Model predictive control approaches for urban traffic networks: A comparison between optimization algorithms.

M. de Vette

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Model predictive control approaches for urban traffic networks: A comparison between optimization algorithms.

Master of Science Thesis

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

M. de Vette

December 4, 2017

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

Abstract

With the increase of human population, more and more humans depend on transportation for their everyday tasks (e.g. going to work, shopping, bringing their children to school). Traffic networks will be more likely to congest with the increase of traffic flow in these networks. Traffic control was introduced to improve the flow of vehicles inside the network by altering the maximum velocity of the vehicles (on freeways) or controlling the time traffic lights are green (urban traffic networks). More recent development also aim at decreasing the amount of emission gasses produced by the vehicles inside the traffic network.

In this thesis, the effect of traffic light control on urban networks is investigated, with the main focus on the optimization algorithms used to solve the optimization problems resulting from a model predictive control approach. This will be done by using a macroscopic traffic flow model (the S-Model) to simulate the real-time traffic flows inside a network. A microscopic emission model (the VT-micro emission model) is further added to also include impact of the vehicle behaviour (accelerating, decelerating, driving at a constant speed, etc.) on the total amount of emission gasses that are exhausted by the vehicles inside the traffic network.

In this thesis the optimization algorithms will be investigated with respect to the reduction of the cost function as well as the computation time needed compute the (sub)optimal solution. The cost function can consist of the Total Time Spent (TTS) by the vehicles inside the network, the Total Emissions (TE) the vehicles exhaust while inside the network, or a combination of both.

We make use of Matlab for implementation of the models and the optimization algorithms and SUMO as a traffic simulator, which will be used to simulate a real-time traffic network. Data from the SUMO simulator is fed to Matlab and used for calculation of the optimal control input. This input is sent back to SUMO and used to control the traffic lights for the given cycle.

The traffic flow model and the emission model are both non-smooth and non-convex, which in general would require the use of a global optimization algorithm together with multiple starting points. We also use a smoothening function on the models to work with a local optimization algorithm that works with derivatives of the cost function and both models. By using multiple starting points for this method, we hope to obtain similar results. The optimization algorithms that are implemented and investigated in this thesis are: the Genetic Algorithm (GA), the Simulated Annealing (SA) algorithm, the Pattern Search (PS) algorithm, and the Resilient backPROPagation (RPROP) algorithm.

The first three optimization algorithm mentioned are more commonly used in practice, and therefore were already implemented into the Matlab toolbox. The RPROP method, the smoothening function for the traffic flow and emission models, and the calculation for the derivatives were implemented in Matlab as part of the thesis work. Furthermore a Fixed Time (FT) controller was also implemented to prove control could increase the performance of the network.

To compare the results of the four optimization algorithms, four different scenarios are considered. The final results show that all control methods perform better than the FT controller. Overall the GA algorithm performs best without using a multi-start approach, with PS and SA having similar results. The RPROP method is either close to the other methods (0-2%) or quite far off (10-15%), depending on the scenario. When looking at computation time, PS is the fastest. It is twice as fast compared to the GA in most scenarios. SA however takes around fifteen times the amount of computation time compared to the GA. RPROP has varying results again compared to the GA algorithm. A better cost function for the TTS as well as a more optimized algorithm for the RPROP method can resolve both of these issues but future work might need to prove this.

Future work consist of adding the emission model to these models and see whether these have an effect on the effectiveness and the computation time of the algorithms. Furthermore, it will be interesting to investigate the differences between only TTS, only TE, and a combination of TTS and TE. Other future work consists of optimizing the RPROP code to improve computation time as well as to try different cost functions instead of the TTS.

Table of symbols

The table below lists the parameters and variables used in this thesis together with their description. Subscripts are added to some to indicate specific vehicles, pieces of road (links) or intersections. In this thesis, vehicle(s) is shortened to "veh" when used for unit notation.

symbol	description	unit
t	time	[s]
t_{cycle}	cycle time for a given junction	[s]
k	discrete time step counter	[-]
v_{free}	freeflow velocity	[m/s]
v_{low}	velocity when vehicle is idling	[m/s]
a_{dec}	deceleration rate	[m/s^2]
a_{acc}	acceleration rate	[m/s^2]
l_{veh}	average vehicle length	[m]
β	the turning rate	[-]
μ	saturated leaving flow rate	[veh/s]
C	capacity of a link	[veh]
n	number of vehicles on a link	[veh]
q	number of vehicles in the queue	[veh]
u	control input (green times)	[s]
α_i	the average traffic flow for type i	[veh/s]

