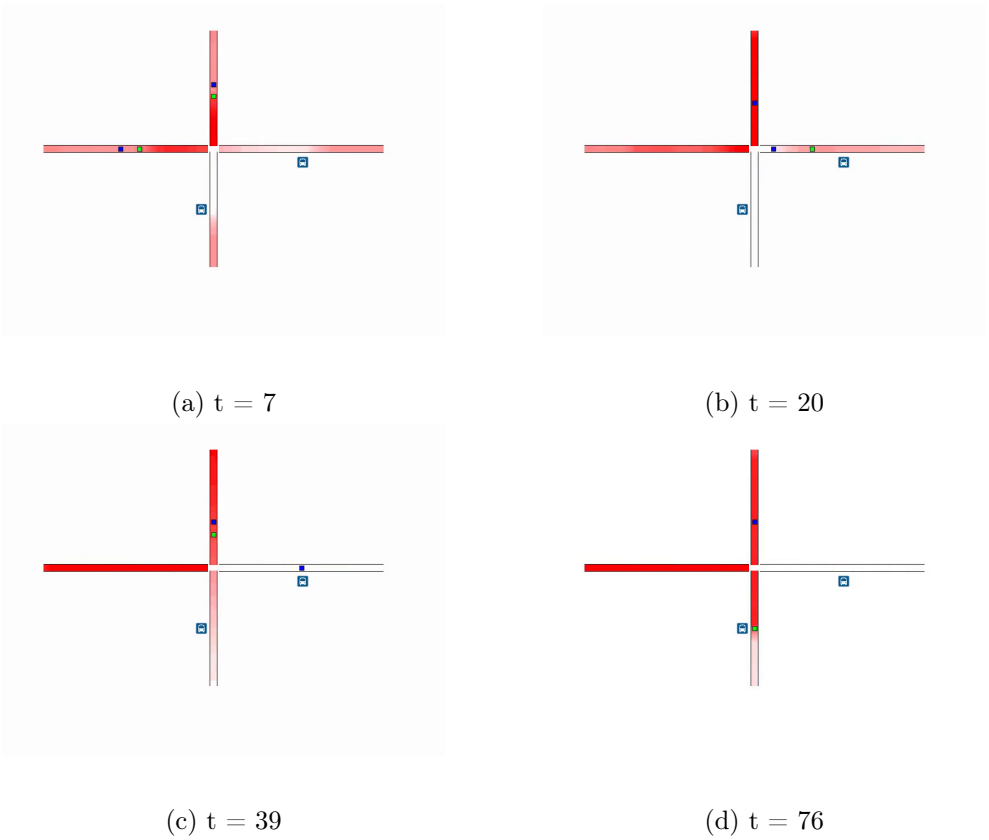
As explained briefly at the end of Section 3.2.2, when computing the gradient in the backward mode, we might experience a prohibitively large use of memory. Further, since the computational effort in the forward mode scales with the number of input variables, it might be inefficient in some cases. Although we do not have any of the aforementioned issues for this small example, we still want to test the gradient estimation that restarts computation at each when a bus stops (see Section 6.2.1) to check how well the objective values match when we are using only an estimated gradient. Table 7.7 lists also the convergence history for the two cases when using an estimated gradient. These results are marked by an asterisk ($*$). We see that both the optimal solutions, as well as the convergence rate are similar although not identical to when using an exact gradient. A video showing the traffic flow for the optimal choice of control parameters compared to the traffic flow for the starting point 1 of Table 7.6, can be found at `torjeihelset.github.io/master_project`. We include some snapshots from this video in Figure 7.19.

Properties of the objective function

Also for this network, we investigate the properties of the objective functions. The results can be seen in Section C.2. As for the single lane network and the single junction network, we investigate the behaviour of the three objective functions (5.6), (5.3) and (5.5), by varying one control parameter at a time with varying increments and calculating the corresponding objective values. As for the single lane network and the single junction network, we are able to capture more irregularity in the objective function when decreasing the

Table 7.7: Convergence history of the optimization method from the two starting points listed in Table 7.6 using objective function (5.6). When the starting id is marked by an asterisk (∗), the gradient was computed by restarting the computational graph every time a new busstop was reached.

Start	**n**	l_1	l_2	r_1	r_2	t_1	t_2	$J(p)$
1	0	40.0	40.0	40.0	40.0	60.0	60.0	67.596
1	1	45.1	39.8	50.0	50.0	48.0	60.0	57.831
1	2	48.605	50	50.0	50.0	28.0	60.0	46.637
1	3	49.637	50	50.0	50.0	18.0	60.0	41.731
1	**4**	**49.8**	**49.8**	**50.0**	**50.0**	**13.0**	**60.0**	**39.063**
2	0	40.0	40.0	40.0	40.0	60.0	60	67.596
2	1	44.2	42.2	50.0	50.0	60.0	48.1	57.698
2	2	50.0	50.0	50.0	50.0	60.0	28.1	46.825
2	3	50.0	50.0	50.0	50.0	60.0	18.1	41.763
2	**3**	**50.0**	**50.0**	**50.0**	**50.0**	**60.0**	**13.1**	**39.170**
1∗	0	40.0	40.0	40.0	40.0	60.0	60.0	67.596
1∗	1	46.3	34.4	50.0	50.0	47.4	60.0	58.639
1∗	2	50.0	43.1	50.0	50.0	27.7	60.0	47.089
1∗	3	50.0	40.6	50.0	50.0	25.7	60.0	46.554
1∗	4	50.0	41.1	50.0	50.0	15.7	60.0	41.341
1∗	**5**	**50.0**	**41.1**	**50.0**	**50.0**	**13.2**	**60.0**	**39.907**
2∗	0	40.0	40.0	40.0	40.0	60.0	60.0	67.596
2∗	1	38.4	44.2	50.0	50.0	60.0	47.5	58.028
2∗	2	44.5	50.0	50.0	50.0	60.0	27.5	46.967
2∗	3	44.6	50.0	50.0	50.0	60.0	17.5	41.903
2∗	**3**	**44.6**	**50.0**	**50.0**	**50.0**	**60.0**	**12.5**	**39.345**



(a) t = 7 (b) t = 20

(c) t = 39 (d) t = 76

Figure 7.19: Snapshots of the simulation for the optimal control parameters of the two-to-two network. The amount of traffic is visualized with a red color scale. White means that there is no traffic, and a darker red color means that the road is congested. The green square shows the position of the bus with optimal control parameters. The blue square shows the position of the bus with non-optimal control parameters.

increments between evaluation points. However, similar to the discussion in Section 7.2, we believe that this irregular behaviour of the zoomed in view of the objective function is mainly a side effect of insufficient numerical accurary to distinguish small variations in the speed limits.

When considering Figure C.23 it looks like there are local minima (potentially global minima for road l_2) near 40 km/h. Interestingly, these minima are all located at the center of the values we are varying. At these points, all speed limits are equal to 40 km/h. Thus, one possible explanation for these minima is that the best way of reducing the total travel times is to choose relatively equal speed limits for the different roads.

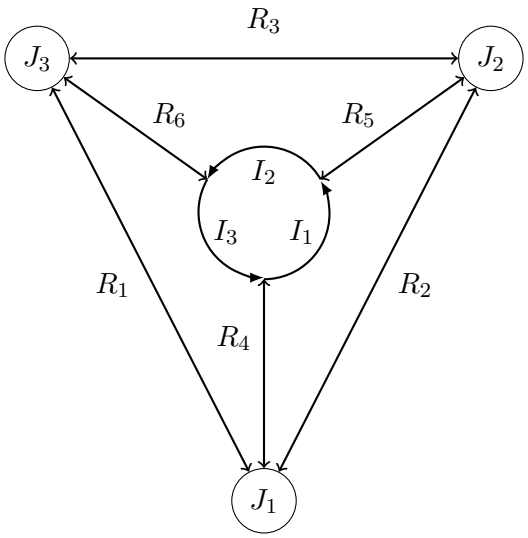


Figure 7.20: A sketch of the network used to check the convergence of the model with respect to decreasing spatial lengths for the cells. All roads except the ones in the roundabout are bidirectional. The network forms a closed loop, meaning that no roads exit the network. We denote the bidirectional roads by R_i and the mainlines of the roundabout by I_i.

7.4 Checking Convergence of the Model

In section 3.6 of [3] it was discussed how different finite-volume schemes give convergent solution to the transport equation on a single unidirectional road. However, we have not proved that this is also the case for a larger road network. We will not prove that this is the case in this thesis but will check whether or not this seems to be the case numerically. To this end, we compare the solution for different choices of spatial grid sizes. A sketch of the network we will use for the convergence test can be seen in Figure 7.20.

We start by checking the convergence order of the network. To this end, we will simulate traffic moving along the network for a fixed amount of time and compare the solutions obtained by using different values for the spatial grid length Δx.

Before discussing how results from different simulations are compared, we start by giving a detailed explanation of the network we are using for the convergence test. As we see in Figure 7.20, the network consists of an outer triangle of roads where each vertex leads to a roundabout that lies in the circle of the triangle. The mainlines of the roundabout are denoted by I_i, and are all assumed to have a length of 25 metres. All roads that are not part of the roundabout are bidirectional roads denoted by R_i. Bidirectional roads are modelled as two unidirectional roads, and hence when modelling this network, we will introduce two unidirectional roads l_i and r_i for each bidirectional road

Table 7.8: Estimated order of convergence for the road network visualized in Figure 7.20.

Δx [m]	Error	Estimated OOC
0.16667	0.61080	—
0.08333	0.30660	0.9940
0.04167	0.14204	1.1103
0.02083	0.06252	1.1835

R_i. We assume that the three roads R_4 and R_5 and R_6 (modelled by l_i, r_1, $i = 4, 5, 6$) leading into the roundabout all have lengths equal to 50 metres. Finally, we assume that the three roads of the outer triangle (modelled by l_i, r_1, $i = 1, 2, 3$) all have lengths of 125 metres. The three junctions J_1, J_2 and J_3 are all governed by coupled traffic lights and some rules that specify how traffic is distributed. Details of the roads of the networks are listen in Table B.1, details of the junctions of the network are listed in Table B.2, and details of the traffic lights are listen in Table B.3. When, for instance, traffic light T_1^c is in state A, it means that the incoming roads l_1 and r_2 have green lights. The first time cycle of the traffic lights is the amount of time the first state is stayed in. The second time is the amount of time the second state is stayed in, and so on.

We now move on to discuss how results from different simulations can be compared. After one simulation, we have for each road a set of values that approximate the densities at each cell at a finite set of times $\{t^n\}_n$; that is, for road l_i, for instance, we have the set $\{\rho_{l_i,j}^n\}_{j,n}$, where $\rho_{l_i,j}^n$ is an approximation to the integral

$$\rho_{l_i,j}^n \approx \int_{x_{j-1/2}}^{x_{j+1/2}} \rho_{l_i}(x, t^n)dx,$$

assuming that ρ_{l_i} is the exact density on road l_i.

A natural norm to measure the convergence in is the L^1-norm. The details on how we do this are included in Section A.3. We perform the simulation for a period of 100 seconds with varying spatial lengths. As we do not have an exact solution we can compare to, we first perform the simulation using a very small step length of 0.13 metres, to which we compare the results from different spatial lengths: 4.1675, 2.0825, 1.0425, 0.52 meters. The estimated convergence order for the different spatial lengths can be seen in Figure 7.21. We also list the computed differences and convergence orders in Table 7.8. From the table and the figure, we see that the convergence rate appears to remain more or less constant around one. In section 3.6 in [3], it was shown that the high-resolution scheme we are using had a convergence rate of approximately two as long as the solution is smooth. In the presence of shocks, however, the convergence rate reduces to first order. As we in this example expect to have many shocks due to traffic lights and junctions, a convergence rate of approximately one seems reasonable.

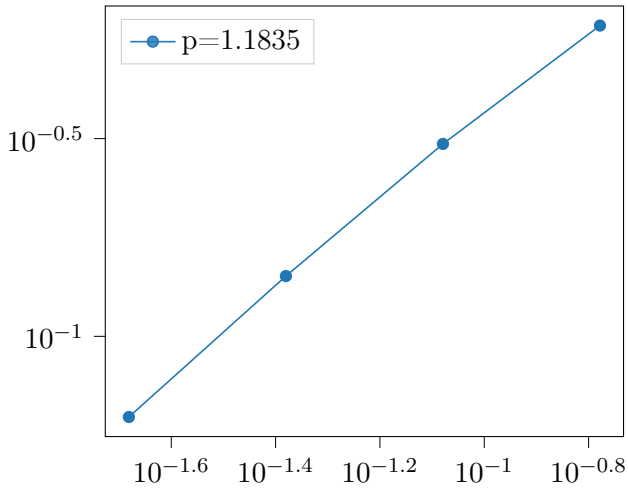


Figure 7.21: Numerical convergence order. The estimated rate of convergence at the final step is given by p.

7.5 Optimal Control Problem for Larger Network

Turning our focus back to the goal of finding the control parameters minimizing an objective function, we consider a more complicated network than those considered so far. A sketch of the network in question can be seen in Figure 7.22. This network consists of six bidirectional roads and one unidirectional road that are connected in three junctions. In addition, the network contains one roundabout with three mainline roads. As in Section 7.4 we model each of the bidirectional roads R_i by two unidirectional roads, l_i and r_i. Since R_5 is a unidirectional road, this road will be modelled only by l_5.

In total, including the three mainlines of the roundabout, the network model consists of 16 unidirectional roads. We assume that all roads have a length of 50 metres. As in Figure 7.22 we denote the three junctions by J_1, J_2 and J_3. The traffic through junction J_1 is controlled by a coupled traffic light, as well as yielding rules. The other two junctions J_2 and J_3 are controlled only by yielding rules. The description of each road in the system is listed in Table B.4. The description of the rules governing each of the junctions is listed in Table B.5. Finally, the incoming and outgoing roads affected by the traffic light in junction J_1 are listed in Table B.6. The influx to the traffic-flow network is listed in Table 7.9.

We assume that the same speed limits are used for l_i and r_i, while allowing for different speed limits on the roads l_i and l_j when $i \neq j$. We will also assume that the same speed limit must be used for the entire roundabout. Finally, we assume that the traffic light governing junction J_1 is described by a cycle consisting of only two times, one for each state. In total, this gives us ten control parameters that we may choose; eight for the different speed limits,

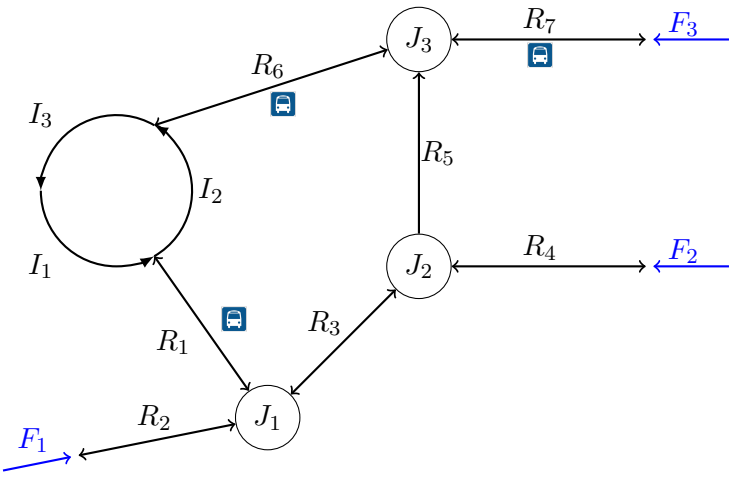


Figure 7.22: Sketch of a larger network showing all of the components of the model. We denote the roads of the network by R_i and the mainlines of the roundabout by I_i. All roads R_i except for R_5 are bidirectional. The junction connecting R_1, R_2 and R_3 is controlled by a traffic light.

Table 7.9: Influx of traffic to the larger traffic-flow network. The influx is calculated as $f_{\text{in}} = \gamma\rho(1-\rho)$, with $\gamma = \frac{v_{\text{in}}}{L}$.

Variable	ρ	v_{in} [km/t]	L [m]
F_1	0.2	40	50
F_2	0.2	40	50
F_3	0.2	40	50

and two for the controlling of the traffic light.

We will use this network for a number of different experiments. First, we try to minimize the objective function (5.6). We will simulate three busses travelling through the network. The specifics of these busses are listed in Table 7.10. This table includes three different versions of bus A, each with different starting times. We make use of the modified steepest descent method from Section 5.2.1 to compute the optimal control parameters, starting from the six different starting points listed in Table 7.11. Since the two other bus lines B and C only have one version, we do not include these in Table 7.11, as the same buses are used for all starting points. We note that since we are using the same speed limits for road l_i and road r_i, we only list the speed limit on road R_i.

Assuming that the traffic light in junction J_1 is described by a cycle consisting of only two times, we have in total ten different control parameters that we try to optimize. The variable s_{start} takes two values, zero or one. When $s_{\text{start}} = 1$, the traffic light T_1^c of junction J_1 starts out in state A given in

Table 7.10: Bus routes of the network from Figure 7.22.

Id	**Roads**	**Stops**	**Times**	t_{start}
A_1	l_2, l_3, l_5, l_7	$(l_7, 25)$	0	0
A_2	l_2, l_3, l_5, l_7	$(l_7, 25)$	0	15
A_3	l_2, l_3, l_5, l_7	$(l_7, 25)$	0	20
B	r_4, r_3, r_1, I_1, I_2	$(r_1, 25)$	0	0
C	I_3, I_1, l_6, l_7	$(l_6, 25)$	0	0

Table 7.11: Starting points for the network shown in Figure 7.22.

Id	R_1	R_2	R_3	R_4	R_5	R_6	R_7	I	t_1	t_2	s_{start}	**A-Bus**
1	40	40	40	40	40	40	40	40	60	60	1	A_1
2	50	50	50	50	50	50	50	50	30	30	1	A_1
3	40	40	40	40	40	40	40	40	60	60	0	A_1
4	50	50	50	50	50	50	50	50	30	30	0	A_1
5	40	40	40	40	40	40	40	40	60	60	1	A_2
6	40	40	40	40	40	40	40	40	60	60	1	A_3

Table B.6. This means that the roads l_1 and r_3 will start out by having green lights, while the road l_2 starts out by having red light. If, on the other hand, $s_{\text{start}} = 0$, roads l_1 and r_3 will start out by having red lights, while road l_2 starts out by having green light.

The reduction of the average delay time can be seen in Figure 7.23 for the three first starting points. We observe in particular that when starting from starting point 3, we reach a different local minimum with a much higher average bus delay. We also visualize the length travelled by the three busses A_1, B and C with control parameters as in starting point 1, and with the optimal choice of control parameters in Figure 7.24. We notice that there is a marginal improvement in the distance travelled by Bus C, while the other two busses experience a more significant change. This can be explained by the fact that only Bus A_1 and B travel through the junction controlled by a traffic light. The travel times of these busses depend to a large degree on how the traffic light is controlled, and finding good settings for the traffic light results in a significant reduction of the average delay.

Table 7.12 lists the full convergence history starting from the six different starting points of Table 7.11. Since the starting state of the traffic light is different for cases 3 and 4, these results cannot be directly compared to the others. Further, since one of the busses starts at later times in cases 5 and 6, also these cannot be directly compared to the rest. We see that for cases 1 and

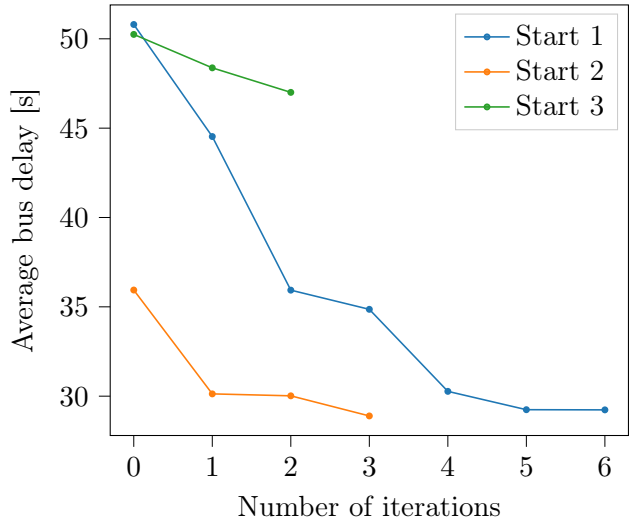


Figure 7.23: Objective value as a function of the number of iterations of the optimization algorithm.

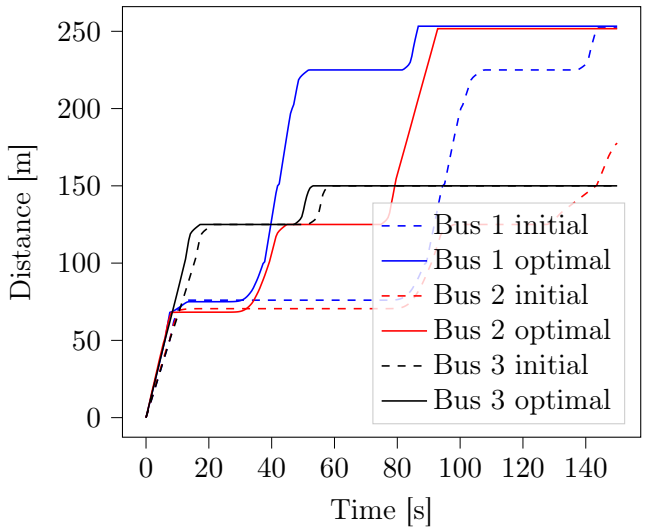


Figure 7.24: The distance travelled by the three busses with an optimal choice of control parameters and with the initial choice of control parameters.

Table 7.12: Convergence history for the control parameters with the starting points listed in Table 7.11 using objective function (5.6).

R_1	R_2	R_3	R_4	R_5	R_6	R_7	I	t_1	t_2	$J(p)$
1:										
40.0	40.0	40.0	40.0	40.0	40.0	40.0	40.0	60.0	60.0	50.80
46.5	48.3	49.4	50.0	34.7	50.0	50.0	33.6	60.0	47.4	44.53
50.0	50.0	50.0	50.0	42.5	50.0	50.0	43.1	60.0	27.4	35.93
50.0	50.0	50.0	45.2	50.0	50.0	50.0	50.0	60.0	27.2	34.86
50.0	50.0	50.0	50.0	45.8	50.0	50.0	43.1	60.0	10.0	30.27
50.0	50.0	50.0	48.0	50.0	50.0	47.9	50.0	60.0	10.0	29.24
50.0	**50.0**	**50.0**	**47.0**	**44.4**	**50.0**	**50.0**	**49.1**	**60.0**	**10.0**	**29.23**
2:										
50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	30.0	30.0	35.93
50.0	50.0	50.0	50.0	47.3	50.0	50.0	42.4	30.0	10.0	30.13
50.0	50.0	41.3	47.6	50.0	50.0	50.0	50.0	30.0	10.0	30.02
50.0	**50.0**	**50.0**	**41.9**	**47.5**	**50.0**	**50.0**	**50.0**	**30.0**	**10.0**	**28.89**
3:										
40.0	40.0	40.0	40.0	40.0	40.0	40.0	40.0	60.0	60.0	50.25
44.7	50.0	43.5	40.0	40.1	42.2	41.8	44.1	60.2	60.0	48.4
50.0	**50.0**	**44.2**	**39.5**	**50.0**	**50.0**	**42.3**	**50.0**	**60.9**	**60.0**	**47.00**
4:										
50.0	**50.0**	**50.0**	**50.0**	**50.0**	**50.0**	**50.0**	**50.0**	**30.0**	**30.0**	**35.51**
5:										
40.0	40.0	40.0	40.0	40.0	40.0	40.0	40.0	60.0	60.0	53.36
49.4	50.0	50.0	50.0	38.4	48.8	40.9	39.3	60.0	44.2	45.5
50.0	50.0	50.0	50.0	45.5	50.0	41.1	48.0	60.0	24.2	37.5
50.0	50.0	50.0	50.0	45.5	50.0	41.1	48.0	60.0	24.2	34.38
50.0	**50.0**	**50.0**	**50.0**	**46.7**	**50.0**	**41.1**	**49.4**	**60.0**	**14.2**	**34.38**
6:										
40.0	40.0	40.0	40.0	40.0	40.0	40.0	40.0	60.0	60.0	54,74
49.5	50.0	50.0	50.0	39.9	44.7	40.3	40.6	60.0	44.6	47.4
50.0	50.0	50.0	50.0	47.6	50.0	50.0	49.8	60.0	24.6	38.78
50.0	**50.0**	**50.0**	**50.0**	**47.7**	**50.0**	**50.0**	**50.0**	**60.0**	**19.6**	**37.26**

2, although slightly different control parameters were found, a similar objective value was reached. Interestingly, we note that in both cases, the speed limits on road R_4 and R_5 were chosen lower than its maximal allowed value. In cases 5 and 6, buses A_2 and A_3 start at a later time than bus A_1. In both cases, this results in different optimal settings for the traffic light. This can be explained by the fact that the optimal control of the traffic light should still allow the A bus to cross, before changing state as soon as possible after.

As for the network in Section 7.3, we want to investigate how easy it is to

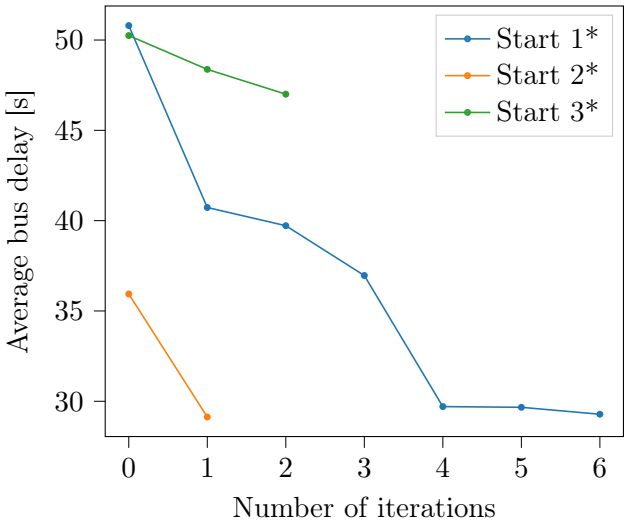
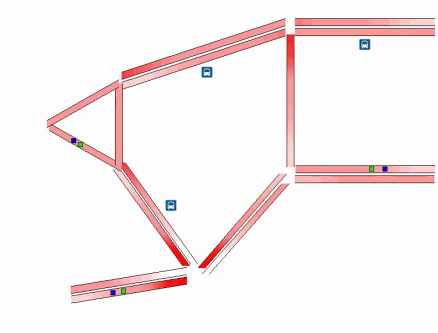


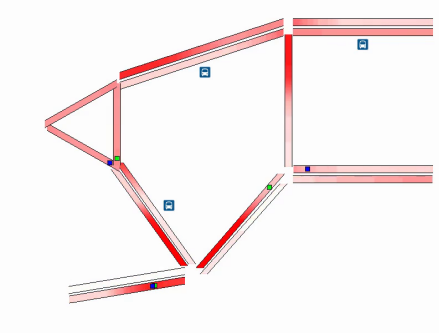
Figure 7.25: Objective value as a function of the number of iterations of the optimization algorithm with the computational graph restarted after every stop was reached.

find the optimal solution when using an approximated gradient instead of an exact one. We therefore estimate the gradient by simulating until a bus stop is reached and compute the local gradient before resetting the computational graph and restarting the network. The results for the four first starting points are listed in Table 7.13. We observe that also for this more complex network, the optimization results when using an approximated gradient are comparable both in convergence rate and quality of the optimal solution reached to the results from using an exact gradient.

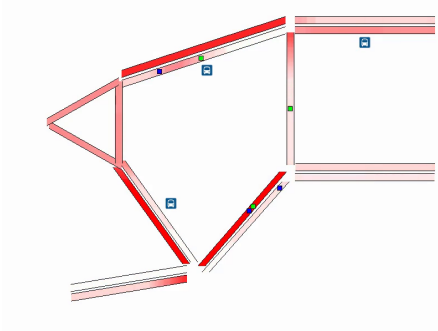
Also for this network we have created a video comparing the flow of traffic for different choice of control parameters. We compare the flow of traffic with control parameters given as starting point 1 in Table 7.11 to flow observed with the control parameters obtained at the end of the optimization algorithm starting from these parameters. A video showing this comparison can be found at `torjeihelset.github.io/master_project`. We include some snapshots from this video in Figure 7.26.



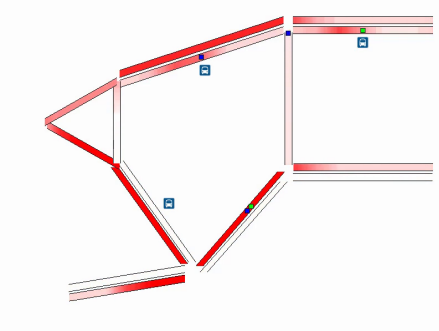
(a) t = 3



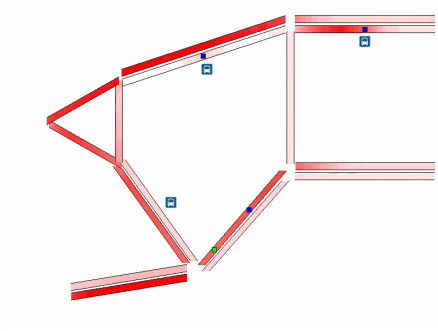
(b) t = 7



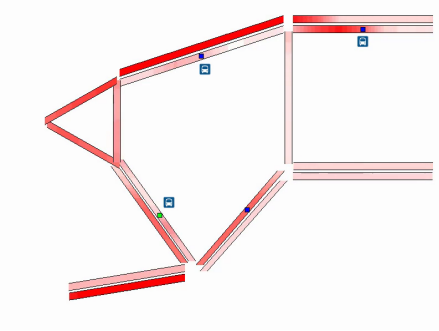
(c) t = 17



(d) t = 25



(e) t = 38



(f) t = 43

Figure 7.26: Snapshots of the simulation for the optimal speed limit of the larger network. The amount of traffic is visualized with a red color scale. White means that there is no traffic, and a darker red color means that the road is congested. The green squares shows the positions of the buses with optimal control parameters. The blue squares shows the position of the bus with non optimal control parameters.

Table 7.13: Convergence history of the optimization method from the starting points listed in Table 7.11 using objective function (5.6) with gradient computation restarting after every stop is reached. For consistency, we mark the starting points with and asterisk (∗) to show that the restarting gradient computation was used.

R_1	R_2	R_3	R_4	R_5	R_6	R_7	I	t_1	t_2	$J(p)$
1∗:										
40.0	40.0	40.0	40.0	40.0	40.0	40.0	40.0	60.0	60.0	50.80
48.5	50.0	48.7	39.8	38.3	46.3	41.4	45.2	60.0	40.0	40.73
50.0	50.0	48.0	39.8	50.0	50.0	41.8	50.0	60.0	39.6	39.72
50.0	50.0	42.4	39.8	46.7	50.0	42.5	50.0	60.0	29.6	36.96
50.0	50.0	48.5	39.8	50.0	50.0	42.7	50.0	60.0	10.0	29.70
50.0	50.0	48.1	39.8	50.0	50.0	50.0	50.0	60.0	10.0	29.67
50.0	**50.0**	**50.0**	**39.8**	**50.0**	**50.0**	**50.0**	**49.1**	**60.0**	**10.0**	**29.28**
2∗:										
50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	30.0	30.0	35.93
50.0	**50.0**	**50.0**	**49.6**	**50.0**	**50.0**	**50.0**	**50.0**	**30.0**	**10.0**	**29.13**
3∗:										
40.0	40.0	40.0	40.0	40.0	40.0	40.0	40.0	60.0	60.0	50.3
43.2	50.0	45.0	40.0	42.2	43.1	42.5	45.7	60.4	60.0	47.8
46.1	**50.0**	**44.9**	**40.0**	**50.0**	**50.0**	**43.1**	**50.0**	**61.2**	**60.0**	**47.06**
4∗:										
50.0	**50.0**	**50.0**	**50.0**	**50.0**	**50.0**	**50.0**	**50.0**	**30.0**	**30.0**	**35.5**

7.5.1 Different Objective Functions

As discussed in Section 5.1, different objective functions may be used to determine the best control parameters. We now investigate different choices and compare the results to those obtained when only considering the delays of busses.

Total Travel Time

One way of measuring the quality of the traffic flow in a network is to measure the total travel time of the drivers in this system. We aim to minimize the total travel time of the drivers in the system, and therefore make use of the objective function (5.5). We only consider the two first starting points from Table 7.11, and list the full convergence history in Table 7.14, and visualize the reduction of the objective function in Figure 7.27. We see that the times t_1 and t_2 are chosen far apart from each other, meaning that one state of the traffic light is favored over the other. Interestingly, we see that the best choice of control parameters found entails setting the speed limits on roads R_1 and R_5 significantly below their maximum values.

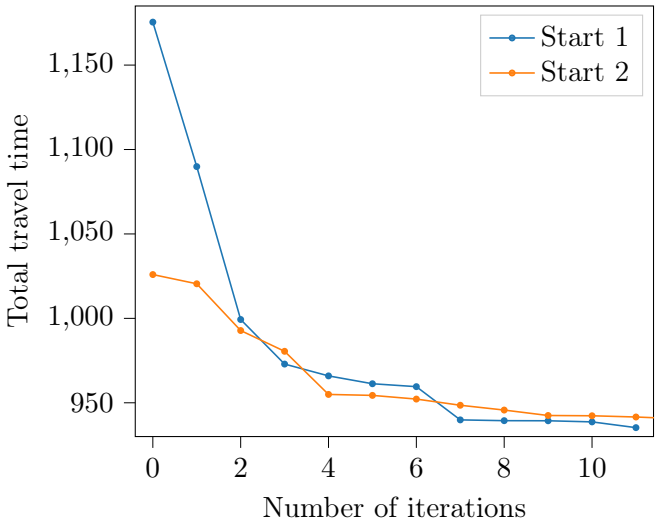


Figure 7.27: Objective value as a function of the number of iterations of the optimization algorithm.

From Table 7.14, we remark that after only a few iterations, the variable t_2 reaches a value between 10 and 15 seconds for both starting points. After this point, the updating of the other variables seem to slow down, and the reduction in the objective value per iteration also decreases. A possible explanation for this behaviour is that the optimal control problem for this particular network and this particular objective function might be *poorly scaled* [40]. A problem is poorly scaled when a change in one direction results in a much larger variation in the objective function than a change in a different direction. In our case, it might be that a small change in t_2 results in a much larger change in the objective function than a change in the other variables. According to [40], the steepest descent method is sensitive to poorly scaled problems, and so other methods such as Newton's method might give better results, as this method is unaffected by poor scaling.

Maximum Throughput

One way of measuring the quality of the traffic flow in a network is to measure the total throughput. Hence, we investigate what control parameters give rise to the maximal throughput. We only consider the two first starting points from Table 7.11. The full convergence history when using objective function (5.3) is listed in Table 7.15, and a graph can be seen in Figure 7.28. The optimization results indicate that setting as high speed limit as possible is favorable. In addition, we see to an even higher extent than when minimizing the travel time that one of the states of the traffic light is favoured, as the two times of the cycle t_1 and t_2 are chosen very differently. Finally, a higher number

Table 7.14: Convergence history of the optimization method from the starting points listed in Table 7.11 using objective function (5.5).

R_1	R_2	R_3	R_4	R_5	R_6	R_7	I	t_1	t_2	$J(p)$
1:										
40.0	40.0	40.0	40.0	40.0	40.0	40.0	40.0	60.0	60.0	1175.44
38.4	41.9	36.8	50.0	50.0	50.0	42.0	42.3	60.3	44.1	1089.88
48.4	49.1	40.9	50.0	46.3	50.0	45.1	50.0	65.3	28.1	999.32
43.2	50.0	50.0	50.0	37.8	50.0	48.7	50.0	76.6	14.0	972.93
50.0	50.0	50.0	50.0	34.8	50.0	49.8	50.0	96.6	17.2	965.93
47.3	50.0	50.0	50.0	34.5	50.0	49.7	50.0	101.1	10.0	961.28
47.3	50.0	50.0	50.0	34.6	50.0	49.7	50.0	102.0	15.0	959.55
42.6	50.0	50.0	50.0	44.6	50.0	48.5	50.0	109.0	10.0	939.87
42.7	50.0	50.0	50.0	44.0	50.0	48.6	50.0	109.2	15.0	939.41
42.8	50.0	50.0	50.0	43.6	50.0	48.6	50.0	109.1	10.0	939.35
42.9	50.0	50.0	50.0	43.5	50.0	48.7	50.0	109.3	15.0	938.69
42.9	**50.0**	**50.0**	**50.0**	**43.5**	**50.0**	**48.7**	**50.0**	**109.3**	**12.5**	935.28
2:										
50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	30.0	30.0	1025.94
39.9	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	32.6	1020.48
50.0	45.5	50.0	50.0	50.0	50.0	50.0	50.0	61.3	14.3	992.79
40.7	50.0	50.0	50.0	44.9	50.0	50.0	50.0	68.4	23.6	980.49
43.3	49.3	50.0	50.0	43.1	50.0	50.0	50.0	75.3	13.6	954.95
43.3	49.3	50.0	50.0	43.1	50.0	50.0	50.0	76.1	14.8	954.38
43.4	49.9	50.0	50.0	42.8	50.0	50.0	50.0	81.1	16.6	952.20
43.5	50.0	50.0	50.0	42.8	50.0	50.0	50.0	83.6	15.9	948.54
43.5	50.0	50.0	50.0	42.7	50.0	50.0	50.0	86.1	14.9	945.69
43.5	50.0	50.0	50.0	42.6	50.0	50.0	50.0	88.6	13.5	942.47
43.6	50.0	50.0	50.0	42.6	50.0	50.0	50.0	91.1	14.7	942.31
43.6	50.0	50.0	50.0	42.6	50.0	50.0	50.0	92.3	14.8	941.58
43.6	**50.0**	**50.0**	**50.0**	**42.6**	**50.0**	**50.0**	**50.0**	**93.6**	**14.8**	940.53

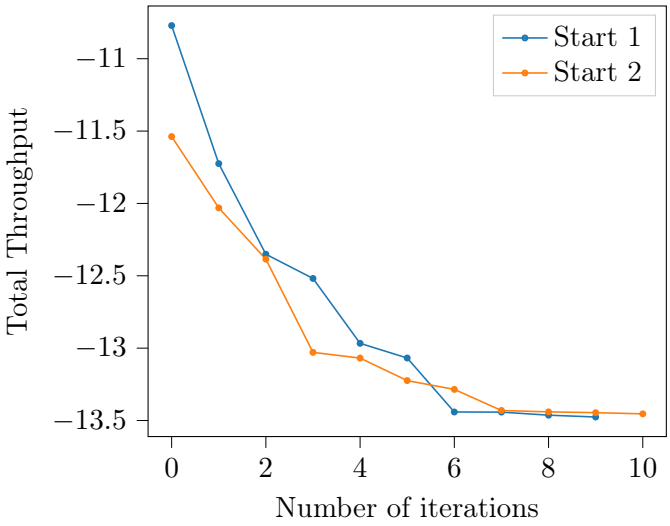


Figure 7.28: Objective value as a function of the number of iterations of the optimization algorithm.

of iterations were needed to find stationary points than what we saw when reducing the delays of the buses. Also for this objective function, we see that after the variable t_2 reaches a certain value, the convergence rate is reduced. This might again be due to poor scaling of the optimal control problem in this case.

Table 7.15: Convergence history of the optimization method from the starting points listed in Table 7.11 using objective function (5.3).

R_1	R_2	R_3	R_4	R_5	R_6	R_7	I	t_1	t_2	$J(p)$
1:										
40.0	40.0	40.0	40.0	40.0	40.0	40.0	40.0	60.0	60.0	-10.77
43.7	37.2	40.5	50.0	50.0	42.6	41.8	49.5	70.0	58.1	-11.72
48.9	41.6	40.6	50.0	50.0	43.8	42.4	50.0	70.1	38.2	-12.35
50.0	43.6	40.5	50.0	50.0	44.4	43.5	50.0	90.1	49.6	-12.52
47.4	46.5	40.6	50.0	50.0	43.7	43.9	50.0	90.1	29.5	-12.97
50.0	48.6	41.5	50.0	50.0	49.1	45.6	50.0	110.1	36.5	-13.07
50.0	46.9	50.0	50.0	50.0	50.0	50.0	50.0	110.1	16.5	-13.44
50.0	49.9	49.9	50.0	50.0	50.0	50.0	50.0	111.3	14.0	-13.44
50.0	50.0	47.4	50.0	50.0	50.0	50.0	50.0	121.3	16.2	-13.46
50.0	**49.9**	**47.4**	**50.0**	**50.0**	**50.0**	**50.0**	**50.0**	**121.3**	**13.7**	-13.48
2:										
50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	30.0	30.0	-11.54
50.0	50.0	48.5	50.0	50.0	50.0	50.0	48.9	50.0	29.0	-12.0
50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	70.0	42.0	-12.3
50.0	42.6	47.0	50.0	50.0	50.0	50.0	50.0	90.0	38.8	-13.03
50.0	42.7	47.0	50.0	50.0	50.0	50.0	50.0	90.1	33.8	-13.07
50.0	47.7	48.1	50.0	50.0	50.0	50.0	50.0	110.1	29.6	-13.22
50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	110.1	10.0	-13.29
50.0	48.9	50.0	50.0	50.0	50.0	50.0	50.0	120.1	18.9	-13.34
50.0	50.0	47.1	50.0	50.0	50.0	50.0	50.0	120.1	10.0	-13.44
50.0	49.5	47.2	50.0	50.0	50.0	50.0	50.0	121.4	10.0	-13.45
50.0	**50.0**	**50.0**	**50.0**	**50.0**	**50.0**	**50.0**	**50.0**	**121.4**	**10.0**	-13.45

7.6 Traffic Model of Kvadraturen in Kristiansand

Finally, we turn our focus to the largest numerical experiment we will do. We consider a road network section from the city center of Kristiansand, called Kvadraturen. This particular section was chosen both because of its grid-like structure, making it easier to visualize, as well as the fact that it contains all the different components of the model discussed in Chapter 4 in a small geographical area. In addition, because it is a relatively complex road network, it shows that the model scales quite well also to larger networks.

A sketch of the part of Kvadraturen that we model can be seen in Figure 7.29. Following the sketch, we denote the roads that are not mainlines of any roundabouts by l_i and r_i, while the mainlines of the roundabouts are denoted by I_i. We note that some road segments are only covered by one unidirectional road. To keep the naming of the roads l_i and r_i consistent with each other, so that index i refers to only one road segments, we do not define $_i$ and

r_i for all indices; this is the case for, e.g., l_8 or r_9.

Connecting the roads are a number of junctions. These junctions are no marked by name in the sketch, but are instead marked by small circles placed in the intersection between the roads. We note that since the end of each road r_i only connects to at most one junction, this leads to a natural way of naming each of the junctions. We denote by J_i the junction where road r_i is one of the incoming roads. In the case where multiple roads r_i enter the same junction, the lowest index i is used to name the junction. Since some r_i are not defined for all indexes, and since some of the r_i roads connect to the same junction, this means that J_i is also not defined for all indices; there is, e.g., no junction J_7. We note that there is one junction connecting the roads l_{24}, l_{25}, l_{29} and r_{29} where the only incoming roads are l_{24} and l_{29}. We denote this junction by r_{24}.

The details of each of the roads is listed in Tables B.7 to B.9. In addition, the specifics of all of the junctions are listed in Table B.10 to Table B.14. Finally, the specifics of all simple traffic lights are listed in Table B.15 and the specifics of the coupled traffic lights are listed in Table B.16. In Table B.15, $s_{\text{start}} = 0$ means that the traffic light starts out as being red. In Table B.16, $s_{\text{start}} = A$ means that the roads listed under "A In" starts out by having green light. These details will be used for all numerical examples related to the Kvadraturen network.

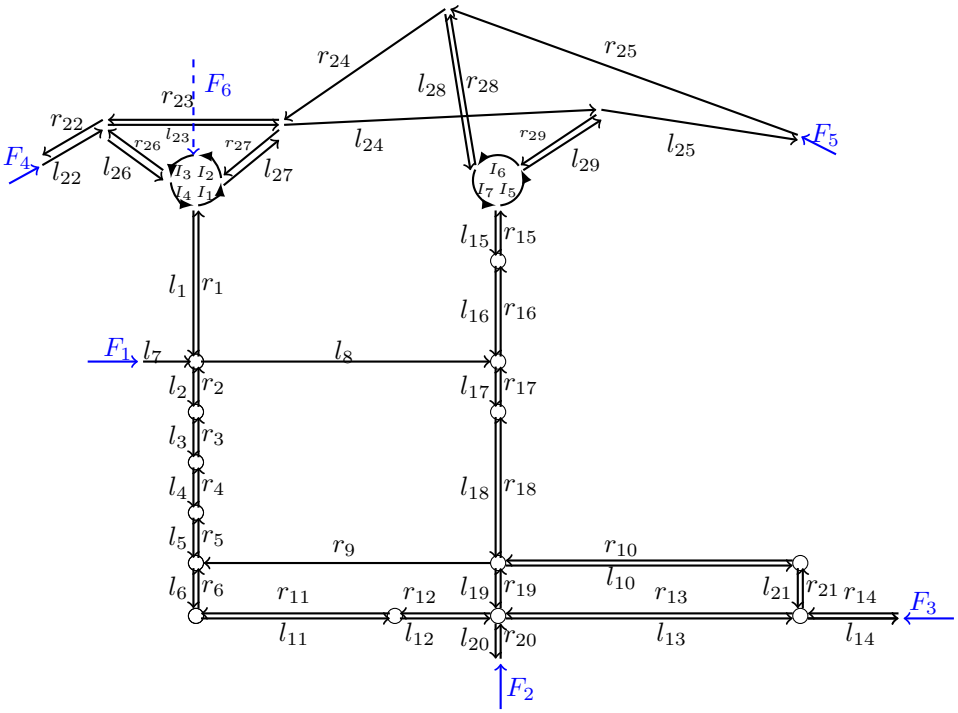


Figure 7.29: Sketch of the part of Kvadraturen that we try to model. Each road is denoted by l_i or r_i, and the mainlines of the roundabouts are denoted by I_i.

Since speed limits do not typically change after every junction, we will assume that this is the case also for the Kvadraturen network. We group the roads of the network by what speed limit is used for them. Denoting these speed limit variables by v_i, we list the different groupings of roads in Table 7.16. There are in total 14 different speed variables being used. In addition to these, there are 7 regular and 6 coupled traffic lights. Assuming that each traffic lights is described by a cycle consisting of only two times and that the speed limits remain constant in time, we have in total 40 control parameters we can try to optimize. For some of the optimization cases we will look at, we also allow the speed limits to change with time. We could, e.g., have three different speed variables for the group of roads l_1–l_6, r_1–r_6. Whenever this is the case, we will denote the speed variables by v_1^1, v_1^2 and v_1^3.

7.6.1 Optimization case A

We start by considering a case where we only simulate a total of sixty seconds. We perform the optimization procedure from two slightly different scenarios using a gradient computed using the forward mode and with a gradient computed with the backward mode with a resetting of the computational graph after each new bus stop is reached. In both cases, we consider the six busses $A_1, B_1, C_1, D_1, E_1, F_1$ listed in Table B.18. We assume that all of the speed limits are constant in time, and consider the two starting points listed in Table 7.17. Here, the variable C_i refers to the cycle determining the behaviour of traffic light i. The incoming flow of traffic of the two cases are listed

Table 7.16: Speed limit groupings for the Kvadraturen network. The same speed limit is used for all roads in the same group.

Speed Variable	**Roads**
v_1	l_1-l_6, r_1-r_6
v_2	l_7, l_8
v_3	l_{10}, r_9, r_{10}
v_4	l_{21}, r_{21}
v_5	l_{11}-l_{13}, r_{11}-r_{13}
v_6	l_{15}-l_{20}, r_{15}-r_{20}
v_7	l_{14}, r_{14}
v_8	l_{22}-l_{25}, r_{22}-r_{25}
v_9	l_{26}, r_{26}
v_{10}	l_{27}, r_{27}
v_{11}	l_{28}, r_{28}
v_{12}	l_{29}, r_{29}
v_{13}	I_1-I_4
v_{14}	I_5-I_7

Table 7.17: Starting points and stationary points found for the control parameters of Kvadraturen for optimization case A.