Table of Contents

Chapter 1

Introduction

With the growing populations more and more people will partake in daily traffic to get from location to location. Traffic in urban areas is being controlled by traffic lights to ensure a more safe environment and to help with managing the flow of vehicles. In the early days of traffic control, a fixed-time controller would be used, which caused congestion if some demands were higher than others. This could result in irritation from vehicle drivers who had to wait on a almost empty intersection because the controller would not account for the demand.
Later on, sensors have been placed under the road to measure the number of vehicles waiting before traffic lights. Based on the demand, a control input can be generated to optimize the flow of vehicles. This, however does only optimize the current intersection, which could result in congested links further inside the network, or leaving a link too full could back propagate the congestion to previous links. To control the intersections not only based on the number of vehicles inside the queue, models were created to simulate the flow of traffic. By prediction the traffic for the intersection and/or the traffic network, a better control input can be generated to improve the overall traffic flow.

In today's society the importance of climate change has also become more relevant. Currently, most vehicles power themselves with the use of gasoline or diesel, which results in the emission of substances that can be harmful for the environment. Different processes require more or less power from the engine and thus, the behaviour of vehicles is important for the fuel consumption. This shows the need for emission control to be added to the control of traffic, whether this is urban or highway traffic.

Although all vehicles are different, the behaviour of the vehicles is key to reducing the emission gasses. Emission models have been constructed to predict the amount of gasses a vehicle emits. The goal of using these models is to reduce the total emission, where the main objective is to force vehicles in behaviours that are causing less exhaust of emission gasses, such as driving at a constant velocity with low RPM of the engine. This can be done on highways by adjusting the maximum velocity and instructing drivers such that they know when to shift gears. For urban areas, most of the gasses are being produced by accelerating the vehicles.

Preventing vehicles from needing to slow down or stop completely can result in less emission gasses.

1-1 Problem statement

Traffic (flow) models can be used to control the vehicles inside a network or on an intersection. Maximizing the flow of vehicles will decrease the number of vehicles waiting inside the network. This results in less congestion inside the network and less irritation from drivers. Controllers need to find the optimal solution to this problem, which could have multiple local minimums. Finding and selecting the correct minimum is key to reducing the congestion problems. This however needs to be done within a reasonable time span, as the controller will have to work in real time. Depending on the minimum time a traffic light is green, more or less computation time can be used for the controller to find a solution. Adding an emission model can hinder the traffic flow, as lower velocities might be better for the reduction of emission gasses and thus increasing the number of vehicles inside the network as compared to only optimizing for vehicle flows.

The controller will be optimizing a cost function which is based on the total number of vehicles inside the network and/or the total emission gasses produced from the vehicles inside the network. Finding the (sub)optimal control action within a reasonable time span is the main objective of this thesis.
There are multiple optimization algorithms already known in literature, all with different working principles and restrictions to the methods. Some require a smooth function to be able to acquire the derivatives from the function, while others will only work for convex problems, otherwise finding a local minimum only. The latter will require some of these algorithms to be run for multiple starting point, after which the best performing control inputs from these starting points is selected. Running the algorithm multiple times for different starting points will obviously increase the total computation time for each control time step.

This thesis will investigate some of these known algorithms and compare them to each other based on reducing the cost function, but we will also look at the computation time needed for each of these algorithms. The differences between using a multiple starting points or only using one initial point will be investigated.

1-2 Overview

In my literature survey the models, both traffic and emission, needed for control were investigated. Chapter 2 is a small recap of the history of the models and provides information about the models that have been chosen for this thesis work. Based on the models, a model predictive controller (MPC) can be implemented. This type of controller does not only check the current time step but also considers a prediction of the performance a number of time steps ahead. Information about MPC was also included in chapter 2.
Chapter 3 is a summary for the optimization algorithms that will be used. Information about genetic algorithms (GA), pattern search (PS), simulated annealing (SA), and resilient backpropagation (RPROP) can be found in this chapter.
Chapter 4 is about the case study performed to compare these algorithms, which includes information about parameters, the network and input sequences for the models and algorithms. Four different scenarios are used to compare and these are explained in this chapter. During the simulations, we found some problems that needed to be dealt with before the model would work. Some adaptations are proposed and discussed in this chapter, of which some are already implemented. Finally the results are presented with some remarks about these results.
Chapter 5 contains the conclusion of the research and simulations. The differences between the algorithms based on reducing the cost function and computation time are discussed. The chapter ends with some recommendations for future work and adaptations we wanted to implement but not finish in time.