Variable	**Start 1**	**Start 2**	**Stationary 1**	**Stationary 2**
v_1	50	40	50.0	39.9
v_2	30	40	50.0	50.0
v_3	30	40	50.0	50.0
v_4	30	40	50.0	50.0
v_5	50	40	50.0	40.8
v_6	50	40	49.6	40.1
v_7	50	40	50.0	50.0
v_8	80	80	80.0	80.0
v_9	60	60	60.0	60.0
v_{10}	60	60	60.0	60.0
v_{11}	60	60	60.0	60.0
v_{12}	60	60	60.0	60.0
v_{13}	50	40	50.0	40.0
v_{14}	50	40	50.0	40.0
C_1	50, 100	50, 100	50, 100	50, 100
C_2	50, 100	50, 100	50, 100	50, 100
C_3	50, 100	50, 100	50, 100	50, 100
C_4	50, 100	50, 100	50, 100	50, 100
C_5	50, 100	50, 100	50, 100	50, 100
C_6	70, 70	70, 70	70, 70	70, 70
C_8	80, 80	80, 80	80, 80	80,0
C_{11}	50, 100	50, 100	50, 100	50, 100
C_{12}	60, 120	60, 120	60, 120	60, 120
C_{13}	60, 120	60, 120	60, 120	60, 120
C_{15}	60, 120	60, 120	60, 120	60, 120
C_{17}	80, 100	80, 100	80, 100	80, 100
C_{18}	80, 100	80, 100	80, 100	80, 100

in Table 7.18.

We list the full convergence history from the two starting points in Table 7.19. A graph showing this reduction in average delay time can be seen in Figure 7.30. We perform the optimization both using an exact gradient and with an approximated gradient where the evaluation trace is reset after every bus stop is reached. Results computed using an approximate gradient are marked by an asterisk ($*$). Finally, we list the best control parameters we found in Table 7.17. From this table we see that only the parameters related to the speed limits have been updated. This is likely due to the fact that none of the busses have reached a new bus stop after crossing a junction controlled by a traffic light, and so their delays do not depend on the settings of the traffic lights. We also

Table 7.18: Influx of traffic to Kvadraturen for optimization case A. The influx is calculated as $f_{\text{in}} = \gamma\rho(1-\rho)$, with $\gamma = \frac{v_{\text{in}}}{L}$.

Variable	ρ	v_{in} [km/t]	L [m]
F_1	0.2	30	50
F_2	0.3	50	50
F_3	0.6	50	50
F_4	1.0	80	50
F_5	1.0	80	50
F_6	0.35	50	50

Table 7.19: Convergence history of the optimization method from the starting points listed in Table 7.17 with influx listed in Table 7.18 using objective function (5.6)

Start	**Iteration**	$J(p)$
1	0	35.8793
1	1	29.7664
1	2	27.3376
1	3	26.8356
1	**4**	**26.1000**
2	0	33.0331
2	1	29.2120
2	**2**	**29.1851**
1∗	0	35.8793
1∗	1	27.9131
1∗	2	26.9191
1∗	**4**	**26.2770**
2∗	0	33.0331
2∗	**1**	**29.0702**

note that the two stationary points are slightly different while the resulting average delay is similar. This might be due to some speed limits being more important for the busses that have reached a bus stop within the time horizon of the simulation.

7.6.2 Optimization case B

We now consider a case where we simulate a total of 750 seconds. We vary the details of the network slightly and search for the optimal solution using a gradient computation in the forward mode and a gradient computation where the computational graph is reset after every new bus stop is reached. As we now are considering a longer simulation period, we consider all of the buses listed

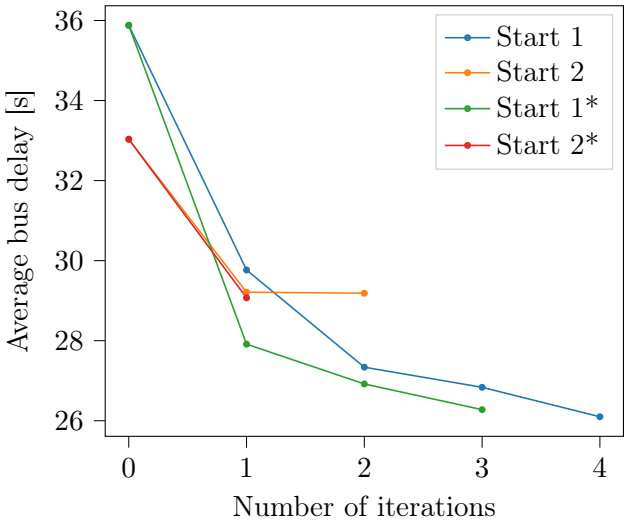


Figure 7.30: Average delay of the buses as a function of the number of iterations.

in Table B.18. We also consider some slightly different starting points, listed in Table 7.20 with influx of traffic listed in Table 7.21. The influxes F_4^A and F_5^A are used for starting point one, and the influxes F_4^B and F_5^A are used for starting point two. We note in particular that we now allow the speed limits on the different roads to vary in time. We assume that the switch between speed limit v_i^1 to speed limit v_i^2 happens at time $t = 600$ for all roads.

The convergence histories from the different starting points computed using an exact gradient and an approximated gradient are listed in Table 7.22. We also show the convergence history in Figure 7.31. We observe that for the first starting point, the optimization algorithm terminates after the first iteration. Closer inspection on what happened in this case showed that for the starting guess, bus B_2 arrived at its first stop just before the end of simulation at time $t = 750$. For the new choice of parameters, this bus did not arrive within the simulation horizon. For starting parameters, bus B_2 was less delayed than the average delay of the busses. However, although updating the control parameters helped lower the delays of the other buses, this was counterbalanced by B_2 not arrive in time and hence the overall average increased. This effect has already been discussed in Section 4.4, where we observed that the most robust way of handling it would be to run the simulation for long enough that all buses have time to exit the network. As this is not feasible for all optimization cases, it would be interesting to search for a different way of making the optimization problem more robust. We note that this effect was not an issue when using an approximate gradient starting from starting point 1, or when starting from starting point 2.

Also for this experiment we have created a video showing the comparison

Table 7.20: Starting points and stationary points found for the control parameters of Kvadraturen for optimization case B.

Variable	**Start 1**	**Start 2**	**Stationary 1**	**Stationary 2**
v_1^1, v_1^2	50, 50	40, 40	50.0, 50.0	50.0, 41.6
v_2^1, v_2^2	30, 30	40, 40	50.0, 30.0	48.6, 40.0
v_3^1, v_3^2	30, 30	40, 40	50.0, 36.5	50.0, 40.0
v_4^1, v_4^2	30, 30	40, 40	39.1, 30.9	47.3, 40.0
v_5^1, v_5^2	50, 50	40, 40	49.9, 49.9	38.4, 40.0
v_6^1, v_6^2	50, 50	40, 40	50.0, 50.0	50.0, 43.8
v_7^1, v_7^2	50, 50	40, 40	49.6, 49.8	40.4, 39.7
v_8^1, v_8^2	80, 80	80, 80	80.0, 80.0	80.0, 79.0
v_9^1, v_9^2	60, 60	50, 50	60.0, 60.0	51.5, 50.0
v_{10}^1, v_{10}^2	60, 60	50, 50	60.0, 60.0	50.0, 50.0
v_{11}^1, v_{11}^2	60, 60	50, 50	60.0, 60.0	51.2, 49.7
v_{12}^1, v_{12}^2	60, 60	50, 50	60.0, 60.0	50.0, 50.0
v_{13}^1, v_{13}^2	50, 50	40, 40	50.0, 50.0	37.5, 40
v_{14}^1, v_{14}^2	50, 50	40, 40	50.0, 50.0	20.0, 43.0
C_1	50, 100	50, 100	50, 100	50.5, 96.3
C_2	50, 100	50, 100	50, 100	47.5, 100.0
C_3	50, 100	50, 100	50, 100	50.0, 100.0
C_4	50, 100	50, 100	50, 100	48.8, 100.3
C_5	50, 100	50, 100	50, 100	48.3, 101.1
C_6	70, 70	70, 70	70, 70	72.3, 70.0
C_8	80, 80	80, 80	67.2, 80.0	63.3, 62.7
C_{11}	50, 100	50, 100	34.6, 100.0	50.6, 100.0
C_{12}	60, 120	60, 120	60, 120	60.0, 120.0
C_{13}	60, 120	60, 120	60, 120	69.8, 129.8
C_{15}	60, 120	60, 120	50.3, 120	57.6, 120.0
C_{17}	80, 100	80, 100	74.2, 100	79.8, 94.9
C_{18}	80, 100	80, 100	37.0, 100	90.6, 118.9

Table 7.21: Influx of traffic to Kvadraturen for optimization case A. The influx is calculated as $f_{\text{in}} = \gamma\rho(1-\rho)$, with $\gamma = \frac{v_{\text{in}}}{L}$.

Variable	ρ	**Control Points** [s]	v_{in} [km/t]	L [m]
F_1^1	0.2, 0.1, 0.2	300, 600	30	50
F_2^1	0.3, 0.4, 0.25	100, 400	50	50
F_3^1	0.6, 0.3, 0.42	200, 800	50	50
F_4^A	0.8	—	80	50
F_4^B	1.0	—	80	50
F_5^A	0.8	—	80	50
F_5^B	1.0	—	80	50
F_6	0.35	—	50	50

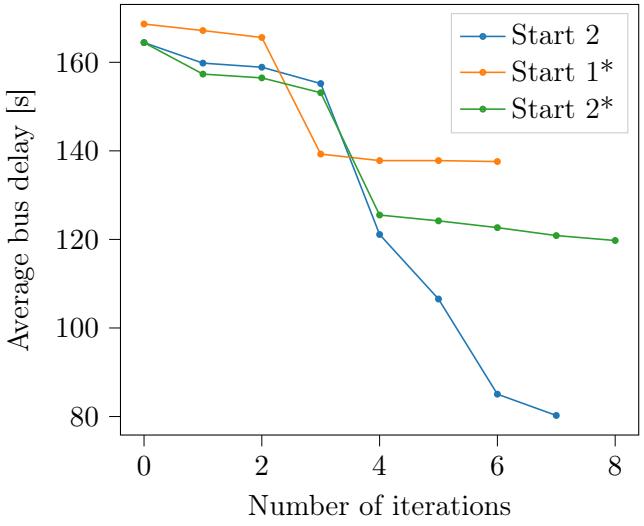
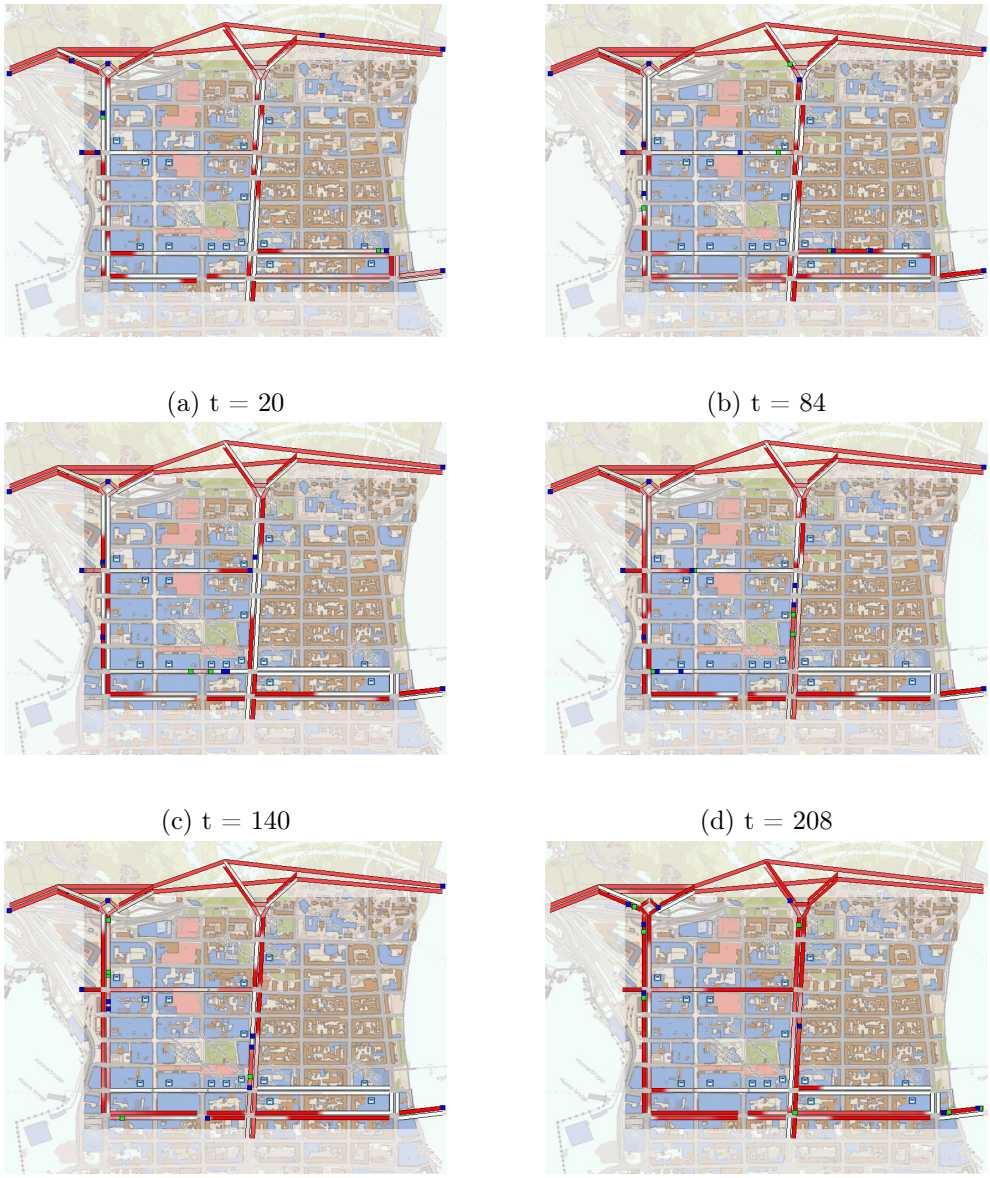


Figure 7.31: Average delay of the buses as a function of the number of iterations for the starting points listed in Table 7.20 and influx listed in Table 7.21.

between different choices of control parameters. A comparison between the flow of traffic with the starting point 1 from Table 7.20 compared with the local optimum that was reached can be seen at `torjeihelset.github.io/master_project`. We include some snapshots from this video in Figure 7.32. The exact details of the best set of control parameters are also listed in Table 7.20.



(a) t = 20 (b) t = 84

(c) t = 140 (d) t = 208

(e) t = 390 (f) t = 740

Figure 7.32: Snapshots of the simulation for the optimal speed limit of the Kvadraturen network. The amount of traffic is visualized with a red color scale. White means that there is no traffic, and a darker red color means that the road is congested. The green squares shows the positions of the buses with optimal control parameters. The blue squares shows the position of the bus with non optimal control parameters.

Table 7.22: Convergence history of the optimization method from the starting points listed in Table 7.20 and influx listed in Table 7.21 using objective function (5.6).

Start	**Iteration**	$J(p)$
1	**0**	**168.6429**
2	0	164.4464
2	1	159.8141
2	2	158.8903
2	3	155.2010
2	4	121.1154
2	5	106.5422
2	6	85.0490
2	**7**	**80.2426**
1∗	0	168.6429
1∗	1	167.1718
1∗	2	165.6042
1∗	3	139.2728
1∗	4	137.8036
1∗	5	137.7996
1∗	**6**	**137.5849**
2∗	0	164.4464
2∗	1	157.3366
2∗	2	156.4803
2∗	3	153.1234
2∗	4	125.5091
2∗	5	124.1884
2∗	6	122.6626
2∗	7	120.8696
2∗	**8**	**119.7531**

Chapter 8

Conclusions

In this thesis, we have looked at how the management of traffic can be improved by making better use of the existing infrastructure. This improvement is due to setting better speed limits and determining better settings for traffic lights. We refer to the speed limits and settings of traffic lights as the control parameters of a traffic-flow network. We have derived a mathematical model for a unidirectional road, modelling the density of vehicles as a continuous "fluid", before extending the model to include junctions, roundabouts, traffic lights and public transport such as buses. Aiming to determine the best set of control parameters of a traffic network, we introduced several ideas that to the best of our knowledge have not been considered before.

- We have looked at how automatic differentiation can be applied to numerical simulation of traffic flow to obtain optimal control parameters.
- We have introduced a new way of determining how traffic merges that depends on the densities of traffic on the roads.
- We have introduced a new way of implementing yielding rules.
- We have looked at a new of modelling traffic lights.

Combining all of these elements, we have analysed how different choices of control parameters impact the flow of traffic for various road networks. For some of these networks, we only considered the delay of public transport, while for others we also considered different objective functionals related to the general flow of traffic.

In future work, we would like to analyse the validity of the new ideas we have presented, and how accurately they model real traffic. This can be done by, e.g., comparing our numerical results to either real traffic data or to numerical results from other models.

We would also like to analyse the optimal control problem in more detail. In particular, it would be interesting to consider by a concrete example the constrained optimal control problem discussed in Section 5.2.2, and to compare the results to results obtained when considering only the delay of public transport, or only the flow of traffic.

In Section 7.1, Section 7.2 and Section 7.3 we looked at the properties of some of the objective functionals from Section 5.1. We observed that in some cases, the objective functionals develop irregularities, which we believe are due to numerical errors. We have not performed any testing for how the optimal control problem is affected by these irregularities. In particular, we would like to check whether the best control parameters from the optimization method correspond to a minima also when performing the simulation on a finer grid. If this is the case, or the two minima at least are close to each other, it could be possible to perform an initial optimization on a coarse grid, before starting from the stationary point using a finer grid.

For all of our numerical experiments, we made use of a modified steepest descent algorithm as discussed in Section 5.2.1. It would therefore be interesting to analyse the optimal control problem by applying different optimization techniques such as, e.g., integer optimization or sequential quadratic programming. We would like to compare both the resulting control parameters as well as the computational effort required for the different optimization techniques. We briefly discussed in Section 7.5.1 that the optimal control problem might be poorly scaled in some cases. Hence, we would like to investigate the behaviour of optimization methods that are not sensitive to poor scaling.

Finally, in Chapter 4, we only considered one way of determining the merging of traffic based on the densities on the roads, and so it would be interesting to also consider different ways of determining the merging and to compare this to the method presented in this thesis.

Bibliography

[1] D. B. Lee, L. A. Klein and G. Camus, 'Induced Traffic and Induced Demand', en, *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1659, no. 1, pp. 68–75, Jan. 1999, ISSN: 0361-1981, 2169-4052. DOI: 10.3141/1659-09.

[2] *THE 17 GOALS | Sustainable Development*. [Online]. Available: https://sdgs.un.org/goals (visited on 04/06/2024).

[3] T. Helset, 'Speed limit and traffic-light control for traffic-flow networks', Department of Mathematical Sciences NTNU – Norwegian University of Science and Technology, Project report in TMA4500, Dec. 2023.

[4] M. J. Lighthill and G. B. Whitham, 'On kinematic waves II. A theory of traffic flow on long crowded roads', *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, vol. 229, no. 1178, pp. 317–345, Jan. 1997, Publisher: Royal Society. DOI: 10.1098/rspa.1955.0089.

[5] M. J. Lighthill and G. B. Whitham, 'On kinematic waves. i. flood movement in long rivers', *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, vol. 229, no. 1178, pp. 281–316, 1955, ISSN: 00804630.

[6] P. I. Richards, 'Shock Waves on the Highway', *Operations Research*, vol. 4, no. 1, pp. 42–51, Feb. 1956, Publisher: INFORMS, ISSN: 0030-364X. DOI: 10.1287/opre.4.1.42.

[7] H. Greenberg, 'An Analysis of Traffic Flow', *Operations Research*, vol. 7, no. 1, pp. 79–85, Feb. 1959. DOI: 10.1287/opre.7.1.79.

[8] N. Gartner, C. Messer and A. Rathi, *Revised Monograph on Traffic Flow Theory*. Jan. 2001.

[9] P. Goatin, 'The aw–rascle vehicular traffic flow model with phase transitions', *Mathematical and Computer Modelling*, vol. 44, no. 3, pp. 287–303, 2006, ISSN: 0895-7177. DOI: 10.1016/j.mcm.2006.01.016.

[10] G. F. Newell, 'Nonlinear Effects in the Dynamics of Car Following', en, *Operations Research*, Apr. 1961, Publisher: INFORMS. DOI: 10.1287/opre.9.2.209.

[11] M. Burger, S. Göttlich and T. Jung, 'Derivation of a first order traffic flow model of Lighthill-Whitham-Richards type', en, *IFAC-PapersOnLine*, vol. 51, no. 9, pp. 49–54, 2018, ISSN: 24058963. DOI: 10.1016/j.ifacol.2018.07.009.

[12] S. Göttlich, E. Iacomini and T. Jung, 'Properties of the lwr model with time delay', *Networks and Heterogeneous Media*, vol. 16, no. 1, pp. 31–47, 2021, ISSN: 1556-1801. DOI: 10.3934/nhm.2020032.

[13] J. A. Laval and L. Leclercq, 'A mechanism to describe the formation and propagation of stop-and-go waves in congested freeway traffic', *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 368, no. 1928, pp. 4519–4541, Oct. 2010, Publisher: Royal Society. DOI: 10.1098/rsta.2010.0138.

[14] B. Goñi-Ros, V. L. Knoop, B. van Arem and S. P. Hoogendoorn, 'Empirical analysis of the causes of stop-and-go waves at sags', en, *IET Intelligent Transport Systems*, vol. 8, no. 5, pp. 499–506, 2014, ISSN: 1751-9578. DOI: 10.1049/iet-its.2013.0102.

[15] A. Aw and M. Rascle, 'Resurrection of "second order" models of traffic flow', *SIAM Journal on Applied Mathematics*, vol. 60, no. 3, pp. 916–938, 2000. DOI: 10.1137/S0036139997332099.

[16] R. Colombo, 'A 2×2 hyperbolic traffic flow model', *Mathematical and Computer Modelling - MATH COMPUT MODELLING*, vol. 35, pp. 683–688, Mar. 2002. DOI: 10.1016/S0895-7177(02)80029-2.

[17] R. J. LeVeque, *Finite Volume Methods for Hyperbolic Problems*, ser. Cambridge Texts in Applied Mathematics. Cambridge: Cambridge University Press, 2002, ISBN: 978-0-521-00924-9. DOI: 10.1017/CBO9780511791253.

[18] A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J. K. Moore, S. Singh, T. Rathnayake, S. Vig, B. E. Granger, R. P. Muller, F. Bonazzi, H. Gupta, S. Vats, F. Johansson, F. Pedregosa, M. J. Curry, A. R. Terrel, Š. Roučka, A. Saboo, I. Fernando, S. Kulal, R. Cimrman and A. Scopatz, 'Sympy: Symbolic computing in python', *PeerJ Computer Science*, vol. 3, e103, Jan. 2017, ISSN: 2376-5992. DOI: 10.7717/peerj-cs.103.

[19] B. Speelpenning, 'Compiling fast partial derivatives of functions given by algorithms', Jan. 1980. DOI: 10.2172/5254402.

[20] T.A.A.B., 'An introduction to the calculus of finite differences. by c.h. richardson pp. vi, 142. 28s. 1954. (van nostrand, new york; macmillan, london)', *The Mathematical Gazette*, vol. 39, no. 330, pp. 339–339, 1955. DOI: 10.2307/3608616.

[21] A. Griewank and A. Walther, *Evaluating Derivatives*, Second.: Society for Industrial and Applied Mathematics, 2008. DOI: 10.1137/1.9780898717761.

[22] D. Cortild, J. Haastert, A. Sanabria and F. Voronine, *A Brief Review of Automatic Differentiation*. Mar. 2023. DOI: 10.13140/RG.2.2.19959.70565.

[23] H. Holden and N. H. Risebro, 'A mathematical model of traffic flow on a network of unidirectional roads', *SIAM Journal on Mathematical Analysis*, vol. 26, no. 4, pp. 999–1017, Jul. 1995, ISSN: 0036-1410, 1095-7154. DOI: 10.1137/S0036141093243289.

[24] M. Garavello and B. Piccoli, *Traffic Flow on Networks: Conservation Laws Models*, ser. AIMS series on applied mathematics. American Institute of Mathematical Sciences, 2006, ISBN: 978-1-60133-000-0.

[25] G. M. Coclite, M. Garavello and B. Piccoli, 'Traffic flow on a road network', *SIAM Journal on Mathematical Analysis*, vol. 36, no. 6, pp. 1862–1886, 2005. DOI: 10.1137/S0036141004402683.

[26] G. Bretti and B. Piccoli, 'A tracking algorithm for car paths on road networks', *SIAM Journal on Applied Dynamical Systems*, vol. 7, no. 2, pp. 510–531, 2008. DOI: 10.1137/070697768.

[27] P. Goatin and E. Rossi, 'Comparative study of macroscopic traffic flow models at road junctions', en, *Networks and Heterogeneous Media*, vol. 15, no. 2, pp. 261–279, Apr. 2020, Publisher: Networks and Heterogeneous Media, ISSN: 1556-1801. DOI: 10.3934/nhm.2020012.

[28] P. Goatin, S. Göttlich and O. Kolb, 'Speed limit and ramp meter control for traffic flow networks', *Engineering Optimization*, vol. 48, no. 7, pp. 1121–1144, 2nd Jul. 2016, ISSN: 0305-215X, 1029-0273. DOI: 10.1080/0305215X.2015.1097099.

[29] 'Hermite Interpolation', in *Curves and Surfaces for Computer Graphics*, New York, NY: Springer New York, 2006, pp. 111–139, ISBN: 978-0-387-28452-1. DOI: 10.1007/0-387-28452-4_4.

[30] P. G. Ciarlet and P. A. Raviart, 'General lagrange and hermite interpolation in Rn with applications to finite element methods', en, *Archive for Rational Mechanics and Analysis*, vol. 46, no. 3, pp. 177–199, Jan. 1972, ISSN: 1432-0673. DOI: 10.1007/BF00252458.

[31] Y. Chitour and B. Piccoli, 'Traffic circles and timing of traffic lights for cars flow', *Discrete Contin. Dyn. Syst. Ser. B*, vol. 5, pp. 599–630, Aug. 2005. DOI: 10.3934/dcdsb.2005.5.599.

[32] L. Obsu, M. L. Delle Monache, P. Goatin and S. Kassa, 'Traffic Flow Optimization on Roundabouts', *Mathematical Methods in the Applied Sciences*, vol. 38, pp. 3075–3096, Sep. 2015. DOI: 10.1002/mma.3283.

[33] E. Chevallier and L. Leclercq, 'A macroscopic single-lane roundabout model to account for insertion delays and o–d patterns', *Computer-Aided Civil and Infrastructure Engineering*, vol. 23, no. 2, pp. 104–115, 2008. DOI: https://doi.org/10.1111/j.1467-8667.2007.00527.x.

[34] J. Lebacque, J. Lesort and F. Giorgi, 'Introducing Buses into First-Order Macroscopic Traffic Flow Models', en, *Transportation Research Record*, vol. 1644, no. 1, pp. 70–79, Jan. 1998, Publisher: SAGE Publications Inc, ISSN: 0361-1981. DOI: 10.3141/1644-08.

[35] I. Gasser, C. Lattanzio and A. Maurizi, 'Vehicular traffic flow dynamics on a bus route', *Multiscale Modeling & Simulation*, vol. 11, no. 3, pp. 925–942, 2013. DOI: 10.1137/130906350.

[36] A. Cutolo, C. D'Apice and R. Manzo, 'Traffic Optimization at Junctions to Improve Vehicular Flows', *ISRN Applied Mathematics*, vol. 2011, E. J. Sellountos, Ed., p. 679 056, Sep. 2011, Publisher: International Scholarly Research Network, ISSN: 2356-7872. DOI: 10.5402/2011/679056.

[37] M. Gugat, M. Herty, A. Klar and G. Leugering, 'Optimal Control for Traffic Flow Networks', en, *Journal of Optimization Theory and Applications*, vol. 126, no. 3, pp. 589–616, Sep. 2005, ISSN: 1573-2878. DOI: 10.1007/s10957-005-5499-z.

[38] M. Herty and A. Klar, 'Modeling, simulation, and optimization of traffic flow networks', *SIAM Journal on Scientific Computing*, vol. 25, no. 3, pp. 1066–1087, 2003. DOI: 10.1137/S106482750241459X.

[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: Emerging Technologies*, vol. 13, no. 3, pp. 185–209, 2005, ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2004.08.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X05000264.

[40] J. Nocedal and S. J. Wright, *Numerical Optimization*. Springer New York, NY, 2006. DOI: https://doi.org/10.1007/978-0-387-40065-5.

[41] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga and A. Lerer, 'Automatic differentiation in pytorch', in *NIPS 2017 Workshop on Autodiff*, Long Beach, California, USA, 2017. [Online]. Available: https://openreview.net/forum?id=BJJsrmfCZ.

Appendix A

Additional Calculations and Details

In this appendix we include some additional calculations that were not included in the main text in order to improve the flow of the text.

A.1 Exact Flux Distribution in 3-to-1 Junctions

We investigate the optimal solution of a 3-to-1 junction in detail. For this junction, there are no crossing connections, and so the upper bounds of the fluxes are given by the demand functions. We start by writing out the incoming fluxes using (4.41) as

$$\begin{aligned} f_1 &= \min\{D_1, \max\{\beta_1 S_r, S_r - D_2 - D_3\}\}, \\ f_2 &= \min\{D_2, \max\{\beta_2 S_r, S_r - D_1 - D_3\}\}, \\ f_3 &= \min\{D_3, \max\{\beta_3 S_r, S_r - D_1 - D_2\}\}. \end{aligned} \tag{A.1}$$

In this case, the feasible region takes the form of a box on the form

$$\Omega = \{(f_1, f_2, f_3) : 0 \leq f_i \leq D_i, i = 1, 2, 3\},$$

where f_i denotes the flux leaving incoming road l_i. Letting f_r denote the flux entering the outgoing road, we maximize the flux by setting

$$f_r = \min\{S_r, D_1 + D_2 + D_3\},$$

where S_r denotes the capacity of the outgoing road. To simplify the notation, we define the sum of the demands as $D := D_1 + D_2 + D_3$. The set of points maximizing the flux is then given by the set

$$Q = (f_1, f_2, f_3) : f_1 + f_2 + f_3 = S_r.$$

In the demand limited case, where $S_r \geq D$, we choose the point in the feasible region maximizing the flux. Thus, the optimal point is given by

$$(f_1^*, f_2^*, f_3^*, f_4^*) = (D_1, D_2, D_3, D).$$

Assume to the contrary that we are in the supply limited case where $S_r < D$. In this case the intersection between the feasible region Ω and the plane maximizing the flux Q is defined as the convex set K defined as

$$K := \{(f_1, f_2, f_3) : f_1 + f_2 + f_3 = S_r; 0 \leq f_i \leq D_i, i = 1, 2, 3\}.$$

It is clear that we need some extra conditions to find a unique optimum from the set K. We generalize the rule (C), and let C denote the total amount of traffic entering outgoing road r. With the priority parameters β_i defined, we want $\beta_i C$ of the traffic to come from road l_i. The set of points satisfying this condition is given by the line r given by

$$\begin{cases} f_3 = \frac{\beta_3}{\beta_1} f_1, \\ f_3 = \frac{\beta_3}{\beta_2} f_2. \end{cases}$$

We now find the point P_I as the unique point on the line r and intersecting the plane Q as

$$P_I = (S_r\beta_1, S_r\beta_2, S_r\beta_3).$$

We notice that since $S_r\beta_1 + S_r\beta_2 + S_r\beta_3 = S_r$ and since

$$S_r\beta_3 = \frac{\beta_3}{\beta_i}\beta_i S_r \text{ for } i = 1, 2,$$

the point lies on both the line r and the plane Q and must therefore be the intersection point. We further note that $S_r\beta_i > 0$ for $i = 1, 2, 3$. We choose the fluxes entering the junction as the unique point $(f_1^*, f_2^*, f_3^*) \in K$ minimizing the distance between the convex set K and the intersection point P_I. That is

$$(f_1^*, f_2^*, f_3^*) = \underset{(f_1,f_2,f_3)\in K}{\arg\min}\ d((f_1, f_2, f_3), P_I),$$

where $d(\cdot, \cdot)$ denotes the Euclidean distance in $\mathbb{R}^3$. If the intersection point P_I lies inside the convex set $P_I \in K$, the optimal point will simply be the intersection point. Assuming that $P_I \notin K$, we need to find the point on the boundary of K minimizing the distance to P_I.

Assuming that $S_r \geq D_i + D_j$ for $i, j \in \{1, 2, 3\}, i \neq j$, the convex set K takes the form of a triangle defined by the three vertices

$$\begin{aligned} A &= (D_1, D_2, S_r - D_1 - D_2), \\ B &= (D_1, S_r - D_1 - D_3, D_3), \\ C &= (S_r - D_2 - D_3, D_2, D_3). \end{aligned} \tag{A.2}$$

Figure A.1 shows the convex set K and a possible location of P_I. In this case, the closest point to P_I will either be on one of the vertices, or it will lie on one

of the lines connecting the vertices. We start by writing the equation for the lines connecting the vertices:

$$\begin{aligned}
AB: &\begin{bmatrix} D_1 \\ D_2 \\ S_r - D_1 - D_2 \end{bmatrix} + \tilde{t} \begin{bmatrix} 0 \\ S - D \\ D - S \end{bmatrix} = \begin{bmatrix} D_1 \\ D_2 \\ S_r - D_1 - D_2 \end{bmatrix} + t \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}, \\
BC: &\begin{bmatrix} D_1 \\ S_r - D_1 - D_3 \\ D_3 \end{bmatrix} + \tilde{t} \begin{bmatrix} S_r - D \\ D - S_r \\ 0 \end{bmatrix} = \begin{bmatrix} D_1 \\ S_r - D_1 - D_3 \\ D_3 \end{bmatrix} + t \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix}, \\
CA: &\begin{bmatrix} S_r - D_2 - D_3 \\ D_2 \\ D_3 \end{bmatrix} + \tilde{t} \begin{bmatrix} D - S_r \\ 0 \\ S_r - D \end{bmatrix} = \begin{bmatrix} S_r - D_2 - D_3 \\ D_2 \\ D_3 \end{bmatrix} + t \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix},
\end{aligned} \tag{A.3}$$

where $\tilde{t} \in [0, 1]$ and $t = D - S_r\tilde{t} \in [0, D - S_r]$. We now make the important realization that f_1 is constant along AB, f_2 is constant along CA and f_3 is constant along BC.

With the aim of determining the optimal solution, we will first split the plane Q into some different segments. First, we note that since $f_1 = D_1$ along AB, whenever the intersection point satisfies $P_I = (\tilde{f}_1, \tilde{f}_2, \tilde{f}_2) : \tilde{f}_1 > D_1$, the optimal value will lie somewhere on the AB line. Similarly, if $\tilde{f}_2 > D_2$ the optimal value will lie somewhere on the CA line, and if $\tilde{f}_3 > D_3$ the optimal value will lie somewhere on the BC line. This means that if $\tilde{f}_i > D_i$ for two of the i's, the optimal value will be one of the vertices of the triangle.

To be able to find the correct points on the boundary of the triangle, we start by finding the three unit vectors that are perpendicular to the plane Q and the three line segments respectively. To this end, we first write the unit normal of the plane as

$$\overrightarrow{n} = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

We also note that the three line segments are parallel to the three vectors

$$\overrightarrow{v}_{\mathrm{AB}} = \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}, \quad \overrightarrow{v}_{\mathrm{BC}} = \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix}, \quad \overrightarrow{v}_{\mathrm{CA}} = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}.$$

Finding the vector perpendicular to two vectors can be done by taking the cross product of the two vectors. We calculate, ignoring the scaling factor $\frac{1}{\sqrt{3}}$

$$\overrightarrow{p}_{\mathrm{AB}} = \overrightarrow{v}_{\mathrm{AB}} \times \overrightarrow{n} = \begin{vmatrix} \overrightarrow{i} & \overrightarrow{j} & \overrightarrow{k} \\ 1 & 1 & 1 \\ 0 & -1 & 1 \end{vmatrix} = \begin{bmatrix} 2 \\ -1 \\ -1 \end{bmatrix},$$

$$\overrightarrow{p}_{\mathrm{BC}} = \overrightarrow{v}_{\mathrm{BC}} \times \overrightarrow{n} = \begin{vmatrix} \overrightarrow{i} & \overrightarrow{j} & \overrightarrow{k} \\ 1 & 1 & 1 \\ -1 & 1 & 0 \end{vmatrix} = \begin{bmatrix} -1 \\ -1 \\ 2 \end{bmatrix}$$

and

$$\overrightarrow{p}_{\mathrm{CA}} = \overrightarrow{v}_{\mathrm{CA}} \times \overrightarrow{n} = \begin{vmatrix} \overrightarrow{i} & \overrightarrow{j} & \overrightarrow{k} \\ 1 & 1 & 1 \\ 1 & 0 & -1 \end{vmatrix} = \begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix}.$$

We now have all the tools needed for segmenting the Q plane and to determine the optimal value depending on where the intersection point $P_I = (\tilde{f}_1, \tilde{f}_2, \tilde{f}_3)$ lies. Using the notation of segments as shown in Figure A.1, we will explain how to determine which segment the intersection point lies in, and how the optimal solution can be found depending on which segment it lies in.

First, we see that if $\tilde{f}_1 \geq D_1$ and $\tilde{f}_2 \geq D_2$, we know that $P_I \in Q_{\mathrm{A}}$. Similarly if $\tilde{f}_1 \geq D_1$ and $\tilde{f}_3 \geq D_3$, then $P_I \in Q_{\mathrm{B}}$ and if $\tilde{f}_2 \geq D_2$ and $\tilde{f}_3 \geq D_3$, then $P_I \in Q_{\mathrm{C}}$. In these cases, the optimal solution will be one of the vertices. If i.e., $P_I \in Q_{\mathrm{A}}$, then the optimal point will be the vertex A.

Assume now that $\tilde{f}_1 \geq D_1, \tilde{f}_2 < D_2$ and $\tilde{f}_3 < D_3$. In this case we know that

$$P_I \in Q^L_{\mathrm{AB}} \cup Q_{\mathrm{AB}} \cup Q^R_{\mathrm{AB}},$$

but we need to determine in which of these sets P_I lies in. This we will do by making use of the normal vector $\overrightarrow{p}_{\mathrm{AB}}$, and finding the intersection between the line

$$P_I + s \cdot \overrightarrow{p}_{\mathrm{AB}}, s \in \mathbb{R},$$

and the extended line

$$AB_{\mathrm{ext}} = \begin{bmatrix} D_1 \\ D_2 \\ S_r - D_1 - D_2 \end{bmatrix} + t \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}, t \in \mathbb{R}.$$

Depending on the value of t at the intersection we can find out in which set the point P_I lies in. The intersection point can be found by solving the set of equations

$$\begin{aligned} D_1 &= \tilde{f}_1 + 2s, \\ D_2 - t &= \tilde{f}_2 - s, \\ S_r - D_1 - D_2 + t &= (S_r - \tilde{f}_1 - \tilde{f}_2) - s. \end{aligned} \tag{A.4}$$

We solve these equations for s and t and obtain

$$s = \frac{D_1 - \tilde{f}_1}{2}, \quad t = D_2 - \tilde{f}_2 + \frac{D_1 - \tilde{f}_1}{2}.$$

If $t \leq 0$ we know that $P_I \in Q^L_{\mathrm{AB}}$ and the vertex A is the optimal solution. Similarly, if $t \geq S_r - D$ then $P_I \in Q^R_{\mathrm{AB}}$ and the vertex B is the optimal value. Otherwise, the intersection point is the optimal solution. That is

$$(f_1^*, f_2^*, f_3^*) = (D_1, D_2 - t, S_r - D_1 - D_2 + t).$$

We note that even though it looks like $D_2 - t$ could potentially be negative, since

$$D_2 - t = D_2 - D_2 + \tilde{f}_2 + \frac{\tilde{f}_1 - D_1}{2} = \tilde{f}_2 + \frac{\tilde{f}_1 - D_1}{2} > 0,$$

all components of the optimal solution are positive.

We calculate the optimal solution for the remaining cases $\tilde{f}_2 > D_2$ and $\tilde{f}_3 > D_3$ in a similar fashion.

When $\tilde{f}_2 > D_2$, we look for the intersection between

$$P_I + s \cdot \overrightarrow{p}_{\mathrm{CA}}, s \in \mathbb{R},$$

and the extended line

$$CA_{\mathrm{ext}} = \begin{bmatrix} S_r - D_2 - D_3 \\ D_2 \\ D_3 \end{bmatrix} + t \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}, t \in \mathbb{R}.$$

This leads to the set of equations

$$\begin{aligned} S_r - D_2 - D_3 + t &= (S_r - \tilde{f}_2 - \tilde{f}_3) - s, \\ D_2 &= \tilde{f}_2 + 2s, \\ D_3 - t &= \tilde{f}_3 - s. \end{aligned} \tag{A.5}$$

with the solution

$$s = \frac{D_2 - \tilde{f}_2}{2}, \quad t = D_3 - \tilde{f}_3 + \frac{D_2 - \tilde{f}_2}{2}.$$

As before, if $t < 0$ then the optimal solution is equal to the vertex C, and if $t > S_r - D$ then the optimal solution is equal to the vertex B. For other values of t the optimal value is given by

$$(f_1^*, f_2^*, f_3^*) = (S_r - D_2 - D_3 + t, D_2, D_3 - t).$$

Finally, we consider the case when $\tilde{f}_3 > D_3$. We now look for the intersection between

$$P_I + s \cdot \overrightarrow{p}_{\mathrm{BC}}, s \in \mathbb{R}$$

and the extended line

$$BC_{\mathrm{ext}} = \begin{bmatrix} D_1 \\ S_r - D_1 - D_3 \\ D_3 \end{bmatrix} + t \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix}, t \in \mathbb{R},$$

which leads to the set of equations

$$\begin{aligned} D_1 - t &= \tilde{f}_1 - s, \\ S_r - D_1 - D_3 + t &= (S_r - \tilde{f}_1 - \tilde{f}_2) - s, \\ D_3 &= \tilde{f}_3 + 2s. \end{aligned} \tag{A.6}$$

We solve for t and s and obtain

$$s = \frac{D_3 - \tilde{f}_3}{2}, \quad t = D_1 - \tilde{f}_1 + \frac{D_3 - \tilde{f}_3}{2}.$$

We conclude that when $t < 0$ the optimal solution is the vertex B, when $t > S_r - D$ the optimal solution is the vertex C and otherwise the solution is given by

$$(f_1^*, f_2^*, f_3^*) = (D_1 - t, S_r - D_1 - D_3 + t, D_3).$$

We will now get rid of the assumption that $S_r \geq D_i + D_j$ for $i, j \in \{1, 2, 3\}, i \neq j$, and assume only that $S_r > 0$ (otherwise the optimal solution would be $(0, 0, 0)$). In this case we can still draw the triangle with vertices as in (A.2). However, the three vertices of the triangle can potentially be outside the feasible region Ω, and so the convex set K is no longer represented by this triangle. However, the analysis we did for the case where $S_r \geq D_i + D_j$ for $i, j \in \{1, 2, 3\}, i \neq j$ can still be used also in this case. When $S_r < D_1 + D_2$, the third component of the vertex A will be negative. By continuity, also a small region close to the third component of the vertex will be negative. The line separating the positive and negative parts of the triangle is given by the line $f_3 = 0, f_1 + f_2 + f_3 = S_r$. This case is shown in Figure A.2. Since we know that all components of the right-of-way line r must be positive, we know that the intersection point P_I must lie to the right of the line $f_3 = 0$ as shown in Figure A.2. Thus, the point will either lie inside K, or the closest point will still be on the borders of the same triangle as before, with the condition that $f_3 \geq 0$. The possible locations of the intersection point is colored in a light gray color. For all these points, we know how to find the closest point on the triangle, which will also be the closest point in the convex set K.

Similarly if $S_r < D_1 + D_2$ and $S_r < D_2 + D_3$ even more of the triangle will be cut out. Nevertheless, the same way of projecting the intersection point onto the borders of the triangle hold. This case is shown in Figure A.3. The possible locations of the intersection point is colored in a light gray color.

In a similar way, when S_r decreases, the lines $f_1 = 0, f_2 = 0$ and $f_3 = 0$ get closer together. When $S_r < D_i, i = 1, 2, 3$, all possible locations of the intersection point P_I will lie inside the feasible region Ω and so the optimal solution will be equal to the intersection point. We conclude that for any $S_r > 0$, the above discussion can be used to find the optimal solution. We summarize the steps of the procedure for finding the optimal solution in the supply limited case $S_r < D_1 + D_2 + D_3$.

We calculate the intersection point as $P_I = (\beta_1 S_r, \beta_2 S_r, \beta_3 S_r)$ and separate into some different cases.

- Intersection point inside feasible region: If $\beta_i S_r \leq D_i, i = 1, 2, 3$ then the optimal solution is simply

$$(f_1^*, f_2^*, f_3^*) = (\beta_1 S_r, \beta_2 S_r, \beta_3 S_r).$$

- Closest point lies on a vertex: If $\beta_1 S_r > D_1$ and $\beta_2 S_r > D_2$ then the optimal point is given by

$$(f_1^*, f_2^*, f_3^*) = (D_1, D_2, S_r - D_1 - D_2).$$

If $\beta_1 S_r > D_1$ and $\beta_3 S_r > D_3$ then the optimal point is given by

$$(f_1^*, f_2^*, f_3^*) = (D_1, S_r - D_1 - D_3, D_3).$$

Finally, if $\beta_2 S_r > D_1$ and $\beta_3 S_r > D_3$, then the optimal point is given by

$$(f_1^*, f_2^*, f_3^*) = (S_r - D_2 - D_3, D_2, D_3).$$

- Closest point lies on the AB line: If $\beta_1 S_r > D_1$, we know that the optimal point lies on the AB line. We calculate

$$s = \frac{D_1 - \beta_1 S_r}{2}, \quad t = D_2 - \beta_2 S_r + \frac{D_1 - \beta_1 S_r}{2},$$

and find the optimal solution according to

$$(f_1^*, f_2^*, f_3^*) \begin{cases} (D_1, D_2, S_r - D_1 - D_2) & \text{if } t \leq 0, \\ (D_1, S_r - D_1 - D_3, D_3) & \text{if } t \geq D_1 + D_2 + D_3 - S_r, \\ (D_1, D_2 - t, S_r - D_1 - D_2 + t) & \text{otherwise.} \end{cases} \tag{A.7}$$

- Closest point lies on the CA line: If $\beta_2 S_r > D_2$, we know that the optimal point lies on the CA line. We calculate

$$s = \frac{D_2 - \beta_2 S_r}{2}, \quad t = D_3 - \beta_3 S_r + \frac{D_2 - \beta_2 S_r}{2},$$

and find the optimal solution according to

$$(f_1^*, f_2^*, f_3^*) \begin{cases} (S_r - D_2 - D_3, D_2, D_3) & \text{if } t \leq 0, \\ (D_1, D_2, S_r - D_1 - D_2) & \text{if } t \geq D_1 + D_2 + D_3 - S_r, \\ (S_r - D_2 - D_3 + t, D_2, D_3 - t) & \text{otherwise.} \end{cases} \tag{A.8}$$

- Closest point lies on the BC line: If $\beta_3 S_r > D_3$, we know that the optimal point lies on the BC line. We calculate

$$s = \frac{D_3 - \beta_3 S_r}{2}, \quad t = D_1 - \beta_1 S_r + \frac{D_3 - \beta_3 S_r}{2},$$

and find the optimal solution according to

$$(f_1^*, f_2^*, f_3^*) \begin{cases} (D_1, S_r - D_1 - D_3, D_3) & \text{if } t \leq 0, \\ (S_r - D_2 - D_3, D_2, D_3) & \text{if } t \geq D_1 + D_2 + D_3 - S_r, \\ (D_1 - t, S_r - D_1 - D_3 + t, D_3) & \text{otherwise.} \end{cases} \tag{A.9}$$

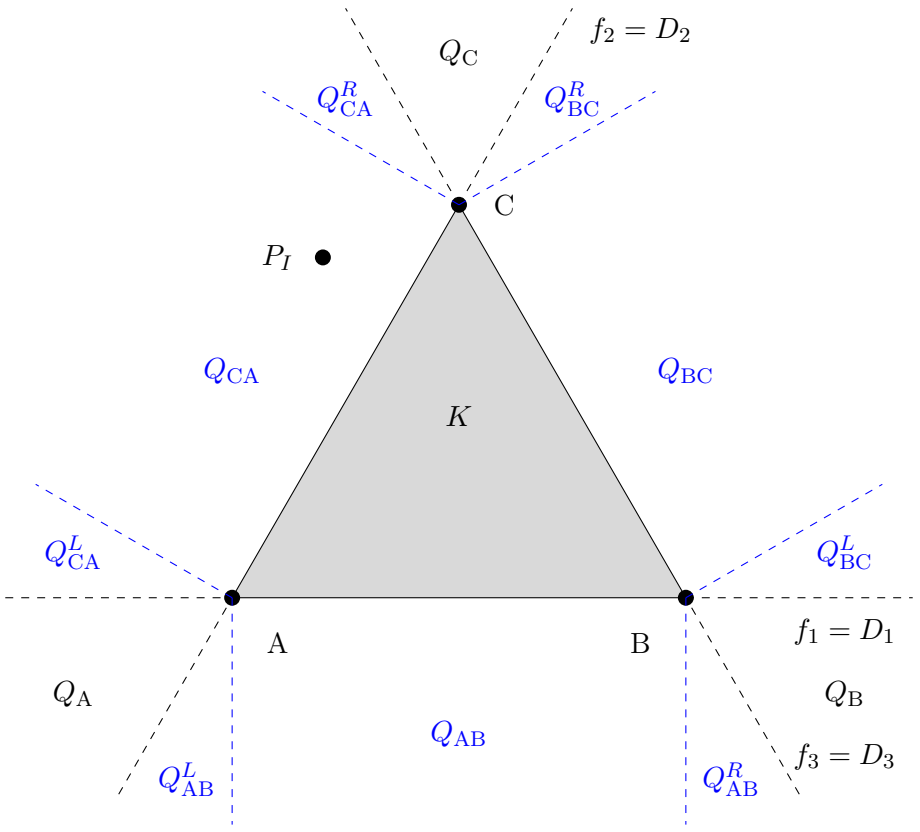


Figure A.1: The convex set $K \subset \mathbb{R}^3$ when $S \geq D_i + D_j$ for $i, j \in \{1, 2, 3\}, i \neq j$, with a possible location of the intersection point P_I.