Chapter 2

Literature survey

2-1 Urban traffic (flow) modelling

In order to predict and mimic the behaviour of traffic (movement and interactions of vehicles, traffic flows, etc.) in analysis and control applications, traffic models can be used. In this thesis, the focus lies within modelling and control of urban traffic (flows). Traffic models can be divided into three categories: microscopic, macroscopic and mesoscopic traffic models.

Microscopic traffic models look at the individual vehicles and use traffic variables such as the velocity, acceleration, time headway [1] and space headway [2] of vehicles [20]. Some examples of microscopic traffic (flow) models are car-following or collision avoidance models [12], which recreate the behaviour of each vehicle with respect to the vehicle in front such that a safe distance is maintained. These models can be developed in more detail by considering the behaviour of different drivers and by changing the parameters of the models for each individual vehicle. Due to the high level of detail, the computations for these models may not always be done in real time for the given time limit.

Macroscopic traffic models describe the behaviour of groups of vehicles and vehicle flows rather than individual vehicles. These vehicle flows have an average velocity and density which can be used to calculate the vehicle flows inside traffic networks. The behaviour of the group of vehicles depends on the traffic situation near the group, depending on the density. Some of these models use look-up tables consisting of average density on one axis and average velocity on the other. These graphs however are different for every intersection or every network, which requires much data gathering to use. Other models in this category use vehicle flows and flow rates to determine the vehicle flows inside the network.

[1]"The time headway is defined as the time difference between two consecutive vehicles that pass a certain location. This can be described as the time needed by the following vehicle to reach the current position of the leading vehicle with its current speed."[20]

[2]The distance between a vehicle and the vehicle directly infront or behind it.

Mesoscopic traffic models group up vehicles with similar or close behaviour. These groups are based on probabilistic functions that divide the vehicles into certain groups with similar behaviour. The vehicle parameters for each individual vehicle can be described by a probabilistic distributions around an average value for the group of vehicles it was placed into. These group properties are then used to predict the traffic flow inside the network. One example of mesoscopic traffic models is the gas-kinetic model [4], which uses the traffic density and the equilibrium velocity on a piece of road to predict the velocity of the vehicles on that given piece of road network.

The category of macroscopic traffic models can be further divided into types of models. One of the approaches used, consisted of finding the flow data for a specific intersection. After which this data was fitted to an auto-regressive model (AR) [9]. This method required to gather data for each intersection that needed to be modelled, but also needs data for different days and times of day to be accurate enough. The same data could be used to create Macroscopic Fundamental Diagrams (MFD) [15], which illustrates the connection between the density of vehicles on the road and the average velocity of these vehicles. Since some intersections have high densities during rush-hour and low outside of these hours, not all data can be collected which gives the need for interpolation/extrapolation. This data is affected by other influences inside the network, such as construction work or accidents which block one or all the lanes of a link inside the network.

An improvement to these models was found in store-and-forward models (SFM) [10]. These models were based on state equations, having the number of vehicles on the road (further referred to as link) as one of the state values. The states were updated each second by adding the number of vehicles entering and subtracting the vehicles leaving the link. This model based approach allows one model to work for all intersections of the system, regardless of time of day. SFM models were the base of more detailed models to model traffic flows on a macroscopic level.

2-1-1 S-model

The urban traffic flow model used in this thesis is the S-model [5], which is a macroscopic traffic flow model based on an earlier model called the SFM. The first improved version of SFM is called the BLX model [5], which added waiting queues to the model to obtain a higher accuracy. Vehicles that enter a link, will arrive in the waiting queue with a time delay.
The main advantage of the S-model over the BLX model is that the traffic flows are updated per cycle time at the downstream traffic light rather than being updated at every second. This can result in reduced computational efforts and improve the computation velocity of the S-model. The theory found for the S-model also include formulas to work if intersections have different cycle times.

The S-model has two states for each link, namely the number of vehicles on the link (n) and the number of vehicles within the waiting queue (q). The variable n is updated based on the conservation of the vehicles on the link, where the variable q is updated similar but for the waiting queue.