What remains is to check if these optimal values are the same as the ones calculated from (A.1). To check this, we go through both the demand and the limited cases, and all possible locations for the intersection point P_I. We start with the demand limited case where

$$S_r \geq D_1 + D_2 + D_3.$$

In this case we know that the optimal solution should be equal to

$$(f_1^*, f_2^*, f_3^*) = (D_1, D_2, D_3).$$

Since we are in the demand limited case, we also know that

$$S_r - D_2 - D_3 \geq D_1.$$

Since $\max\{\beta_1 S_r, S_r - D_2 - D_3\} \geq S_r - D_2 - D_3$, we know that $f_1 = D_1$. By symmetry, we also conclude that $f_2 = D_2$ and $f_3 = D_3$, and so the equation (A.1) is correct in the demand limited case.

We now turn our focus to the supply limited case where

$$S_r < D_1 + D_2 + D_3.$$

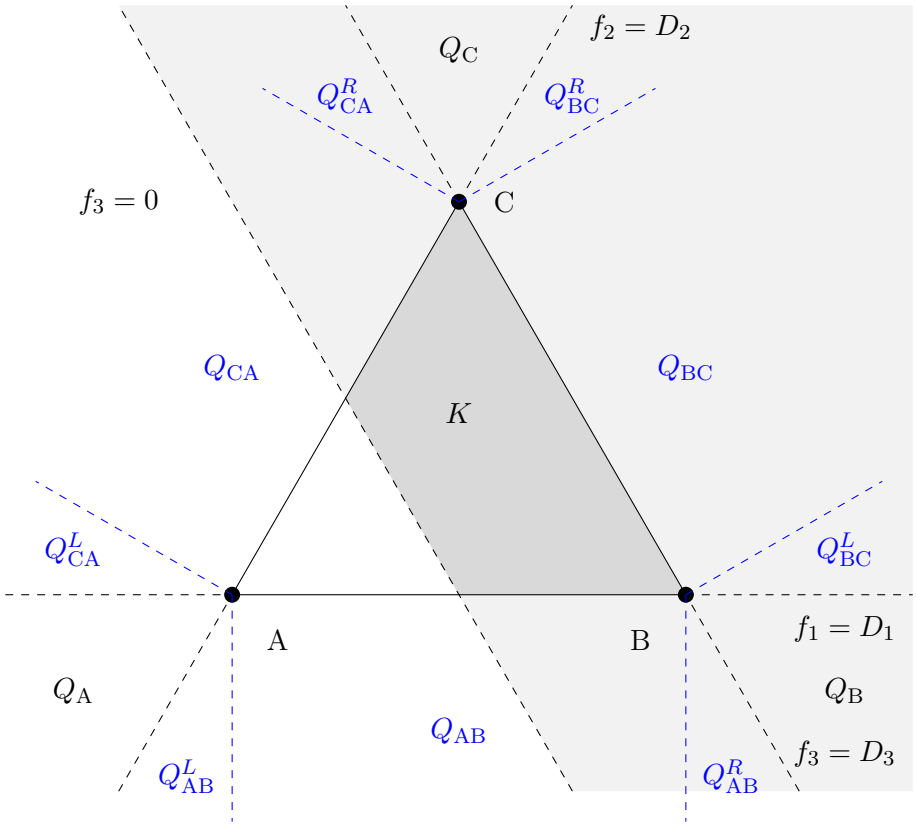


Figure A.2: The convex set $K \subset \mathbb{R}^3$ when $S_r < D_1 + D_2$.

We first look at the case where

$$\beta_1 S_r \leq D_1, \beta_2 S_r \leq D_2 \text{ and } \beta_3 S_r \leq D_3.$$

In this case we know the optimal solution to be

$$(f_1^*, f_2^*, f_3^*) = (\beta_1 S_r, \beta_2 S_r, \beta_3 S_r).$$

Since $S_r - D_2 - D_3 < D_1$ and $\beta_1 S_r \leq D_2$, we know that

$$f_1 = \max\{\beta_1 S_r, S_r - D_2 - D_3\}.$$

Furthermore, we know that

$$S_r - D_2 - D_3 = \beta_1 S_r + (\beta_2 S_r - D_2) + (\beta_3 S_r - D_3) \leq \beta_1 S_r,$$

which implies that

$$f_1 = \beta_1 S_r.$$

By symmetry, we conclude that the fluxes defined in (A.1) is correct also in this case.

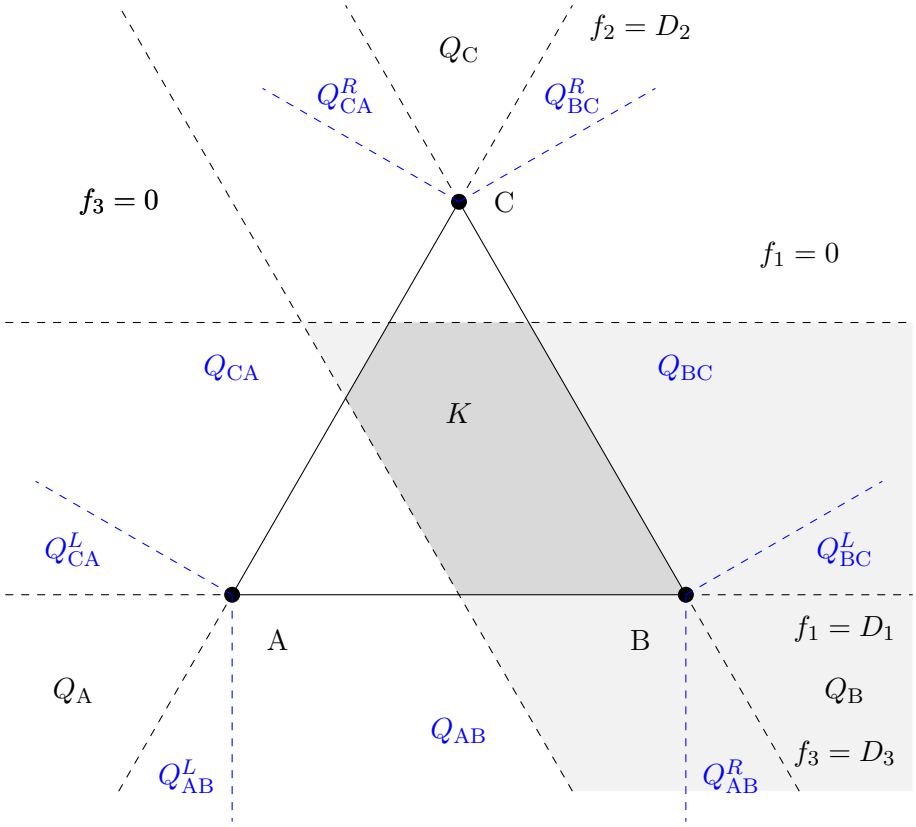


Figure A.3: The convex set $K \subset \mathbb{R}^3$ when $S_r < D_1 + D_2$ and $S_r < D_2 + D_3$

Now we will look at the cases where the intersection point P_I is located outside the triangle K. We start by looking at the cases where the optimal point lies on one of the vertices. By symmetry, we only look at one of these cases, and assume that $\beta_1 S_r \geq D_1$ and $\beta_2 S_r \geq D_2$, which leads to the optimal solution

$$(f_1^*, f_2^*, f_3^*) = (D_1, D_2, S_r - D_1 - D_2).$$

Since we assumed that $\beta_1 S_r \geq D_1$ we know that

$$S_r - D_2 - D_3 < D_1 \leq \beta_1 S - r,$$

and we also know that

$$f_1 = \min\{D_1, \beta_1 S - r\} = D_1.$$

The same argument can be used to show that $f_2 = D_2$. For the third component we have

$$S_r - D_1 - D_2 = \beta_3 S_r + (\beta_1 S_r - D_1) + (\beta_2 S_r - D_2) \geq \beta_3 S_r,$$

which implies that

$$\max\{\beta_3 S_r, S_r - D_1 - D_2\} = S_r - D_1 - D_2$$

which in turn implies that $f_3 = S_r - D_1 - D_2$. Hence, the flux definitions in (A.1) still holds true.

Finally we consider the case where the intersection point lies in the set

$$P_I \in Q^L_{\mathrm{AB}} \cup Q_{\mathrm{AB}} \cup Q^R_{\mathrm{AB}}$$

where the sets $Q^L_{\mathrm{AB}}, Q_{\mathrm{AB}}, Q^R_{\mathrm{AB}}$ are visualized in i.e., Figure A.1. Again, by symmetry, we only consider one of the line segments of the triangle. This is the case where

$$\beta_1 S_r > D_1, \beta_2 S_r < D_2, \text{ and } \beta_3 S_3 < D_3.$$

To find the optimal point we first calculate

$$s = \frac{D_1 - \beta_1 S_r}{2}, \quad t = D_2 - \beta_2 S_r + \frac{D_1 - \beta_1 S_r}{2},$$

and find the optimal solution according to

$$(f_1^*, f_2^*, f_3^*) \begin{cases} (D_1, D_2, S_r - D_1 - D_2) & \text{if } t \leq 0, \\ (D_1, S_r - D_1 - D_3, D_3) & \text{if } t \geq D_1 + D_2 + D_3 - S_r, \\ (D_1, D_2 - t, S_r - D_1 - D_2 + t) & \text{otherwise.} \end{cases} \tag{A.10}$$

Using (A.1), the first component can be computed using

$$\beta_1 S_r > D_1 > S_r - D_2 - D_3$$

which implies that

$$f_1 = \min\{D_1, \beta_1 S_r\} = D_1.$$

For the second and the third component we have

$$f_2 = \min\{D_2, \max\{\beta_2 S_r, S_r - D_1 - D_3\}\}, \quad f_3 = \min\{D_3, \max\{\beta_3 S_r, S_r - D_1 - D_3\}\}.$$

We first assume that $t \leq 0$. Clearly, for the second component we have

$$\beta_2 S_r < D_2, S_r - D_1 - D_3 < D_2 \implies f_2 < D_2.$$

For the third component we have

$$\begin{aligned} S_r - D_1 - D_2 &= (\beta_1 S_r - D_1) + (\beta_2 S_r - D_2) + \beta_3 S_r, \\ &= \beta_3 S_r - t + \frac{1}{2}(D_1 - \beta_1 S_r) > \beta_3 S_r, \\ &\implies f_3 = S_r - D_1 - D_2. \end{aligned}$$

When comparing with the exact solution we see that the first and third component are correct, but the second component is chosen too small.

We now assume that $t \geq D_1 + D_2 + D_3 - S_r$. Now, for the third component we have

$$\beta_3 S_r < D_3, S_r - D_1 - D_2 < D_3 \implies f_3 < D_3.$$

For the second component we have

$$S_r - D_1 - D_3 \geq D_2 - t = \beta_2 S_r - \frac{1}{2}(D_1 - \beta_1 S_r) > \beta_2 S_r,$$

which implies that $f_2 = S_r - D_1 - D_3$. Hence, in this case the two first fluxes calculated using (A.1) are correct, but the third one is chosen slightly too small.

Lastly we consider the case where $0 < t < D_1 + D_2 + D_3 - S_r$. The upper limit implies that

$$D_2 - t > D_2 - D_2 + S_r - D_1 - D_3 = S_r - D_1 - D_3.$$

In addition we have

$$D_2 - t = \beta_2 S_r - \frac{1}{2}(D_1 - \beta_1 S_r) > \beta_2 S_r.$$

Hence, for the second component we have

$$f_2 \leq \max\{\beta_2 S_r, S_r - D_1 - D_3\} < D_2 - t.$$

For the third component we clearly have

$$S_r - D_1 - D_2 + t > S_r - D_1 - D_2,$$

since $t > 0$. In addition we have that

$$S_r - D_1 - D_2 + t = (\beta_1 S_r - D_1) + (\beta_2 S_r - D_2) + \beta_3 S_r + t = \beta_3 S_r + t - t + \frac{1}{2}(\beta_1 S_r - D_1) > \beta_3 S_r.$$

Hence, the third component is chosen as

$$f_3 \leq \max\{\beta_3, S_r - D_1 - D_3\} < S_r - D_1 - D_2 + t.$$

Hence, in this case, both the second and third component are chosen too small.

A.2 Upper Bound on Flux Through Junction

In section 4.1.5 we introduced a way of calculating the upper bound on the amount of flux that is allowed through a junction when it crosses roads with higher priorities. The final expression included an integral that we have yet to describe. The integral contains a product of piece-wise linear functions, and hence it is possible to calculate explicitly. However, when the number of roads being crossed increases, the expression becomes quite complicated. Thus, instead of calculating explicitly, some quadrature formula will be made use of when the number of roads being crossed becomes too big.

A.2.1 One Road Being Crossed

In the case where only one road is being crossed, there is no need to introduce the random distribution and later taking the expectation. As in Section 4.1.5, we may imagine a crossing road being represented as a stick lying somewhere on the unit interval. When there is only one stick of length d, no matter where it is placed on the unit interval, the region not covered by it will be equal to $1-d$. Nevertheless, we will perform the calculation to check that expression matches our expectation. In the following subsections, we will refer to the length of the stick as d.

$0 < d < 0.5$

We recall from Section 4.1.5 that the probability that a point $x \in [0,1]$ is covered by a stick of length d is equal to

$$p(x; d_{l,k}) = \begin{cases} \frac{x}{1-d_{l,k}} & \text{if } x < d_{l,k}, \\ \frac{d_{l,k}}{1-d_{l,k}} & \text{if } d_{l,k} < x < 1 - d_{l,k}, \\ \frac{1-x}{1-d_{l,k}} & \text{if } 1 - d_{l,k} < x < 1, \end{cases} \tag{A.11}$$

when $0 < d < 0.5$. We also recall that the expected length of the region covered by at least one stick was equal to

$$E[L(w)] = \int_{x=0}^{1} 1 - \Pi_{s \in S_{i,j}}(1 - p(x; d))dx.$$

When only road is being crossed, this reduces to

$$E[L(w)] = \int_{x=0}^{1} 1 - (1 - p(x; d))dx = \int_{x=0}^{1} p(x; d)dx.$$

We plug in the expression for the probability, and split the integral into the regions of the piece-wise function to get

$$\begin{aligned} E[L(w)] = \int_{x=0}^{1} p(x; d)dx &= \frac{1}{1-d}(\int_0^d x dx + \int_d^{1-d} d dx + \int_{1-d}^{1} 1 - x dx) \\ &= \frac{1}{1-d}(\frac{1}{2}d^2 + d(1-d) + 1 - (1-d) - \frac{1}{2}(1 - (1-d)^2)) \\ &= \frac{1}{1-d}(\frac{1}{2}d^2 + d - 2d^2 + d - \frac{1}{2}(1 - 1 + 2d - d^2)) \\ &= \frac{1}{2}(-d^2 + d) = d\frac{1-d}{1-d} = d. \end{aligned} \tag{A.12}$$

That is, the expected value of the region covered by a stick of length d is equal to d when $0 < d < 0.5$, as expected.

$0.5 \leq d < 1$

We now consider the case where the length d is greater or equal to 0.5. We recall the expression for the probability in this case as

$$p(x; d_{l,k}) = \begin{cases} \frac{x}{1-d_{l,k}} & \text{if } x < 1 - d_{l,k}, \\ 1 & \text{if } 1 - d_{l,k} < x < d_{l,k}, \\ \frac{1-x}{1-d_{l,k}} & \text{if } d_{l,k} < x < 1. \end{cases} \tag{A.13}$$

We calculate the integral similarly to as in the previous subsection

$$\begin{aligned} E[L(w)] = \int_{x=0}^{1} p(x; d)dx &= \frac{1}{1-d}(\int_{0}^{1-d} xdx + \int_{1-d}^{d} 1 - ddx + \int_{d}^{1} 1 - xdx) \\ &= \frac{1}{1-d}(\frac{1}{2}(1-d)^2 + (1-d)(d-(1-d)) + 1 - d - \frac{1}{2}(1-d^2)) \\ &= \frac{1}{1-d}(\frac{1}{2} - d + \frac{1}{2}^2 + d - 1 + d - d^2 + d - d^2 + 1 - d - \frac{1}{2} + \frac{1}{2}d^2) \\ = \frac{1}{1-d}((\frac{1}{2} - 1 + 1 - \frac{1}{2}) + d(-1 + 1 + 1 + 1 - 1) + d^2(\frac{1}{2} - 1 - 1 + \frac{1}{2})) \\ &= \frac{1}{1-d}(d - d^2) = d. \end{aligned} \tag{A.14}$$

A.2.2 Two Roads Being Crossed

We now consider the case of two roads of higher priorities being crossed. Since the expressions for the probabilities vary depending on the lengths d we split into some different cases. We consider two sticks of lengths d_1 and d_2 respectively. We will assume without loss of generality that $d_1 \leq d_2$.

$0 < d_1, d_2 < 0.5$

We first consider the case where $0 < d_1, d_2 < 0.5$. The product of two piece-wise functions will still be a piece-wise function, and thus we split the integral

into its different parts.

$$\begin{aligned}
E[L(w)] = \int_0^1 1 - p(x;d_1)\cdot p(x;d_2)dx = 1 \\
- \int_0^{d_1} (1-\frac{x}{1-d_1})(1-\frac{x}{1-d_2})dx \\
- \int_{d_1}^{d_2} (1-\frac{d_1}{1-d_1})(1-\frac{x}{1-d_2})dx \\
- \int_{d_2}^{1-d_2} (1-\frac{d_1}{1-d_1})(1-\frac{d_2}{1-d_2})dx \\
- \int_{1-d_2}^{1-d_1} (1-\frac{d_1}{1-d_1})(1-\frac{1-x}{1-d_2})dx \\
- \int_{1-d_1}^{1} (1-\frac{1-x}{1-d_1})(1-\frac{1-x}{1-d_2})dx.
\end{aligned} \tag{A.15}$$

We now calculate the integrals separately

$$\begin{aligned}
I_1 := \int_0^{d_1} (1-\frac{x}{1-d_1})(1-\frac{x}{1-d_2})dx \\
= \int_0^{d_1} 1 - \frac{x}{1-d_2} - \frac{x}{1-d_1} + \frac{x^2}{(1-d_1)(1-d_2)}dx \\
= d_1 - \frac{d_1^2}{2(1-d_2)} - \frac{d_1^2}{2(1-d_1)} + \frac{d_1^3}{3(1-d_1)(1-d_2)}
\end{aligned} \tag{A.16}$$

Collecting all the terms using a common denominator we get

$$\begin{aligned}
I_1 = \frac{1}{(1-d_1)(1-d_2)}\big(d_1(1-d_1)(1-d_2) - \frac{1}{2}d_1^2(1-d_1) - \frac{1}{2}d_1^2(1-d_2) + \frac{1}{3}d_1^3\big) \\
= \frac{1}{(1-d_1)(1-d_2)}\big(d_1 - d_1d_2 - d_1^2 + d_1^2d_2 - \frac{1}{2}d_1 + \frac{1}{2}d_1^3 - \frac{1}{2}d_1^2 + \frac{1}{2}d_1^2d_2 + \frac{1}{3}d_1^3\big) \\
= \frac{d_1}{(1-d_1)(1-d_2)}\big(\frac{5}{6}d_1^2 + \frac{3}{2}d_1d_2 - 2d_1 - d_2 + 1\big).
\end{aligned} \tag{A.17}$$

For the second integral we have

$$\begin{aligned}
I_2 := \int_{d_1}^{d_2} (1-\frac{d_1}{1-d_1})(1-\frac{x}{1-d_2})dx \\
= (1-\frac{d_1}{1-d_1})(d_2 - d_1 - \frac{d_2^2-d_1^2}{2(1-d_2)}) \\
= \frac{d_1}{1-d_1}\Big(-\frac{3}{2}d_2^2 - d_1d_2 - d_1 + \frac{5}{2}d_1^2 + d_2 + 3d_1d_2^2 - 2d_1^2d_2 - d_1^3\Big)
\end{aligned} \tag{A.18}$$

To be able to sum to the two integrals together, we write this integral using the same denominator as before

$$\begin{aligned} I_2 &= \frac{1-d_1-d_1}{1-d_1}\left(\frac{(d_2-d_1)(1-d_2)-\frac{1}{2}(d_2^2-d_1^2)}{1-d_2}\right) \\ &= \frac{1-2d_1}{(1-d_1)(1-d_2)}\left(\frac{1}{2}d_1^2-\frac{3}{2}d_2^2-d_1+d_1d_2+d_2\right). \end{aligned} \tag{A.19}$$

We calculate the third integral

$$\begin{aligned} I_3 &:= \int_{d_2}^{1-d_2}\left(1-\frac{d_1}{1-d_1}\right)\left(1-\frac{d_2}{1-d_2}\right)dx \\ &= \left(1-\frac{d_1}{1-d_1}\right)\left(1-\frac{d_2}{1-d_2}\right)(1-d_2-d_2) \\ &= \left(\frac{1-2d_1}{1-d_1}\right)\left(\frac{1-2d_2}{1-d_2}\right)(1-d_2-d_2) \\ &= \frac{1}{(1-d_1)(1-d_2)}\left(1-4d_2+4d_2^2-2d_1+8d_1d_2-8d_1d_2^2\right). \end{aligned} \tag{A.20}$$

The fourth integral is calculated as

$$\begin{aligned} I_4 &:= \int_{1-d_2}^{1-d_1}\left(1-\frac{d_1}{1-d_1}\right)\left(1-\frac{1-x}{1-d_2}\right)dx \\ &= \left(1-\frac{d_1}{1-d_1}\right)\left((d_2-d_1)\left(1-\frac{1}{1-d_2}\right)+\frac{1}{2}\frac{(1-d_1)^2-(1-d_2)^2}{1-d_2}\right). \end{aligned} \tag{A.21}$$

Writing the fourth integral using the common denominator

$$I_4 = \frac{d_1}{1-d_1}\Big(-\frac{3}{2}d_2^2-d_1d_2-d_1+\frac{5}{2}d_1^2+d_2+3d_1d_2^2-2d_1^2d_2-d_1^3\Big). \tag{A.22}$$

We see that this is equal to I_2, which is due to the fact that the probabilities (and hence the product of probabilities) is symmetric around 0.5. Finally we calculate the fifth integral

$$\begin{aligned} I_5 &:= \int_{1-d_1}^{1}\left(1-\frac{1-x}{1-d_1}\right)\left(1-\frac{1-x}{1-d_2}\right)dx \\ &= \int_{1-d_1}^{1} 1-\frac{1-x}{1-d_2}-\frac{1-x}{1-d_1}+\frac{(1-x)(1-x)}{(1-d_1)(1-d_2)}dx \\ &= 1-(1-d_1)-\frac{d_1}{1-d_2}+\frac{1-(1-d_1)^2}{2(1-d_2)}-\frac{d_1}{1-d_1}+\frac{1-(1-d_1)^2}{2(1-d_1)} \\ &\quad +\frac{1}{(1-d_1)(1-d_2)}\int_{1-d_1}^{1} 1-2x+x^2dx \\ &= d_1-\frac{1}{1-d_2}\left(d_1-\frac{1}{2}(2d_1-d_1^2)\right)-\frac{1}{1-d_1}\left(d_1-\frac{1}{2}(2d_1-d_1^2)\right) \\ &\quad +\frac{1}{(1-d_1)(1-d_2)}\left(d_1-(1-(1-d_1)^2)+\frac{1}{3}(1-(1-d_1)^3)\right). \end{aligned} \tag{A.23}$$

As for the previous integrals, but skipping details, we write this integral using the common denominator

$$I_5 = \frac{d_1}{(1-d_1)(1-d_2)}\big(\frac{5}{6}d_1^2 + \frac{3}{2}d_1d_2 - 2d_1 - d_2 + 1\big). \tag{A.24}$$

Again, due to symmetry, we have $I_1 = I_5$. With all integrals expressed in the same way, we can sum them together. Skipping details, we get

$$\sum_{i=1}^{5} I_i = \frac{1}{(1-d_1)(1-d_2)}\Big(1 - \frac{1}{3}d_1^3 - d_1^2d_2 + d_1^2 - 2d_1d_2^2 + 4d_1d_2 - 2d_1 + d_2^2 - 2d_2\Big). \tag{A.25}$$

With the sum of the integrals calculated, we get the expected length covered

$$\begin{aligned} E[L(w)] &= 1 - \sum_{i=1}^{5} I_i \\ &= \frac{1}{(1-d_1)(1-d_2)}\left(1 - d_1 - d_2 + d_1d_2 - 1 + \frac{1}{3}d_1^3 + d_1^2d_2 - d_1^2 \right. \\ &\qquad \left. + 2d_1d_2^2 - 4d_1d_2 + 2d_1 - d_2^2 + 2d_2\right) \\ &= \frac{1}{(1-d_1)(1-d_2)}\left(\frac{1}{3}d_1^3 + d_1^2d_2 - d_1^2 + 2d_1d_2^2 - 3d_1d_2 + d_1 - d_2^2 + d_2\right). \end{aligned} \tag{A.26}$$

A.2.3 Evaluating the Integral with a Quadrature Formula

As we saw in Section A.2.2, evaluating the integral needed for calculating the upper bound quickly becomes complicated when more roads are being crossed. Instead of evaluating the integral for a general case, a quadrature rule will be used. In our simulations, we made use of the trapezoidal rule. Using the limited scenarios we have computed the integral exactly, we will investigate how accurate an approximation using the trapezoidal rule is for a varying number of quadrature points. The only non-trivial case where we have computed the integral (4.38) is the case of two sticks where both have lengths less than or equal to 0.5. Hence, we will use this example and the exact solution (A.26) to evaluate the convergence order of the trapezoidal rule. In Figure A.4 we see that for all choices of the two stick lengths, the trapezoidal rule converges with a rate close to 2.