$$n(k+1) = n(k) + t_{\text{cycle}} \cdot (\alpha_{\text{enter}}(k) - \alpha_{\text{leave}}(k)) \tag{2-1}$$

$$q(k+1) = q(k) + t_{\text{cycle}} \cdot (\alpha_{\text{arrive}}(k) - \alpha_{\text{leave}}(k)) \tag{2-2}$$

with

$$\alpha_{\text{arrive}}(k) = \frac{t_{\text{cycle}} - \tau(k)}{t_{\text{cycle}}} \cdot \alpha_{\text{enter}}(k - \gamma(k)) + \frac{\tau(k)}{t_{\text{cycle}}} \cdot \alpha_{\text{enter}}(k - \gamma(k) - 1) \tag{2-3}$$

$$\tau(k) = \text{rem}\left(\frac{(C - q(k)) \cdot l_{\text{veh}}}{v_{\text{free}} \cdot t_{\text{cycle}}}\right)$$

$$\gamma(k) = \text{floor}\left(\frac{(C - q(k)) \cdot l_{\text{veh}}}{v_{\text{free}} \cdot t_{\text{cycle}}}\right)$$

In these equations α is a vehicle flow (entering, arriving or leaving flow depending on the subscript), t_{cycle} is the cycle time and k is the time index. The rem-function is the remainder after division function (Matlab) and the floor-function (Matlab) rounds the value it is given to the nearest integer rounded down.

In this thesis, some simplifications will be made to the S-model. We assume that the vehicles that enter the link can reach the beginning of the waiting queue in one or less than one cycle. This can only be done if the parameters v_{free} and t_{cycle} are large enough compared to the total length of the link (uncongested the top part of the fracture becomes equal to the length of the link or a slightly smaller value). Considering this model will be used for urban traffic control, the parameter for v_{free} should range somewhere between 30km/h and 50km/h. Multiplying this with the cycle time of 60 seconds the length of the link should be less or equal to 500m for the 30km/h case. In this thesis the links are equal to 500m and since there are no constraints on the velocity, it is presumed larger than 30km/h. Therefore this assumption should be justified and due to this simplification the new equations become:

$$\alpha_{\text{arrive}}(k) = \frac{t_{\text{cycle}} - \tau(k)}{t_{\text{cycle}}} \cdot \alpha_{\text{enter}}(k) + \frac{\tau(k)}{t_{\text{cycle}}} \cdot \alpha_{\text{enter}}(k-1) \tag{2-4}$$

$$\tau(k) = \frac{(C - q(k)) \cdot l_{\text{veh}}}{v_{\text{free}} \cdot t_{\text{cycle}}}$$

The number of vehicles that leave a link per cycle time may depend on the following three factors:

1. The saturated leaving flow rate.
2. The number of vehicles waiting in and arriving at the waiting queue within the given cycle.
3. The level of congestion in the next link, which determines the available free space on that link.

This results in the following equation:

$$\alpha_{\text{leave}}(k) = \min\left(\frac{\beta \cdot \mu \cdot g(k)}{t_{\text{cycle}}}, \frac{\beta \cdot q(k)}{t_{\text{cycle}}} + \alpha_{\text{arrive}}(k), \frac{\beta \cdot (C_{next} - n_{next}(k))}{t_{\text{cycle}}}\right) \tag{2-5}$$

where μ is the saturated vehicle rate [veh/s], C_{next} and n_{next} are the capacity and number of vehicles on the link of the next link, g is the green time [s], which is our control input and β, is the turning rate, i.e., the fraction of vehicles in the waiting queue that intend to go to the next link. From Figure 2-1, that for a 4-way intersection there are three values for α_{leave} and thus three different fractions of parameter β. These leaving flows are part of the entering flows for the other links and can be substituted in the equations.

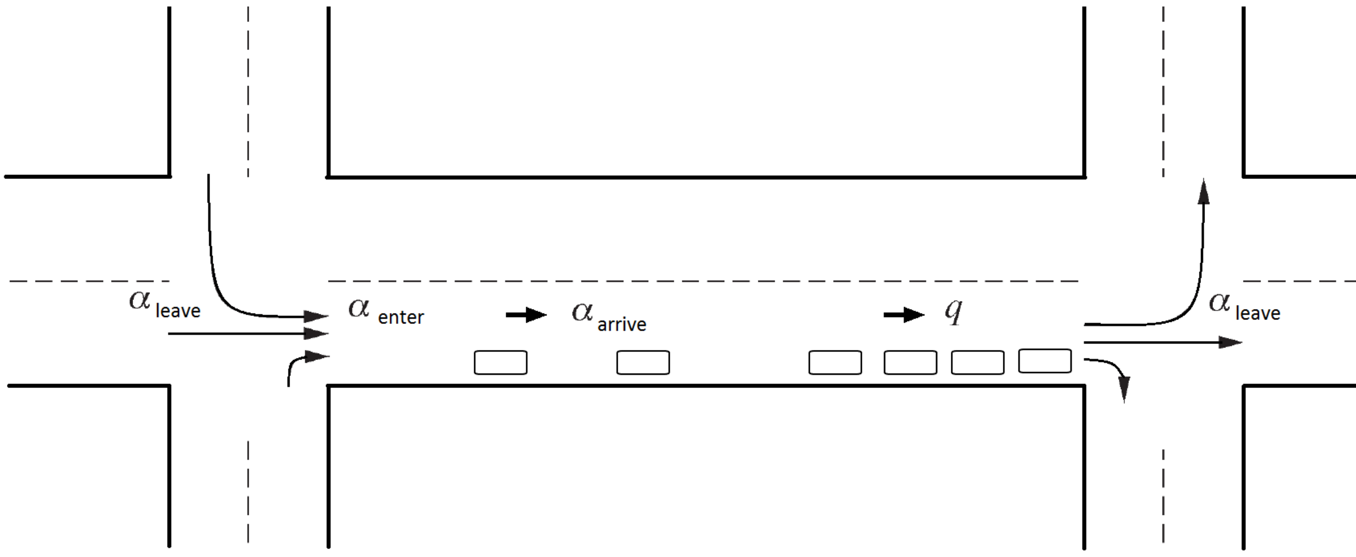

Figure 2-1: Two intersections with variables used in the S-model

2-2 Emission modelling

In this thesis, in addition to the traffic congestion, we will consider emissions of the vehicles. Therefore, we should use the emission models as well. Similarly to the traffic flow models, the level of detail in emission models can range from low to high.
Some of the emission models such as COPERT [16] and MOBILE [3], both macroscopic models, compute the emissions based on the vehicles parameters (type/weight/ageing) and the average velocity of the vehicles. More recent models take into account the effect of the acceleration and deceleration of the vehicles on the emissions. Examples of such models include CMEM [19], which (like COPERT and MOBILE) is very detailed and hence suitable for offline simulation of large traffic networks.
For this thesis however the size of these models were presumed too large for the scope of this thesis. Since we cannot change the vehicles in the network, their behaviour is of more importance than parameters like weight/ageing etc. used in these models. For real-time networks this data is not available for the controller to begin with, therefore it seems off to implement it for this thesis.
Because of this, less detailed models were needed which could be used in this thesis. The two models I found were the VT-Micro emission model[1] [18] and the VERSIT+ emission model [14] [2]. These models are microscopic models, which use individual vehicle variables such as velocity and acceleration of the individual vehicles to calculate the emission for each vehicle.

Since the S-model is a macroscopic flow model, data regarding the individual vehicles should be extracted from the model first. The S-model was integrated with the VT-micro emission

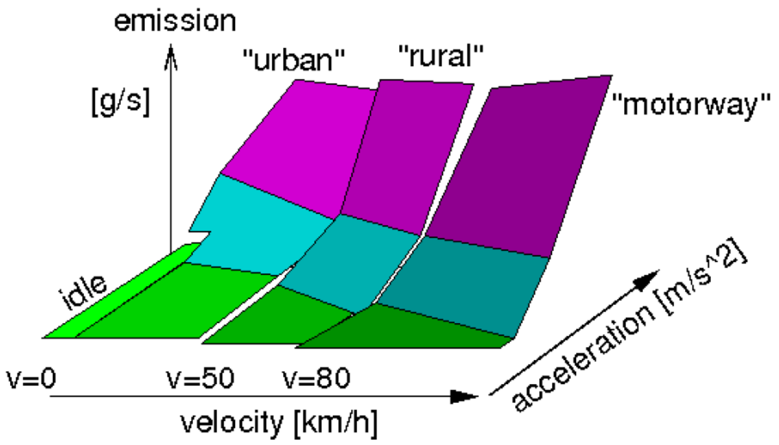

Figure 2-2: Emission zones sketch for VERSIT+

model in [6] based on six vehicle behaviours:free flow speed, idling, accelerating, decelerating, stop-and-start and non-stop behaviours. Each of these behaviours correspond to a velocity and acceleration/deceleration, as well as a total time they exhibit this behaviour. This groups each individual vehicle into six behaviours, making it possible to reduce overall calculations of the emission model. The emission for each group needs to be calculated and multiplied by the number of vehicles exhibiting the behaviour, rather than each individual vehicle.