A.3 Measuring Difference Between Simulations

We will in this section discuss how to compare the difference between two different simulations. As a first step, we assume that two different simulations

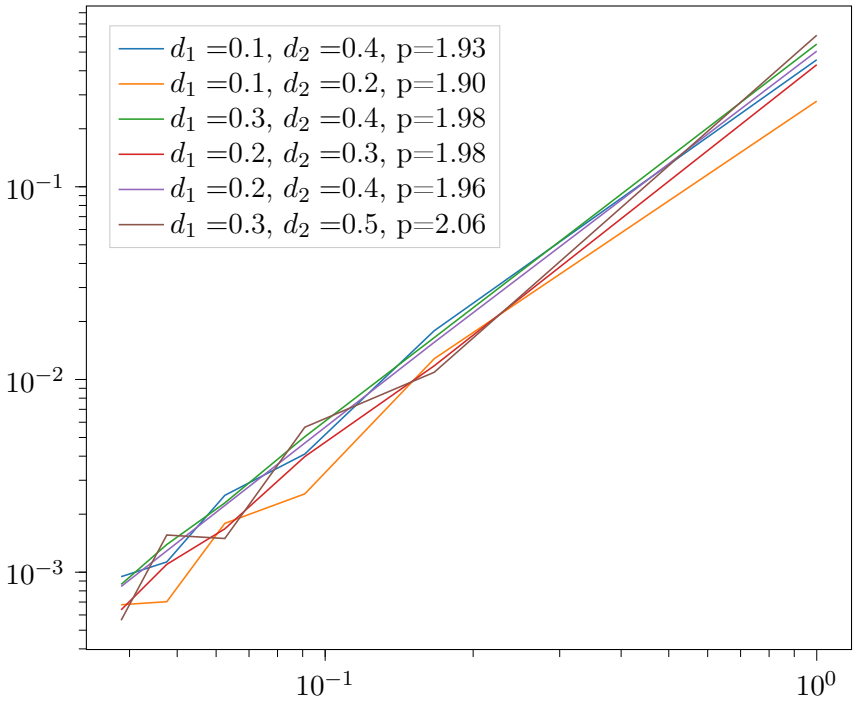


Figure A.4: Convergence of the trapezoidal rule for evaluating the integral (4.38) with two sticks of different lengths for an increasing number of quadrature points.

are performed with the same time steps and the same spatial lengths. We will in the following consider without loss of generality only road l_i. We denote by $\rho^n_{l_i,j}$ and $\tilde{\rho}^n_{l_i,j}$ the density approximations on cell j of road l_i at time t^n from the two simulations. We start by measuring the error at one time of the simulation t^n. To this end, we make use of a piecewise constant reconstruction of the densities on the cells. That is, for each road, at each time t^n we define the piecewise constant function

$$\rho^n_{l_i}(x) = \begin{cases} \rho^n_{l_i,0} & \text{if } x \in [x_{-1/2}, x_{1/2}), \\ \vdots & \\ \rho^n_{l_i,j} & \text{if } x \in [x_{j-1/2}, x_{j+1/2}), \\ \vdots & \end{cases} \tag{A.27}$$

Then, at each time we measure the difference between the simulations as

$$\left\|\rho^n_{l_i} - \tilde{\rho}^n_{l_i}\right\|_{L^1} = \int_{x_{L_{l_i}}}^{x_{R_{l_i}}} |\rho^n_{l_i}(x) - \tilde{\rho}^n_{l_i}(x)| dx = \Delta x \sum_j |\rho^n_{l_i,j} - \tilde{\rho}^n_{l_i,j}|,$$

where $x_{L_{l_i}}$ and $x_{R_{l_i}}$ denote the left and right edges of road l_i, and $\tilde{\rho}^n_{l_i}$ denotes the piecewise reconstruction from the second simulation. Aiming to measure

the difference between the full simulations, we can make use of a linear interpolation between the time steps to construct the function

$$\rho_{l_1}(x,t) = \begin{cases} t \cdot \rho_{l_1}^n(x) + (t^1 - t) \cdot \rho_{l_1}^{n+1}(x) & \text{if } t \in [0, t^1), \\ \vdots, & \\ (t - t^n) \cdot \rho_{l_1}^n(x) + (t^{n+1} - t) \cdot \rho_{l_1}^{n+1}(x) & \text{if } t \in [t^n, t^{n+1}), \\ \vdots & \end{cases} \tag{A.28}$$

with $\rho_{l_1}^n(x)$ being the piecewise reconstruction defined in (A.27). Defining a similar function for the second simulation, $\tilde{\rho}_{l_1}(x,t)$, we measure the differences between the simulations as

$$\begin{aligned} \|\rho_{l_1} - \tilde{\rho}_{l_1}\|_{L^1} &= \int_0^T \int_{x_{L_{l_i}}}^{x_{R_{l_i}}} |\rho_{l_1}(x,t) - \tilde{\rho}_{l_1}(x,t)| dx dt \\ &= \sum_{k=1}^{n-1} \frac{\Delta t^n}{2} \Big(\big(\sum_j \Delta x |\rho_{l_i,j}^k - \tilde{\rho}_{l_i,j}^k|\big) + \big(\sum_j \Delta x |\rho_{l_i,j}^{k+1} - \tilde{\rho}_{l_i,j}^{k+1}|\big) \Big). \end{aligned} \tag{A.29}$$

We note that this measure of the error is only valid when the spatial length Δx and all of the time steps Δt^n are the same for the two simulations. However, as we stated in the beginning of this section, we want to measure the difference using different spatial lengths. Assume now that the two simulations are performed with two different spatial lengths $\Delta x = 2 \cdot \Delta \tilde{x}$. Assume further that time step t^n is a part of both simulations. Then, we want to measure the error at time t^n. Following the same idea as before, we can make use of a piecewise constant reconstruction. As before we measure the difference as

$$\big\|\rho_{l_i}^n - \tilde{\rho}_{l_i}^n\big\|_{L^1} = \int_{x_{L_{l_i}}}^{x_{R_{l_i}}} |\rho_{l_i}^n(x) - \tilde{\rho}_{l_i}^n(x)| dx = \Delta \tilde{x} \sum_j |\rho_{l_i,k(j)}^n - \tilde{\rho}_{l_i,j}^n|, \tag{A.30}$$

where $k(j)$ is defined as the mapping

$$k(j) = (j - j \bmod 2)/2.$$

We could also have different ratios between the spatial lengths $\Delta x = r \cdot \Delta \tilde{x}$ with $r \in \mathbb{N}$, leading to the same expression for the difference as (A.30), but with a new mapping function

$$k(j; r) = (j - j \bmod r)/r. \tag{A.31}$$

The error measure (A.30) is only valid when time step t^n is a part of both simulations. Since we do not fix the time step Δt at the start of the simulation, but rather choose the next time step using a CFL condition, also the time steps of different simulations might differ. We now consider the case with two simulations with $\Delta x = r \cdot \Delta \tilde{x}$ and the sets of evaluation times $\{t^n\}_n$ and $\{t^{\tilde{m}}\}_m$,

where the time steps Δt^n and $\Delta\tilde{t}^m$ may all be different. Also the sizes of the sets Δt^n and $\Delta\tilde{t}^m$ may be different. We start by creating the interpolated functions $\rho_{l_1}(x,t)$ and $\tilde{\rho}_{l_1}(x,t)$. We want to write the difference between these functions as a sum of the difference at a finite number of evaluation point. Therefore, we will evaluate the function $\rho_{l_1}(x,t)$ at the times of the second iteration $\{\tilde{t^n}\}_n$. We denote the evaluation of $\rho_{l_1}(x,t)$ at cell j at time $t^{\tilde{m}}$ by $\bar{\rho}^m_{l_i,j}$. This evaluation point can be computed as

$$\bar{\rho}^m_{l_i,j} = (t^{\tilde{m}} - t^n) \cdot \rho^n_{l_i,j} + (t^{n+1} - \tilde{t^n}) \cdot \rho^{n+1}_{l_i,j},$$

where t^n and t^{n+1} are the two time points of the first simulation satisfying $t^n \leq t^{\tilde{m}} \leq t^{n+1}$. Finally, we compute the difference between two simulations with different time steps and spatial lengths as

$$\|\rho_{l_1} - \tilde{\rho}_{l_1}\|_{L^1} = \sum_{m=1}^{n-1} \frac{\Delta\tilde{t}^n}{2}\Big(\big(\sum_j \Delta\tilde{x} |\bar{\rho}^m_{l_i,k(j;r)} - \tilde{\rho}^m_{l_i,j}|\big) + \big(\sum_j \Delta\tilde{x} |\bar{\rho}^{m+1}_{l_i,k(j)} - \tilde{\rho}^{m+1}_{l_i,j}|\big)\Big), \quad \text{(A.32)}$$

where again $k(j)$ is the mapping function defined in (A.31). We note that (A.32) only measures the differences in simulations for one road. When measuring the full difference, we measure the difference for each of the roads by using (A.32) but replacing l_i with each of the roads, before taking the sum.

Appendix B

Specifications of Networks Used for Optimization

In this section we list all of the details of the networks used for the numerical experiments in Chapter 7. We have decided to move most of these details in the appendix to improve the flow of the chapter.

B.1 Network Used for Checking Convergence of the Model

In this section, we include some tables specifying the configuration of the network used to check the convergence of the model.

Table B.1: Road configuration of network used in convergence test

Road	L [m]	ρ_{init}	$\rho_{\max}$
l_1	125	0.3	1
l_2	125	0.3	1
l_3	125	0.3	1
l_4	50	0.3	1
l_5	50	0.3	1
l_6	50	0.3	1
r_1	125	0.3	1
r_2	125	0.3	1
r_3	125	0.3	1
r_4	50	0.3	1
r_5	50	0.3	1
r_6	50	0.3	1
I_1	25	0.3	1
I_2	25	0.3	1
I_3	25	0.3	1

Table B.2: Junction configuration of network used in convergence test

Id	**In**	**Out**	**Distribution**	**Priority**	**Crossing**
J_1	l_1, r_2, r_4	r_1, l_2, l_4	$\begin{bmatrix} 0.0 & 0.6 & 0.4 \\ 0.6 & 0.0 & 0.4 \\ 0.5 & 0.5 & 0.0 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 2 & 0 \end{bmatrix}$	$\begin{bmatrix} \cdot & \cdot & (1,0) \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix}$
J_2	l_2, r_3, r_5	r_2, l_3, l_5	$\begin{bmatrix} 0.0 & 0.6 & 0.4 \\ 0.6 & 0.0 & 0.4 \\ 0.5 & 0.5 & 0.0 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 2 & 0 \end{bmatrix}$	$\begin{bmatrix} \cdot & \cdot & (1,0) \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix}$
J_3	l_3, r_1, r_6	r_3, l_1, l_6	$\begin{bmatrix} 0.0 & 0.6 & 0.4 \\ 0.6 & 0.0 & 0.4 \\ 0.5 & 0.5 & 0.0 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 2 & 0 \end{bmatrix}$	$\begin{bmatrix} \cdot & \cdot & (1,0) \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix}$

Table B.3: Coupled Traffic Light configuration of network used in convergence test

Id	**A In**	**A Out**	**B In**	**B Out**	**Starting State**	**Cycle**
T_1^c	l_1, r_2	r_1, l_2, l_4	r_4	r_1, l_2	A	60,80
T_2^c	l_2, r_3	r_2, l_3, l_5	r_5	r_2, l_3	A	50,50
T_3^c	l_3, r_1	r_3, l_1, l_6	r_6	r_3, l_1	A	70,70

B.2 Specifications for Larger Network

In this section, we include some tables specifying the configuration of the larger network where the optimal control problem was considered.

Table B.4: Road configuration of larger network

Road	L [m]	ρ_{init}	$\rho_{\max}$
l_1	50	0.4	1
l_2	50	0.4	1
l_3	50	0.4	1
l_4	50	0.4	1
l_5	50	0.4	1
l_6	50	0.4	1
l_7	50	0.4	1
r_1	50	0.4	1
r_2	50	0.4	1
r_3	50	0.4	1
r_4	50	0.4	1
r_6	50	0.4	1
r_7	50	0.4	1
I_1	50	0.4	1
I_2	50	0.4	1
I_3	50	0.4	1

Table B.5: Junction configuration of larger network

Id	**In**	**Out**	**Distribution**	**Priority**	**Crossing**
J_1	l_1, l_2, r_3	r_1, r_2, l_3	$\begin{bmatrix} 0.0 & 0.4 & 0.6 \\ 0.5 & 0.0 & 0.5 \\ 0.6 & 0.4 & 0.0 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 2 \\ 2 & 2 & 0 \end{bmatrix}$	$\begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix}$
J_2	l_3, r_4	r_3, l_4, l_5	$\begin{bmatrix} 0.0 & 0.8 & 0.2 \\ 0.7 & 0.0 & 0.3 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} \cdot & \cdot & (1,0) \\ \cdot & \cdot & \cdot \end{bmatrix}$
J_3	l_5, l_6, r_7	r_6, l_7	$\begin{bmatrix} 0.5 & 0.5 \\ 0.0 & 1.0 \\ 1.0 & 0.0 \end{bmatrix}$	$\begin{bmatrix} 2 & 2 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$	$\begin{bmatrix} (1,1) & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \end{bmatrix}$

Table B.6: Coupled traffic light configuration of larger network

Id	**A In**	**A Out**	**B In**	**B Out**
T_1^c	l_1, r_3	r_1, r_2, l_3	l_2	r_1, r_2, l_3

B.3 Specifications for Kvadraturen Network

In this section, we include some tables specifying the configuration of the Kvadraturen network.

Table B.7: Road configuration of Kvadraturen part 1

Road	L [m]	ρ_{init}	ρ_{max}	v_{min} [km/h]	v_{max} [km/h]
l_1	150	0.2	1	20	50
l_2	50	0.2	1	20	50
l_3	50	0.2	1	20	50
l_4	50	0.2	1	20	50
l_5	50	0.2	1	20	50
l_6	50	0.2	1	20	50
l_7	50	0.2	1	20	50
l_8	300	0.2	1	20	50
l_{10}	300	0.2	1	20	50
l_{11}	200	0.2	1	20	50
l_{12}	100	0.2	1	20	50
l_{13}	300	0.2	1	20	50
l_{14}	100	0.2	1	20	50
l_{15}	50	0.2	1	20	50
l_{16}	100	0.2	1	20	50
l_{17}	50	0.2	1	20	50
l_{18}	150	0.2	1	20	50
l_{19}	50	0.2	1	20	50
l_{20}	50	0.2	1	20	50
l_{21}	50	0.2	1	20	50
l_{22}	100	0.7	1	70	80
l_{23}	200	0.7	1	70	80
l_{24}	250	0.7	1	70	80
l_{25}	150	0.7	1	70	80
l_{26}	100	0.5	1	40	60
l_{27}	100	0.5	1	40	60
l_{28}	100	0.5	1	40	60
l_{29}	100	0.5	1	40	60

Table B.8: Road configuration of Kvadraturen part 2

Road	L [m]	ρ_{init}	$\rho_{\max}$	$v_{\min}$ [km/h]	$v_{\max}$ [km/h]
r_1	150	0.2	1	20	50
r_2	50	0.2	1	20	50
r_3	50	0.2	1	20	50
r_4	50	0.2	1	20	50
r_5	50	0.2	1	20	50
r_6	50	0.2	1	20	50
r_9	300	0.2	1	20	50
r_{10}	300	0.2	1	20	50
r_{11}	200	0.2	1	20	50
r_{12}	100	0.2	1	20	50
r_{13}	300	0.2	1	20	50
r_{14}	100	0.2	1	20	50
r_{15}	50	0.2	1	20	50
r_{16}	100	0.2	1	20	50
r_{17}	50	0.2	1	20	50
r_{18}	150	0.2	1	20	50
r_{19}	50	0.2	1	20	50
r_{20}	50	0.2	1	20	50
r_{21}	50	0.2	1	20	50
r_{22}	100	0.7	1	70	80
r_{23}	200	0.7	1	70	80
r_{24}	250	0.7	1	70	80
r_{25}	150	0.7	1	70	80
r_{26}	100	0.5	1	40	60
r_{27}	100	0.5	1	40	60
r_{28}	100	0.5	1	40	60
r_{29}	100	0.5	1	40	60

Table B.9: Roundabout road configuration of Kvadraturen

Road	L [m]	ρ_{init}	$\rho_{\max}$	$v_{\min}$ [km/h]	$v_{\max}$ [km/h]
I_1	50	0.2	1	20	50
I_2	50	0.2	1	20	50
I_3	50	0.2	1	20	50
I_4	50	0.2	1	20	50
I_5	50	0.2	1	20	50
I_6	50	0.2	1	20	50
I_7	50	0.2	1	20	50

Table B.10: Junction configuration of Kvadraturen part 1.1

Id	In	Out
J_1	l_1, r_2, l_7	l_2, r_1, l_8
J_2	l_2, r_3	l_3, r_2
J_3	l_3, r_4	l_4, r_3
J_4	l_4, r_5	l_5, r_4
J_5	l_5, r_6, r_9	l_6, r_5
J_6	l_6, r_{11}	l_{11}, r_6
J_8	l_8, l_{16}, r_{17}	l_{17}, r_{16}
J_{10}	l_{10}, r_{21}	l_{21}, r_{10}
J_{11}	l_{11}, r_{12}	l_{12}, r_{11}

Table B.11: Junction configuration of Kvadraturen part 1.2

Id	Distribution	Priority	Crossing
J_1	$\begin{bmatrix} 0.7 & 0.0 & 0.3 \\ 0.7 & 0.0 & 0.3 \\ 0.3 & 0.3 & 0.4 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 1 \\ 2 & 2 & 3 \end{bmatrix}$	$\begin{bmatrix} \cdot & \cdot & (1,1) \\ \cdot & \cdot & \cdot \\ \cdot & (0,0),(0,2) & (0,0),(1,1) \end{bmatrix}$
J_2	$\begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{bmatrix}$	.	.
J_3	$\begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{bmatrix}$	.	.
J_4	$\begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{bmatrix}$	.	.
J_5	$\begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \\ 0.3 & 0.7 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 2 & 2 \end{bmatrix}$	$\begin{bmatrix} \cdot & \cdot \\ (1,1) & \cdot \end{bmatrix}$
J_6	$\begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{bmatrix}$	.	.
J_8	$\begin{bmatrix} 0.5 & 0.5 \\ 1.0 & 0.0 \\ 0.0 & 1.0 \end{bmatrix}$	$\begin{bmatrix} 2 & 2 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} \cdot & (1,0) \\ \cdot & \cdot \\ \cdot & \cdot \end{bmatrix}$
J_{10}	$\begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{bmatrix}$	.	.
J_{11}	$\begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{bmatrix}$	.	.