2-2-1 VERSIT+

The VERSIT+ emission model [14][2] is based on a combination of linear functions of the velocity and acceleration of individual vehicles. The equations used by this model are:

$$w_i = a_i + 0.014 v_i \tag{2-6}$$

$$a_i = \frac{v_i - v_{i-1}}{3.6}$$

$$E_e = c_0 + c_1 * \mathbf{max}(0, w) + c_2 * \mathbf{max}(0, w - 1).$$

where w_i is called the dynamic variable, a_i is the acceleration [m/s^2] and v_i is the velocity [km/h], E_e is the emission of emission gas type e for each individual vehicle per second [g/s]. This value can be summed up over all vehicles and over all the simulation time steps to get the total emission for emission gas type e. The equation for the acceleration should be replaced, since for our model, it will calculate the average acceleration for the entire cycle. Using the method described above from [6], we can substitute the emission for each behaviour time rather than the average acceleration for each vehicle per cycle.

As seen in Figure 2-2, the emission is defined for 10 different areas or domains in the velocity/acceleration plane. Depending on the domain, different constants for c_1 and c_2 are inserted into 2-6.These constants could also vary for different groups of vehicles with roughly equal parameters (weight/fuel type etc.).

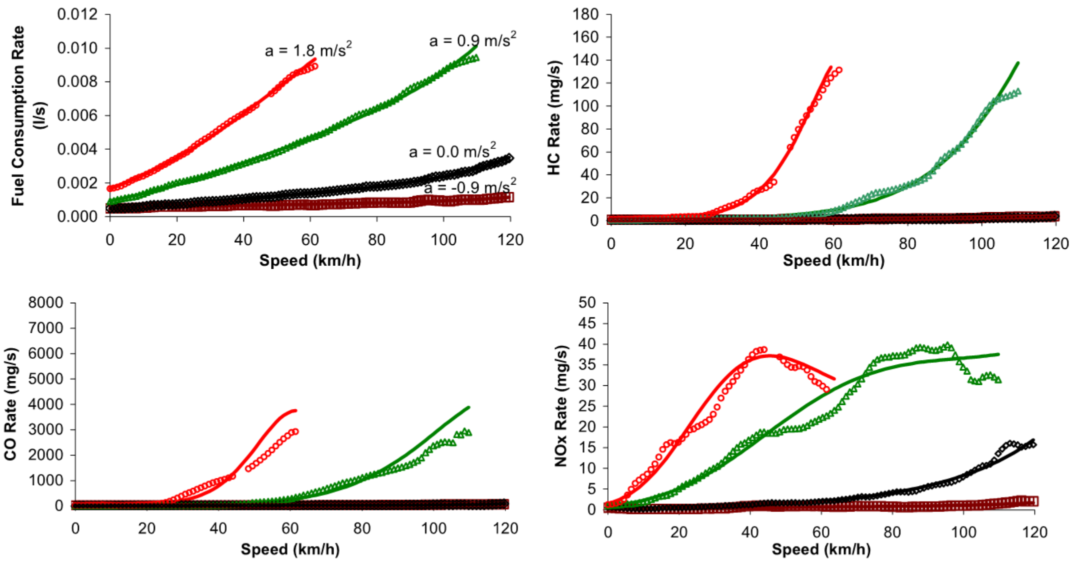

Figure 2-3: Validation data for the VT-micro model

The domain for the velocities are 0-50 [km/h], 50-80 [km/h] and any velocity over 80 [km/h]. A small area is also created for idling velocity where the velocity and acceleration are both low.

2-2-2 Virginia Tech Microscopic (VT-Micro) emission model

The VT-Micro model [1] [18] uses polynomials based on the individual velocity and acceleration of the vehicles to compute the emissions. This model uses both linear, quadratic and cubic terms for both these variables. The formulas used by this model are:

$$E_e = \exp(\bar{v} K_e \bar{a}), \tag{2-7}$$

with

$$\bar{v} = \begin{bmatrix} 1 & \mathbf{v} & \mathbf{v}^2 & \mathbf{v}^3 \end{bmatrix}$$

$$\bar{a} = \begin{bmatrix} 1 & \mathbf{a} & \mathbf{a}^2 & \mathbf{a}^3 \end{bmatrix}^T.$$

where E_e is the emission of type e emission gas for each individual vehicle per second. This value can be summed up over all the vehicles and over all the simulation time steps to get the total emissions. The matrix K_e is filled with some emission coefficients for a given emission type e (e.g., CO_x, NO_x, HC), which are used determined by fitting the model with real-life measured data. For more accuracy of the model, different matrices for K_e can be used to separate acceleration and deceleration. The matrix K_e is defined for a group of vehicles exhibiting the same behaviour, averaging out the data gotten from each vehicle into data for a given class of vehicles to reduce the amount of matrices stored by the model.