Table B.12: Junction configuration of Kvadraturen part 2.1

Id	In	Out
J_{12}	$l_{12}, r_{13}, l_{19}, r_{20}$	$l_{13}, r_{12}, l_{20}, r_{19}$
J_{13}	l_{13}, r_{14}, l_{21}	l_{14}, r_{13}, r_{21}
J_{15}	l_{15}, r_{16}	l_{16}, r_{15}
J_{17}	l_{17}, r_{18}	l_{18}, r_{17}
J_{18}	l_{18}, r_{19}, r_{10}	$r_9, l_{19}, r_{18}, l_{10}$
J_{22}	l_{22}, r_{23}, r_{26}	r_{22}, l_{23}, r_{26}
J_{23}	l_{23}, r_{24}, l_{27}	r_{23}, l_{24}, r_{27}
J_{24}	l_{24}, l_{29}	l_{25}, r_{29}
J_{25}	r_{25}, r_{28}	r_{24}, l_{28}

Table B.13: Junction configuration of Kvadraturen part 2.2

Id	Distribution	Priority
J_{12}	$\begin{bmatrix} 0.7 & 0.0 & 0.15 & 0.15 \\ 0.0 & 0.7 & 0.15 & 0.15 \\ 0.15 & 0.15 & 0.7 & 0.0 \\ 0.15 & 0.15 & 0.0 & 0.7 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 1 & 3 \\ 0 & 1 & 3 & 1 \\ 2 & 3 & 2 & 0 \\ 3 & 2 & 0 & 2 \end{bmatrix}$
J_{13}	$\begin{bmatrix} 0.7 & 0.0 & 0.3 \\ 0.0 & 0.7 & 0.4 \\ 0.5 & 0.5 & 0.0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 1 \\ 2 & 2 & 0 \end{bmatrix}$
J_{15}	$\begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{bmatrix}$	.
J_{17}	$\begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{bmatrix}$	.
J_{18}	$\begin{bmatrix} 0.1 & 0.8 & 0.0 & 0.1 \\ 0.1 & 0.0 & 0.8 & 0.1 \\ 0.5 & 0.25 & 0.25 & 0.0 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 0 & 2 \\ 2 & 0 & 1 & 1 \\ 3 & 2 & 2 & 0 \end{bmatrix}$
J_{22}	$\begin{bmatrix} 0.0 & 0.8 & 0.2 \\ 0.9 & 0.0 & 0.1 \\ 0.7 & 0.3 & 0.0 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 2 \\ 2 & 2 & 0 \end{bmatrix}$
J_{23}	$\begin{bmatrix} 0.0 & 0.9 & 0.1 \\ 0.8 & 0.0 & 0.2 \\ 0.7 & 0.3 & 0.0 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 2 \\ 2 & 2 & 0 \end{bmatrix}$
J_{24}	$\begin{bmatrix} 0.8 & 0.2 \\ 1.0 & 0.0 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 \\ 2 & 0 \end{bmatrix}$
J_{25}	$\begin{bmatrix} 0.8 & 0.2 \\ 1.0 & 0.0 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 \\ 2 & 0 \end{bmatrix}$

Table B.14: Junction configuration of Kvadraturen part 2.3

Id	Crossing
J_{12}	$\begin{bmatrix} \cdot & \cdot & \cdot & (1,1),(2,2),(2,0) \\ \cdot & \cdot & (0,0),(3,1),(3,3) & \cdot \\ \cdot & (1,1),(3,3) & (0,0),(1,1) & \cdot \\ (0,0) & \cdot & \cdot & (0,0),(1,1) \end{bmatrix}$
J_{13}	$\begin{bmatrix} \cdot & \cdot & (1,1) \\ \cdot & \cdot & \cdot \\ (1,1) & \cdot & \cdot \end{bmatrix}$
J_{15}	$\cdot$
J_{17}	$\cdot$
J_{18}	$\begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ (0,1) & \cdot & \cdot & \cdot \\ (0,1),(1,2) & (1,2) & \cdot & \cdot \end{bmatrix}$
J_{22}	$\cdot$
J_{23}	$\cdot$
J_{24}	$\cdot$
J_{25}	$\cdot$

Table B.15: Traffic light configuration of Kvadraturen

Id	In	Out	s_{start}
T_2	l_2, r_3	l_3, r_2	0
T_3	l_3, r_4	l_4, r_3	0
T_4	l_4, r_5	l_5, r_4	0
T_6	l_6, r_{11}	l_{11}, r_6	0
T_{11}	l_{11}, r_{12}	l_{12}, r_{11}	0
T_{15}	l_{15}, r_{16}	l_{16}, r_{15}	0
T_{17}	l_{17}, r_{18}	l_{18}, r_{17}	0

Table B.16: Coupled Traffic light configuration of Kvadraturen

Id	A In	A Out	B In	B Out	s_{start}
T_1^c	l_1, r_2	l_2, r_1, l_8	l_7	l_2, r_1, l_8	A
T_5^c	l_5, r_6	l_6, r_5	r_9	l_6, r_5	A
T_8^c	l_8	l_{17}, r_{16}	l_{16}, r_{17}	l_{17}, r_{16}	A
T_{12}^c	l_{12}, r_{13}	$l_{13}, r_{12}, l_{20}, r_{19}$	l_{19}, r_{20}	$l_{13}, r_{12}, l_{20}, r_{19}$	A
T_{13}^c	l_{13}, r_{14}	l_{14}, r_{13}, r_{21}	l_{21}	l_{14}, r_{13}, r_{21}	A
T_{18}^c	l_{18}, r_{19}	$r_9, l_{19}, r_{18}, l_{10}$	r_{10}	$r_9, l_{19}, r_{18}, l_{10}$	A

Table B.17: Bus lines of Kvadraturen

Id	**Roads**
A	$I_3, I_4, l_1, l_8, l_{17}, l_{18}, l_{10}, l_{21}, l_{14}$
B	$r_{14}, r_{21}, r_{10}, r_9, r_5, r_4, r_3, r_2, r_1, I_1, I_2$
C	$l_{22}, l_{26}, I_4, l_1, r_8, l_{17}, l_{18}, l_{10}, l_{21}, l_{14}$
D	$r_{14}, r_{21}, r_{10}, r_9, r_5, r_4, r_3, r_2, r_1, I_1, I_2, I_3, r_{26}, r_{22}$
E	$l_7, l_2, l_3, l_4, l_5, l_6, l_{11}, l_{12}, r_{19}, r_{18}, r_{17}, r_{16}, r_{15}, I_5, l_{29}, l_{25}$
F	$r_{25}, l_{28}, I_7, l_{15}, l_{16}, l_{17}, l_{18}, r_9, r_5, r_4, r_3, r_2, r_1, I_1, I_2, I_3, r_{26}, r_{22}$

Table B.18: Busses of Kvadraturen with stops and starting times

Id	**Stops**	**Times**	t_{start}
A_1	(l_8, 130), (l_{18}, 35), (l_{10}, 30), (l_{10}, 260)	3, 110, 130, 230	0
A_2	(l_8, 130), (l_{18}, 35), (l_{10}, 30), (l_{10}, 260)	600, 710, 730, 830	600
B_1	(r_{10}, 45), (r_9, 80), (r_9, 235), (r_1, 25)	4, 130, 190, 250	0
B_2	(r_{10}, 45), (r_9, 80), (r_9, 235), (r_1, 25)	700, 830, 870, 900	650
C_1	(l_8, 90), (l_{18}, 130)	50, 100	0
C_2	(l_8, 90), (l_{18}, 130)	650, 700	600
D_1	(r_9, 50), (r_9, 180)	50, 100	0
D_1	(r_9, 50), (r_9, 180)	750, 800	700
E_1	(r_{18}, 20), (r_{16}, 70)	50, 100	0
E_1	(r_{18}, 20), (r_{16}, 70)	680, 730	630
F_1	(l_{16}, 75), (r_9, 50), (r_9, 180)	720, 770, 820	670

Appendix C

Checking Properties of Objective Functionals

We have created a large number of plots for the purposes of checking how smooth the objective functionals we are trying to optimize with respect to are. To avoid unnecessary repetition of similar plots in the main text, we include these plots here. For all sections in this appendix, the objective values are represented by a blue dots. At each of the evaluation points, also the partial derivatives of the objective functions have been computed. We represent the tangent line at each evaluation point by a dashed red line where the slope of the line is equal to the partial derivative computed using automatic differentiation.

For all networks and objective functions being considered, the speed limits are varied around 40km/h, and the settings of the traffic lights are varied around 60 seconds. When one parameter is being changed, all other parameters are fixed at their centers (40 km/h and 60 seconds).

C.1 Single Junction Network

In this section we include plots showing the smoothness and the associated gradient for different objective functional for a single junction.

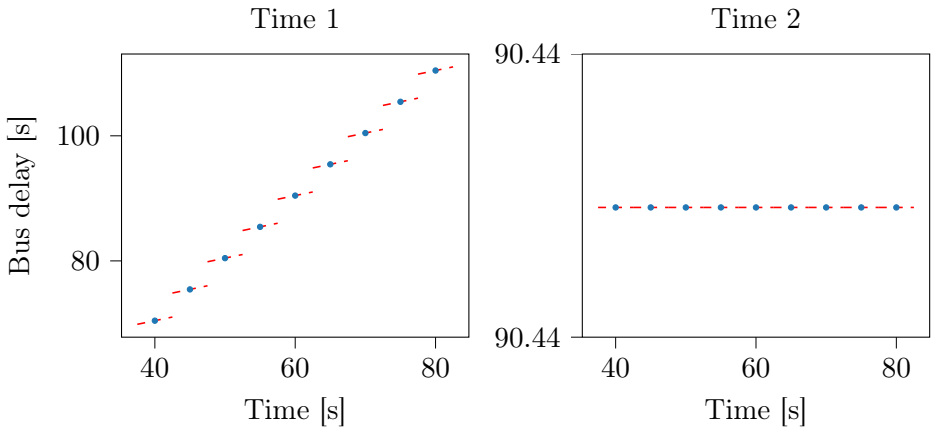


Figure C.1: Bus delays through a single junction for different cycle times of the traffic light. Increment of 5 seconds.

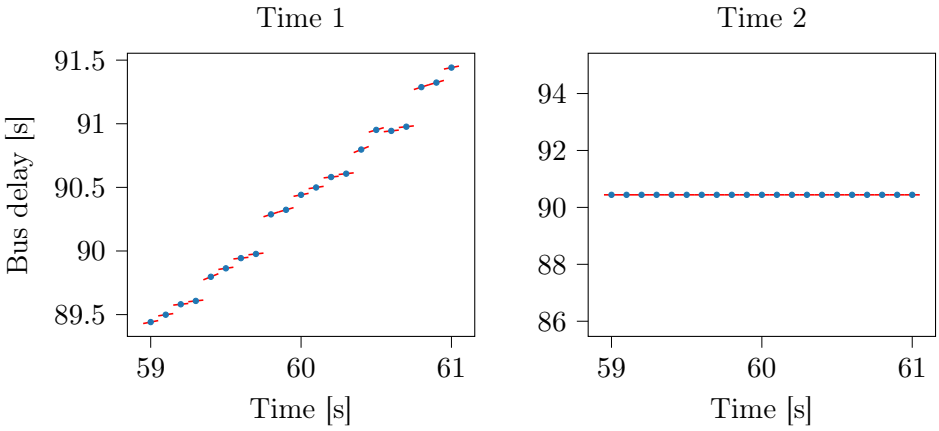


Figure C.2: Bus delays through a single junction for different cycle times of the traffic light. Increment of 0.1 seconds.

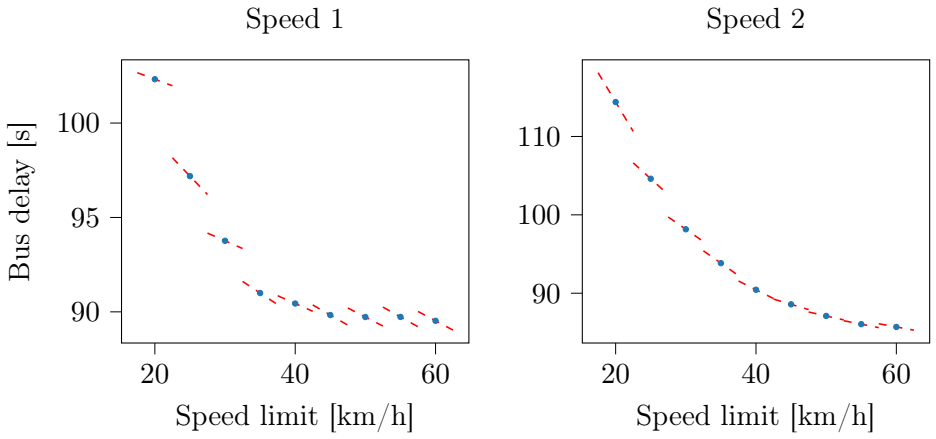


Figure C.3: Bus delays through a single junction for different speed limits. Increment of 5 km/h.

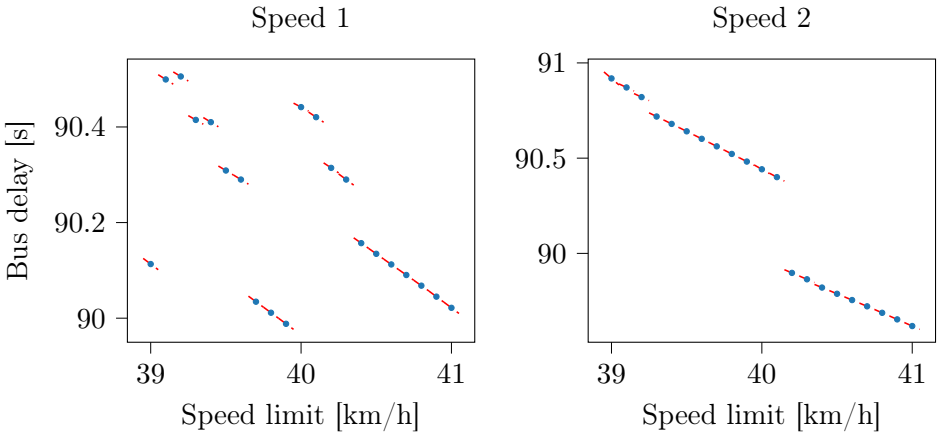


Figure C.4: Bus delays through a single junction for different speed limits. Increment of 0.1 km/h.

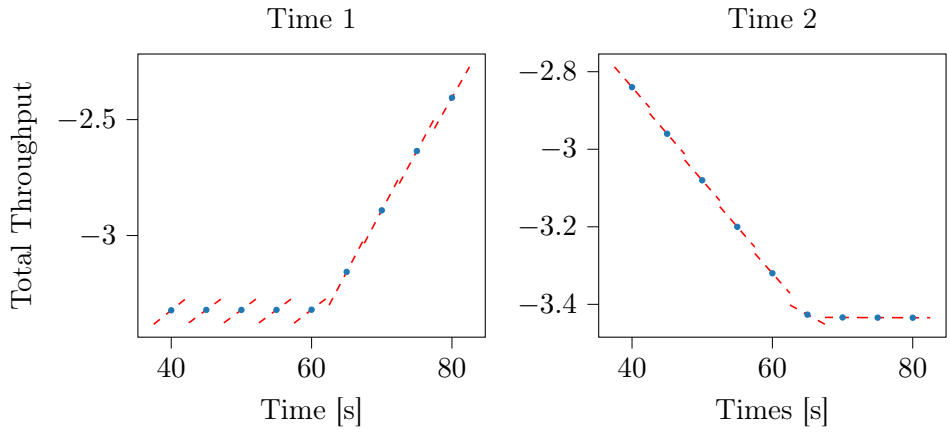


Figure C.5: Total throughput from a network with a single junction for different cycle times of the traffic light. Increment of 5 seconds.

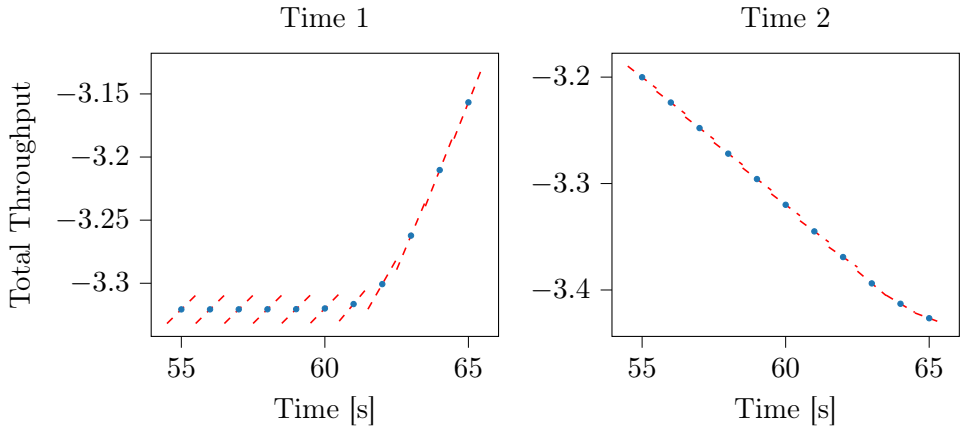


Figure C.6: Total throughput from a network with a single junction for different cycle times of the traffic light. Increment of 0.1 seconds.

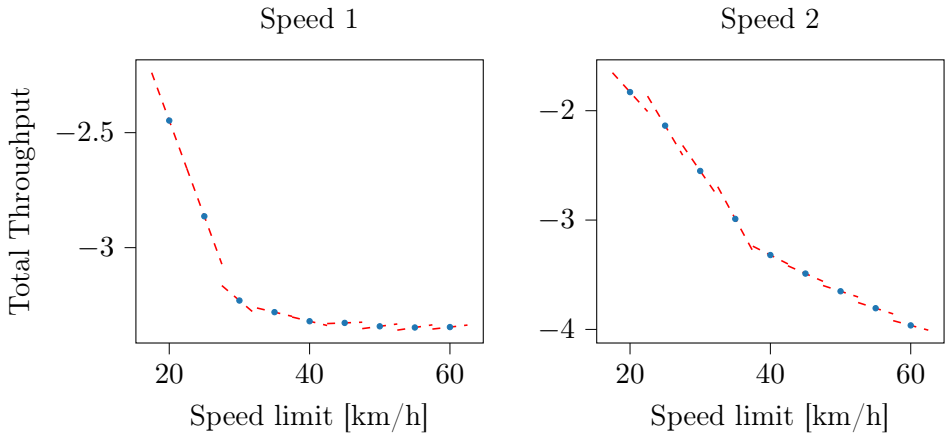


Figure C.7: Total throughput from a network with a single junction for different speed limits. Increment of 5 km/h.

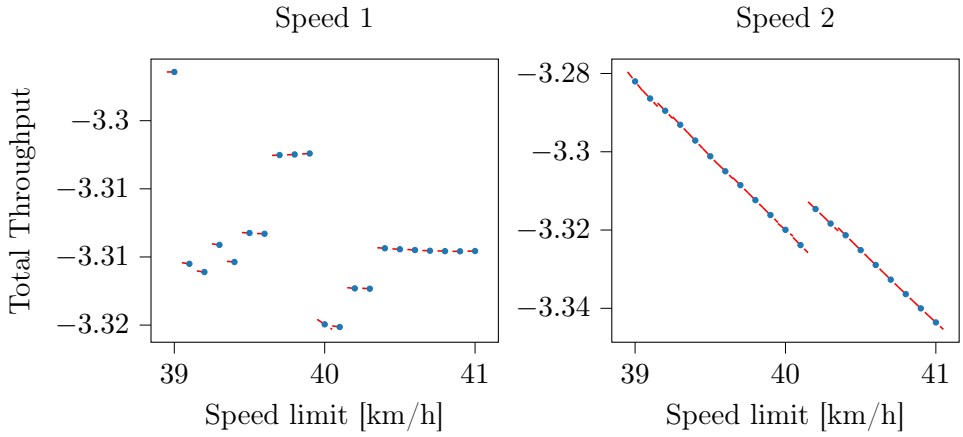


Figure C.8: Total throughput from a network with a single junction for different speed limits. Increment of 0.1 km/h.

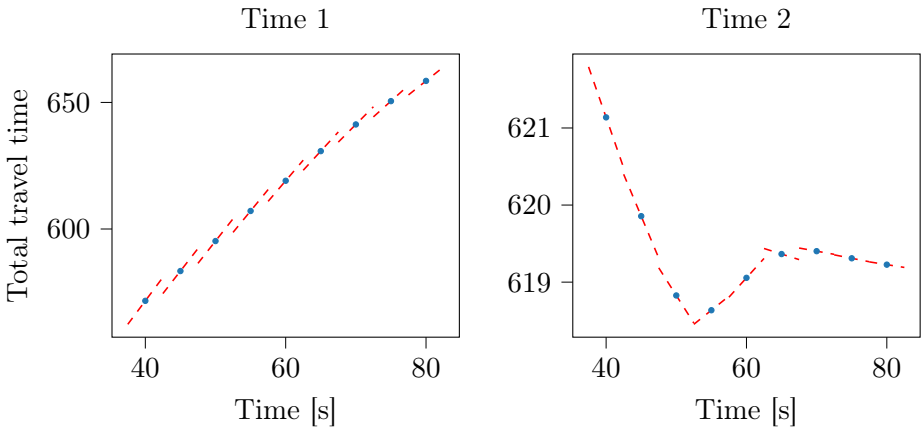


Figure C.9: Total travel time in a network with a single junction for different cycle times of the traffic light. Increment of 5 seconds.

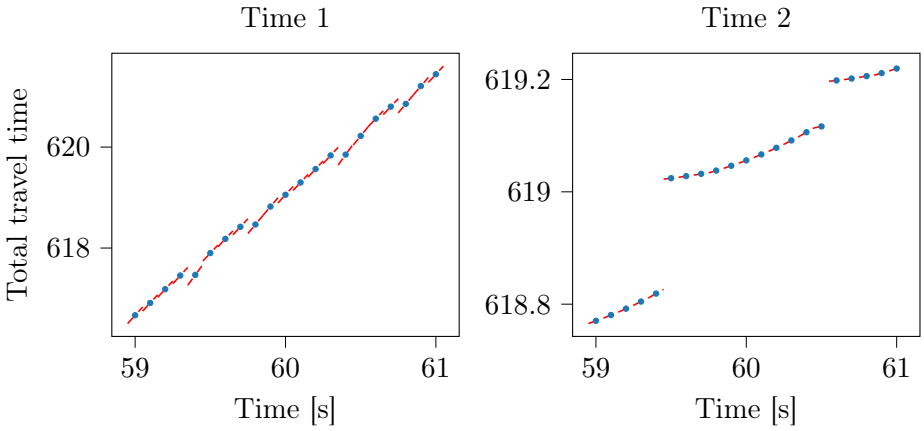


Figure C.10: Total travel time in a network with a single junction for different cycle times of the traffic light. Increment of 0.1 seconds.

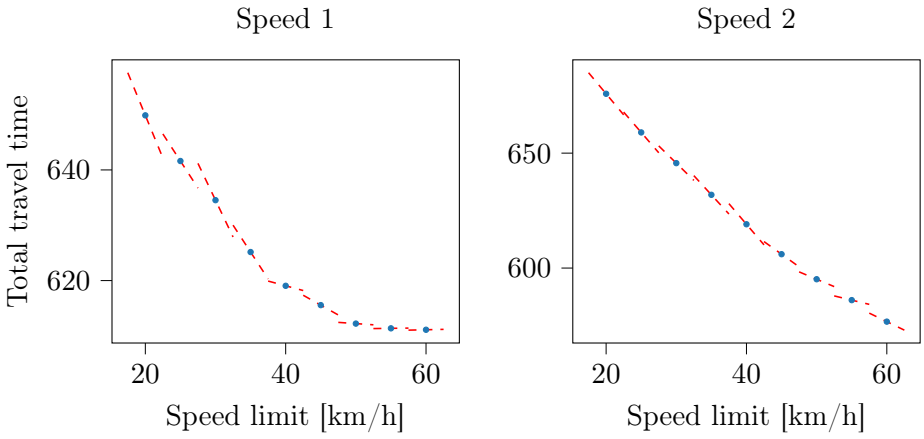


Figure C.11: Total travel time in a network with a single junction for different speed limits. Increment of 5 km/h.

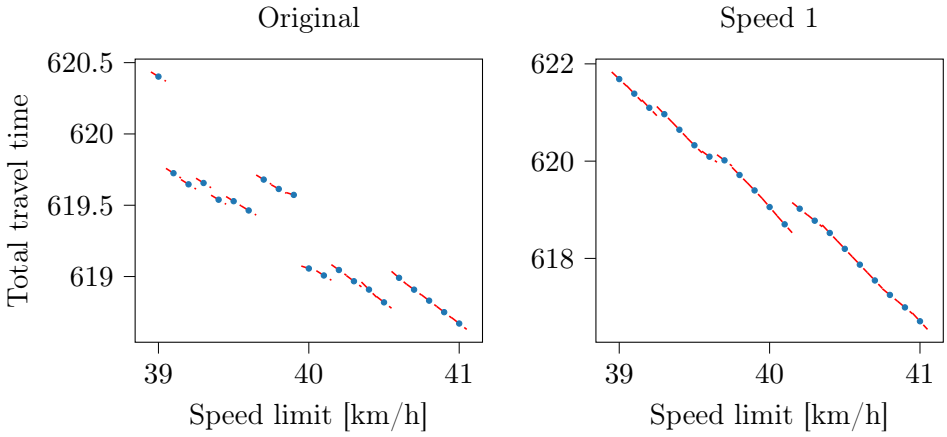


Figure C.12: Total travel time in a network with a single junction for different speed limits. Increment of 0.1 km/h.

C.2 2-to-2 Junction Network

In this section we include plots showing the smoothness and the associated gradient for different objective functional for a traffic-flow network consisting of four roads connected in a 2-to-2 junction.

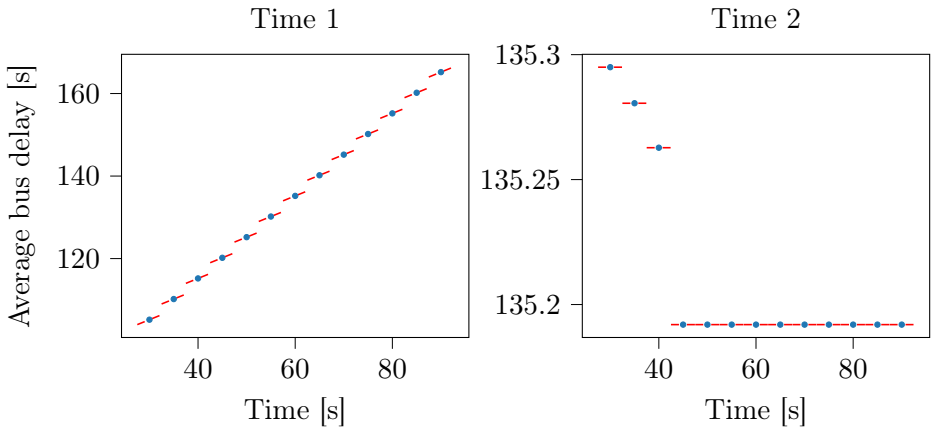


Figure C.13: Bus delays through two-to-two junction for different cycle times of the traffic light. Increment of 5 seconds.