VT-Micro uses the microscopic data from individual vehicles (i.e., velocity and acceleration) to calculate the emissions. The simulations with the VT-micro model (Figure 2-3) shows there is a difference between no acceleration and deceleration on the emission gasses. The given data for the validation of the model in the paper loses its accuracy for higher velocities, however for urban traffic the maximum velocity should not exceed 60 [km/h]. This does not verify the fit to the S-model we will be using, nor if this will work with the vehicle behaviours discussed above and in [6]

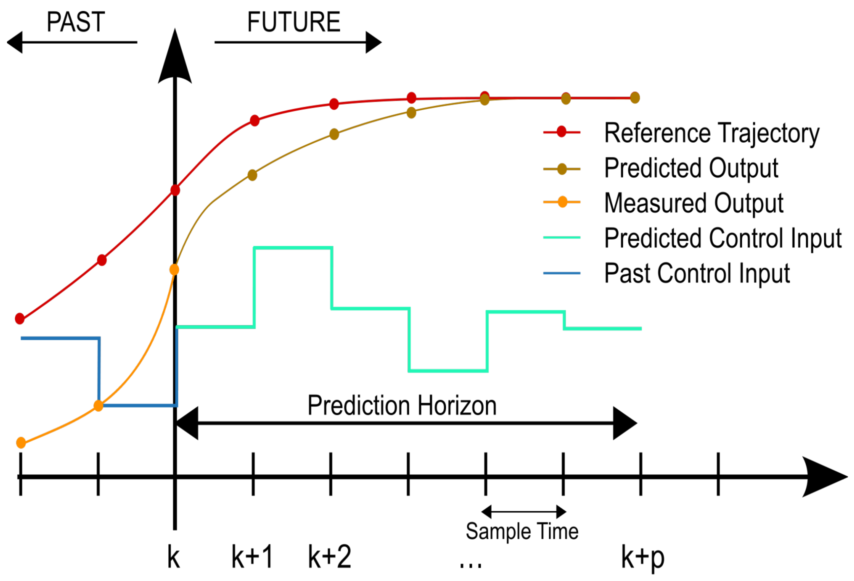

Figure 2-4: Concept of model predictive control.

2-3 Model Predictive Control (MPC)

Model predictive control uses a model and disturbance model of a given system (or process) to predict the future states of the given model. The amount of steps the model will look ahead is called the prediction horizon of the model. Using this approach the system or process can be optimized for not one but multiple time steps ahead. The MPC problem can have constraints, both equality and inequality constraints, and bound on control inputs or states. The control input can also be kept constant after a given time for the rest of the prediction horizon. The amount of steps ahead the controller is allowed to change the value for the control input is called the control horizon.
Given the constraints and bounds, and with the models for both the system and the disturbance the controller tries to compute the optimal solution for all the given time steps ahead. After finding the solution only control input for the first time step is implemented. At the next time step the system will sent new states to the controller and a new prediction is made. This removes inconsistencies between the system and the model, mainly caused by model inaccuracies or different disturbances than the ones modelled.
The optimal solution for the model predictive controller is defined by minimizing a cost function. This cost function is defined for the entire prediction horizon, therefore needs to be minimized for the entire prediction. This can lead to a more optimal solution, since the prediction can foresee aspects like overshoot and future disturbances. Figure 2-4 shows a more slow reduction between reference and predicted output since it uses a prediction over multiple time steps, rather than using a huge spike to compensate for the difference in just one time step, having to remove the overshoot in the next time step.

2-3-1 MPC application used in this thesis

One of the reasons MPC was opted for this thesis was the time variables in some of the equations. The equation for the arriving flow 2-4 depends on the previous and current time step. The leaving flows of some links are used as entering flows for other links one time step later. To avoid saturating links further into the network, looking ahead and predicting the entering flow from the previous link can avoid this saturation by letting more vehicles leave the link if possible.

For this thesis the cost function consists of the total time spent (TTS) and/or the total emission gasses (TE). Since the system has no mechanical constraints but only needs to sent an electric signal, the control input can vary quite heavily between time steps. For mechanical systems each actuator has a maximum difference for the change in control input and also a maximum for the control input. Therefore the term to minimize either the control input or the difference in control input is not included into the overall cost function.