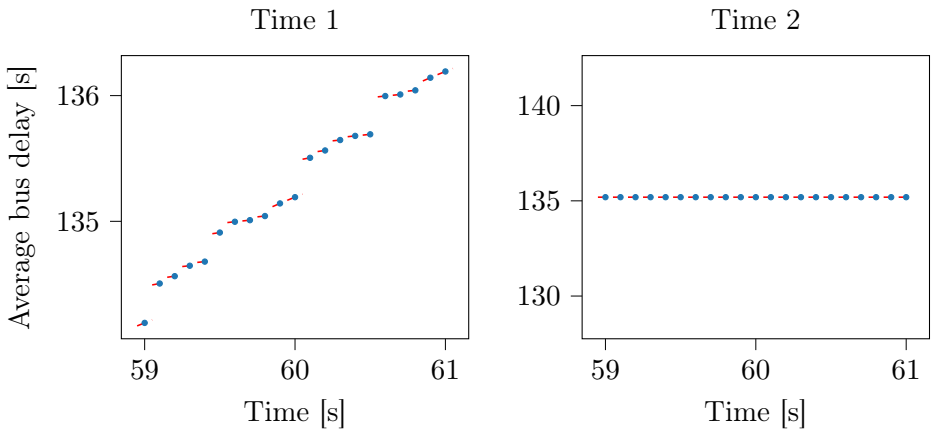


Figure C.14: Bus delays through two-to-two junction for different cycle times of the traffic light. Increment of 0.1 seconds.

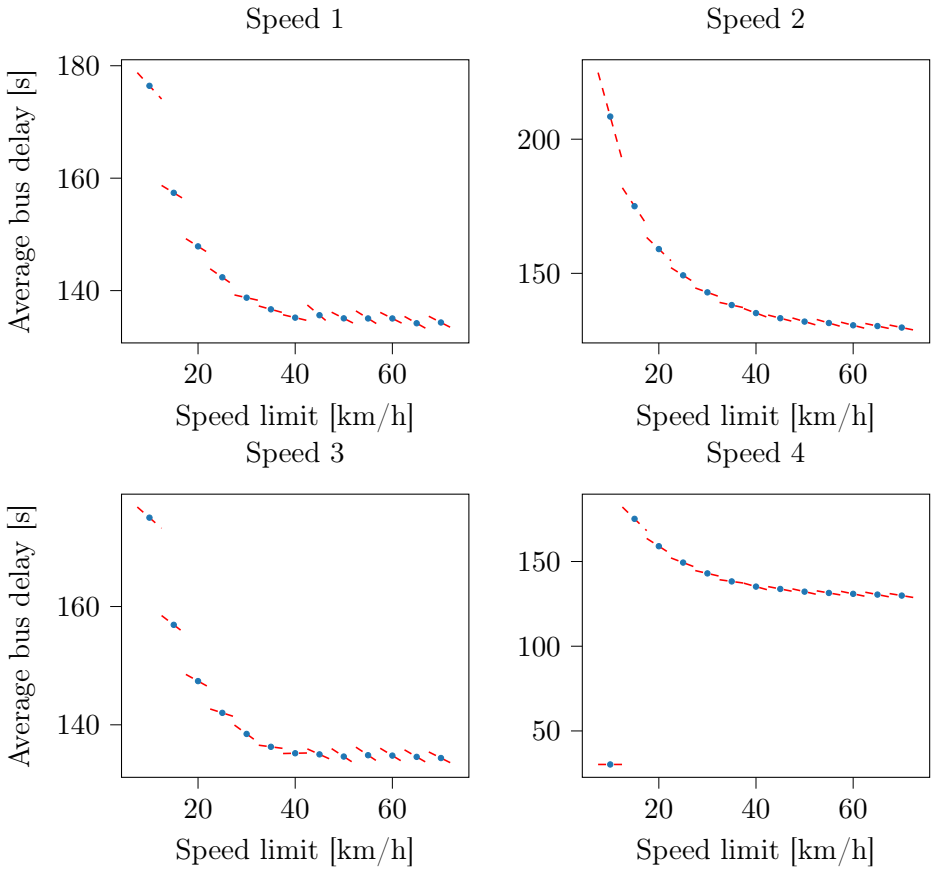


Figure C.15: Bus delays through a two-to-two junction for different speed limits. Increment of 5 km/h.

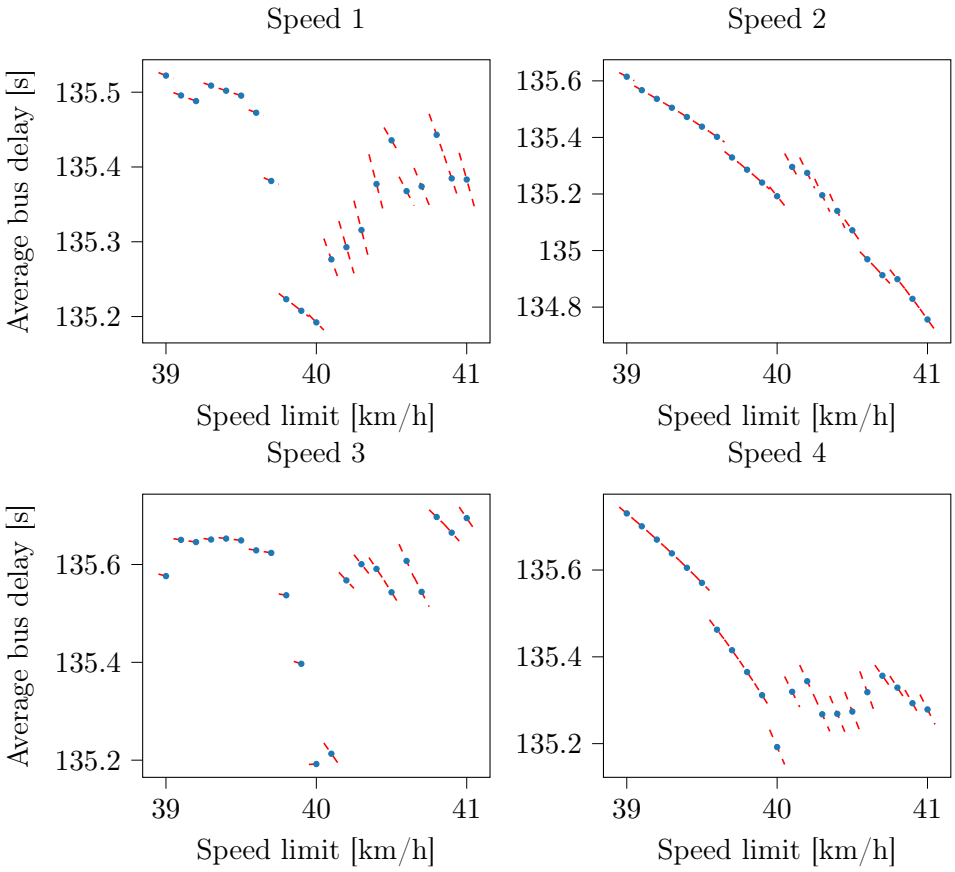


Figure C.16: Bus delays through a two-to-two junction for different speed limits. Increment of 0.1 km/h.

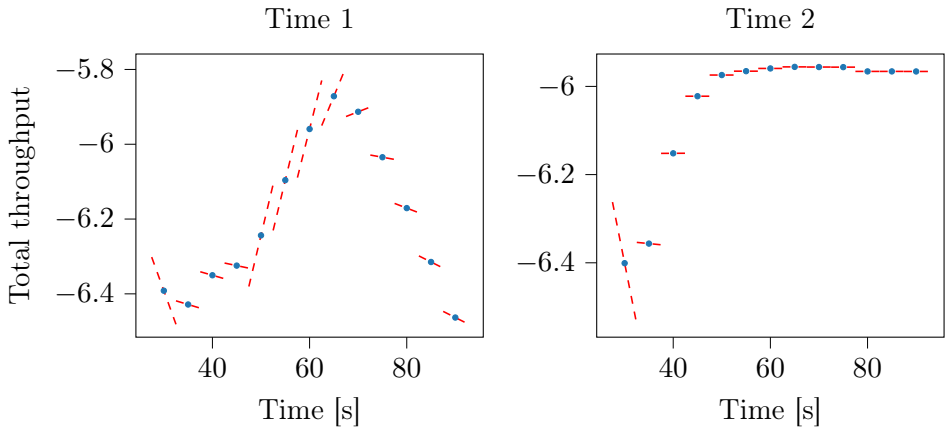


Figure C.17: Total throughput from a network with a two-to-two junction for different cycle times of the traffic light. Increment of 5 seconds.

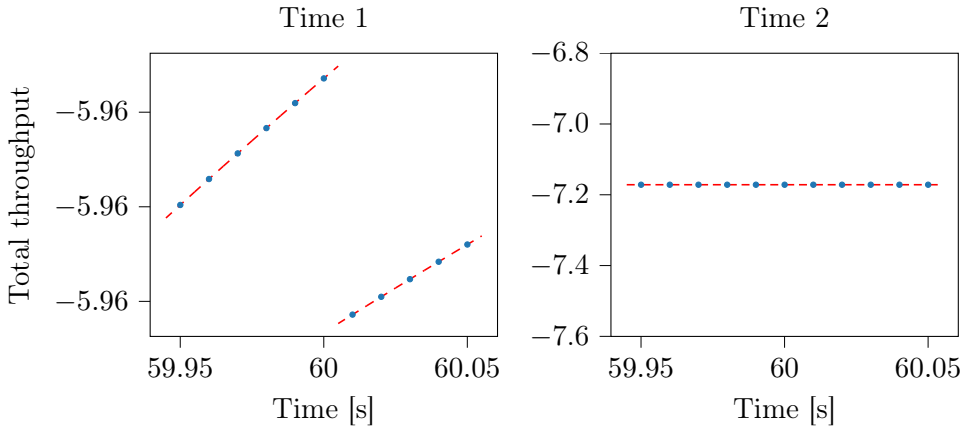


Figure C.18: Total throughput from a network with a two-to-two junction for different cycle times of the traffic light. Increment of 0.01 seconds.

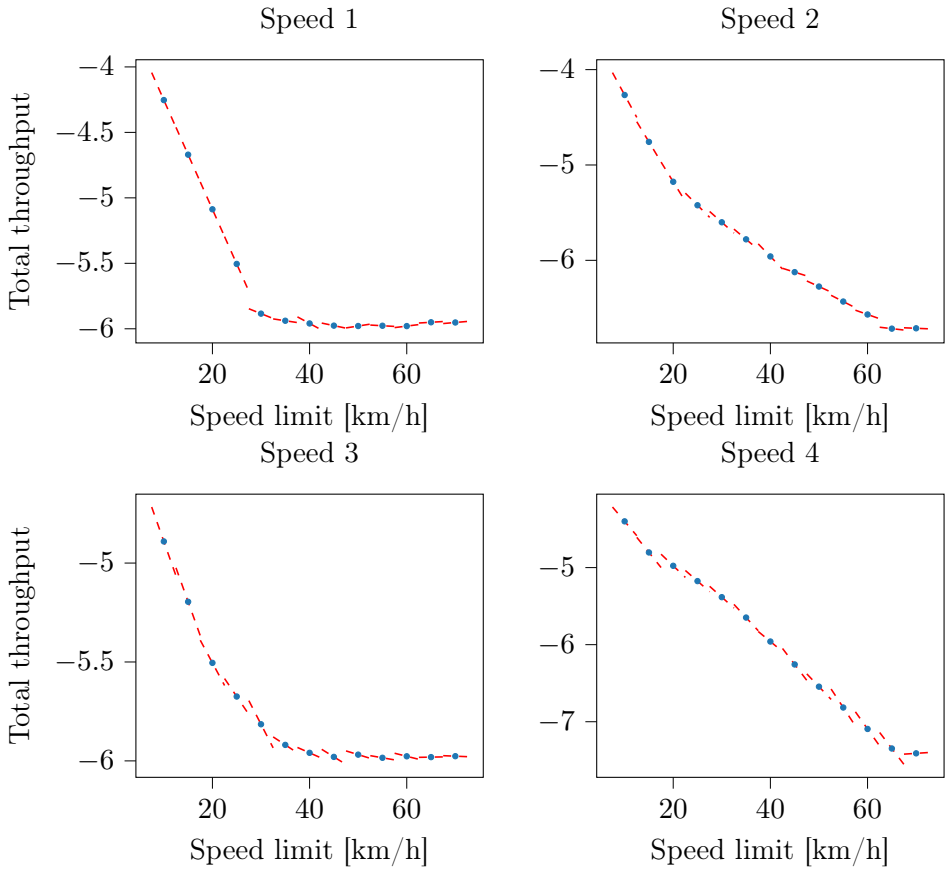


Figure C.19: Total throughput from a network with a two-to-two junction for different speed limits. Increment of 5 km/h.

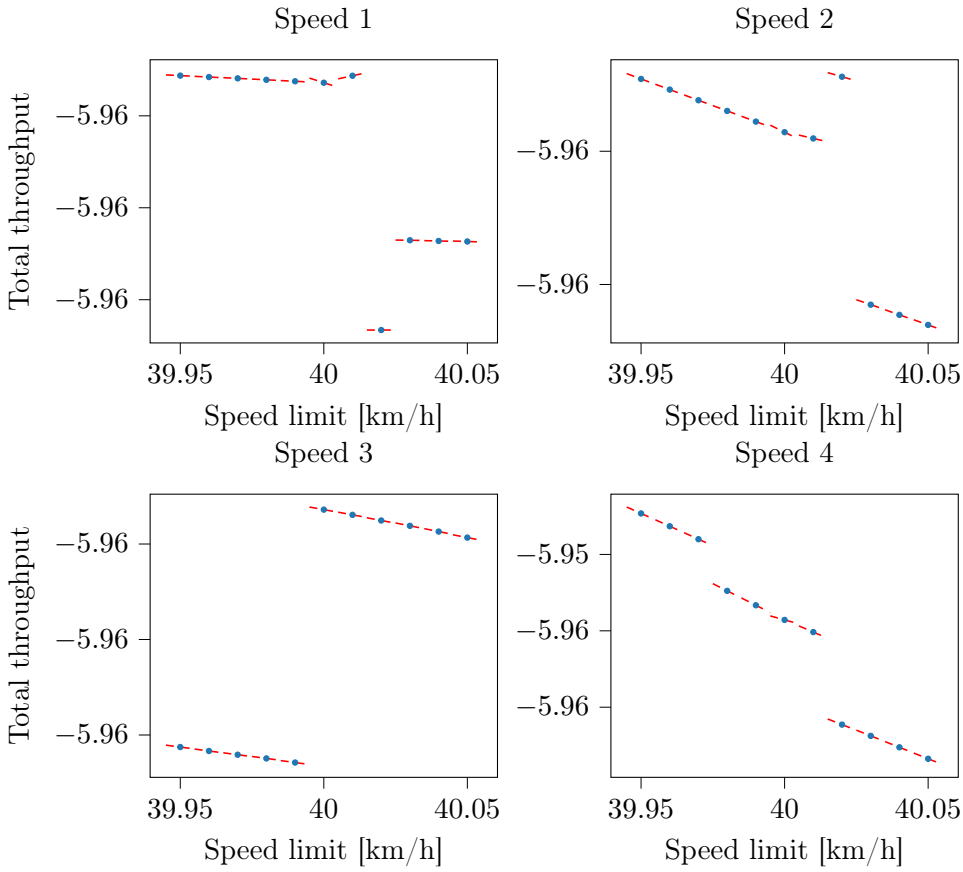


Figure C.20: Total throughput from a network with a two-to-two junction for different speed limits. Increment of 0.01 km/h.

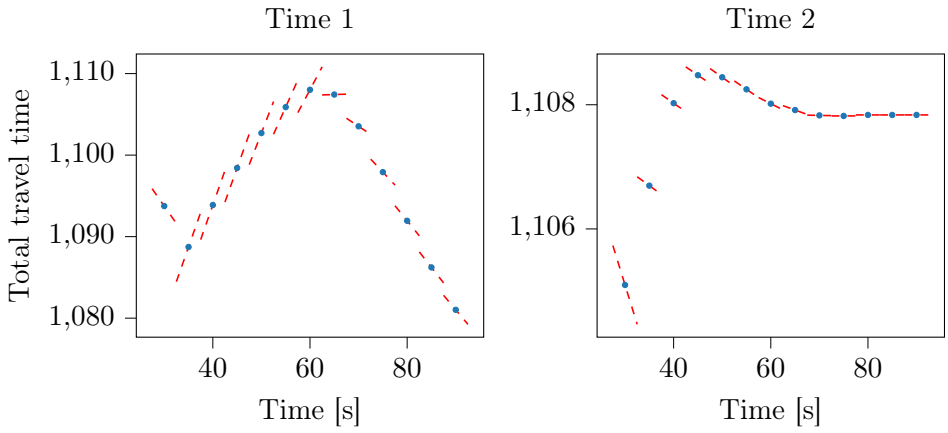


Figure C.21: Total travel time in a network with a two-to-two junction for different cycle times of the traffic light. Increment of 5 seconds.

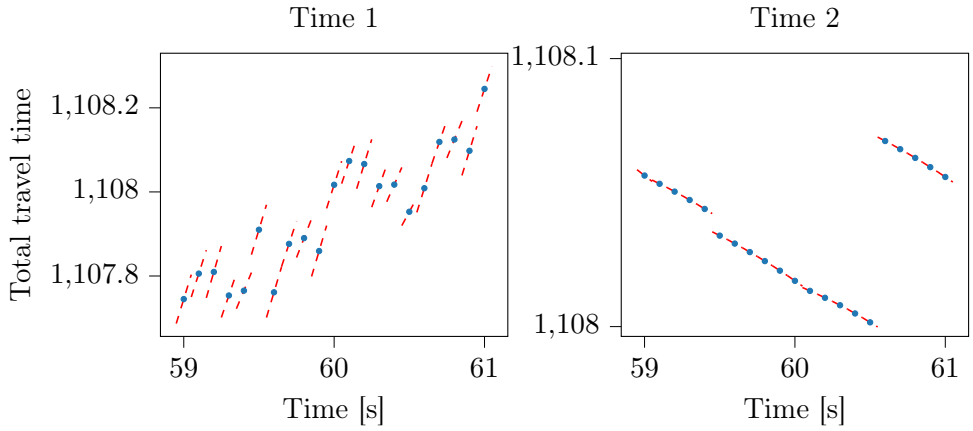


Figure C.22: Total travel time in a network with a two-to-two junction for different cycle times of the traffic light. Increment of 0.1 seconds.

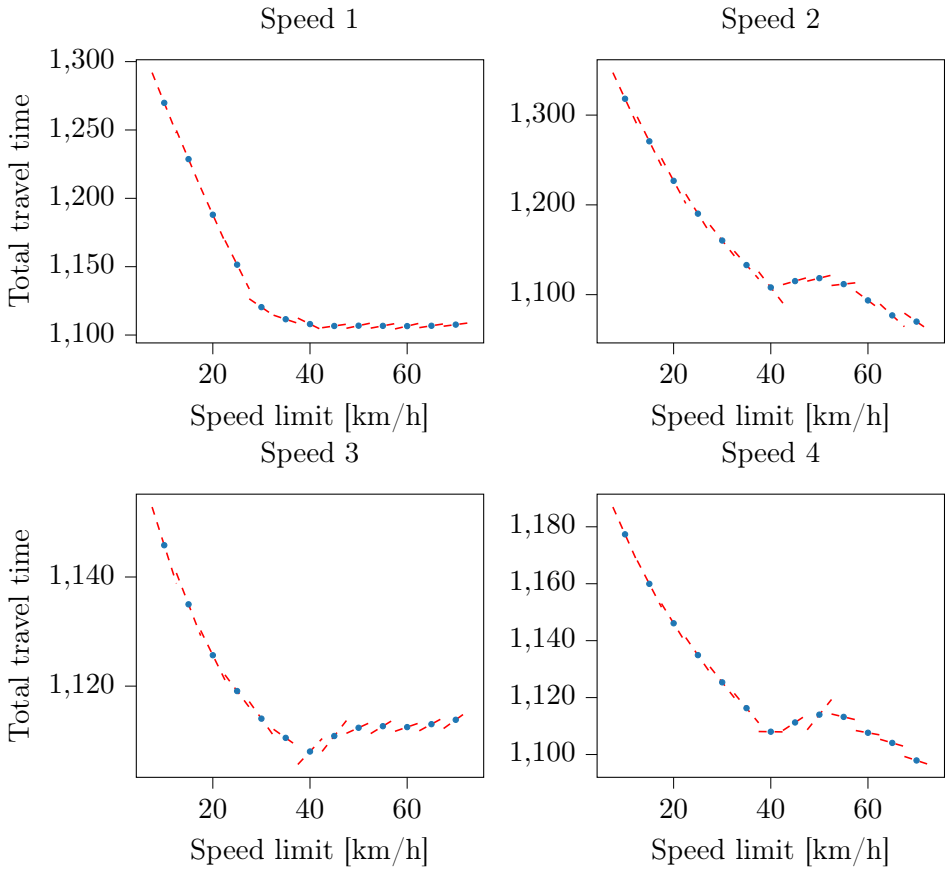


Figure C.23: Total travel time in a network with a two-to-two junction for different speed limits. Increment of 5 km/h.

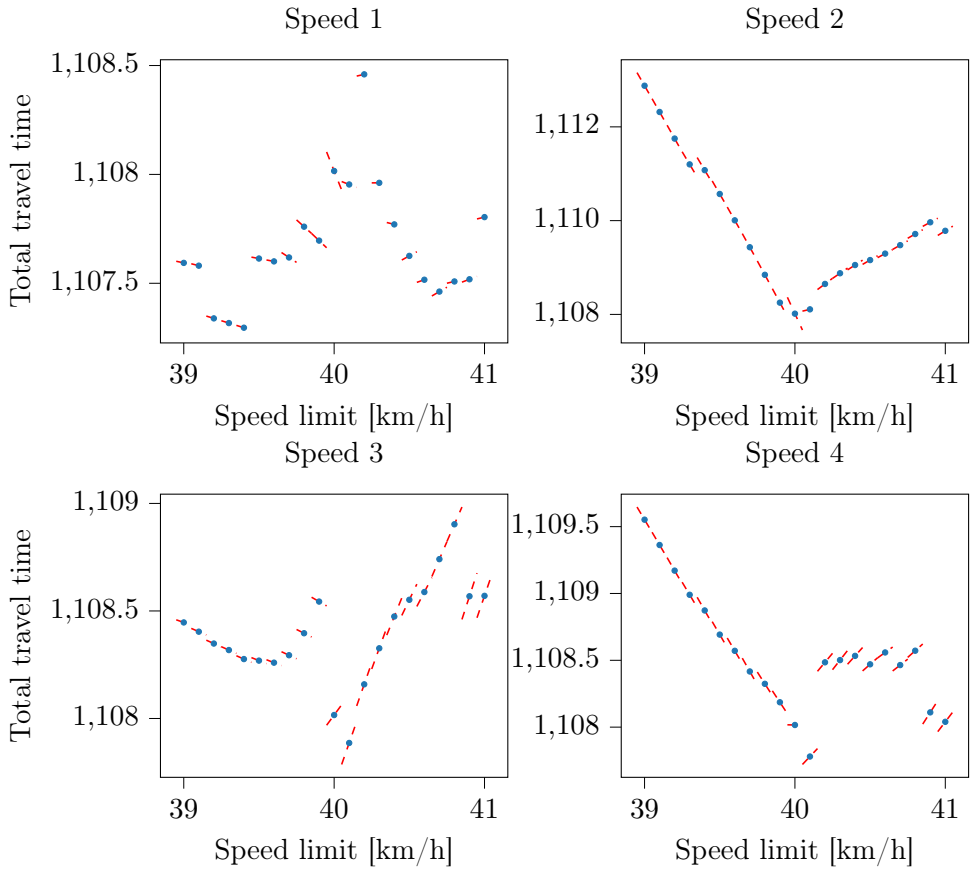


Figure C.24: Total travel time in a network with a two-to-two junction for different speed limits. Increment of 0.1 km/h.