The TTS is a summation of the time a vehicles spents inside the network, summed over all of the vehicles present:
$TTS = \sum_{k=1}^{\mathrm{N}_p} \sum_{i=1}^{11} n(k,i) * t_{\text{cycle}}$
where N_p is the prediction horizon, k the time step and i the link number. By reducing the TTS the time vehicles spent inside the network should be reduced.
For the TE the following equation holds:
$TE = \sum_{k=1}^{\mathrm{N}_p} \sum_{j=1}^{3} te(k,j)$
where j stands for the type of emission gasses and te equals the total emission for the network for the given time step, as found from the emission models discussed in the previous section. This thesis opted to investigate three types of emission gasses, namely CO_x, NO_x and HC.

The value for the TTS and/or the TE will be summed up for all the time steps the MPC is predicting. This can be changed by altering the prediction horizon (N_p). As stated, vehicles leaving a link will be used for the entering flow of the next link. Having a higher prediction horizon can ensure links further into the network can already prepare for incoming traffic in a few time steps. Larger prediction horizons do increase the computation time, so the value cannot be too large.
Standard MPC also has the option to set a control horizon, after which the control action is kept constant. Since this value is equal or smaller than the prediction horizon it can reduce the different amount of control inputs an optimizer has to check, reducing the computation time. Since these values depend on the model for the network used, I will specify them in chapter 4 together with this network.

2-4 Summary

A model is needed to simulate the traffic flow for urban traffic networks. In this thesis, we will compare certain optimization algorithms for traffic signal control in real-time applications. Two options are possible: a very accurate model or a less detailed model for real-time optimization. The second option has been considered in this thesis such that the computation time of the optimization algorithm would remain low enough such that it works for real-time applications.
This led to the choice of a macroscopic traffic flow model, which uses the average velocity of the vehicles to predict the flow of vehicles inside a traffic network. We have chosen for the S-model in this thesis, which predicts the flow of vehicles inside the network based on the states, which are the number of vehicles on the pieces of road (links) and the vehicles in the waiting queue of those links. The entrance flows at the start of the network are seen as disturbances on the system and the traffic signals are controlled for an optimal flow of vehicles inside the network.
The S-model works for each cycle rather than each second, which creates less computations over time, which is one of the main reasons this model was chosen. Another advantage are the states of this system, which knows how many vehicles are inside the network but also how many vehicles are already waiting inside the queue at the end of the link. Adding these waiting queues has improved the model and thus the accuracy over existing store-and-forward models (SFM). This model can work for any given intersection given that the rough estimate for the turning rate is close to the real-time value. Some other macroscopic models were using specific data for a given intersection, which needed different data for all intersections, which this system does not need.

In this thesis, we also investigate the effect of emission models combined with the traffic flow model. To investigate this effect, emission models were chosen which depend on both the velocity as on the acceleration/deceleration of the vehicles. Both these variables were proven to be of great significance for the accuracy of emission based models.
In this thesis we have not considered the characteristics of the vehicles (i.e., engine type, age, weight) but created one type of data for a general vehicle group. The effect of vehicle behaviour (breaking, accelerating, moving at constant velocity, etc.) will be investigated to see if control of the traffic lights to alter this behaviour can reduce the emission gasses exhausted over the network.
To see these kinds of behaviour, microscopic emission models were chosen. A conversion between the macroscopic traffic flow model and the microscopic emission model was already created in [6], by having six different vehicle behaviours. By only considering these six vehicle behaviour classes and dividing all vehicles in one of these six classes, the computation time can be lowered for real-time applications.
Two emission models were investigated, namely the VERSIT+ and the VT-micro model. VERSIT+ uses a combination of linear equations with regards to the velocity and acceleration where VT-micro uses linear, quadratic and cubic relationships between these variables. The VT-micro model has been chosen, the main reason being that it uses the deceleration in the equations. VERSIT+ can have negative accelerations, but if the velocity is too low it will have no impact on the emission (which is almost equal to no acceleration).

Chapter 3

Optimization algorithms

In this thesis we compare the performance of different optimization algorithms for solving the MPC control problem of an urban traffic network. Optimization algorithms can be divided into two categories: global and local. Global optimization algorithms search the entire domain to find the global minimum, while local optimization algorithms will stop searching as soon as the first minimum is found (although it may be a local minimum only and not the global minimum). Local optimization algorithms can be used with different starting criteria to find several local minimums and choose the smallest one from the found set of minimums. An optimization problem may be convex or non-convex. Convex function only have one minimum, which by definition is the global minimum. Rather than using global optimization algorithms, methods to approach a given model as a convex function have been developed. Although reducing the accuracy, only one minimum will be found, which enables to use of local optimization algorithms. An optimization problem may be smooth or non-smooth. Smoothness allows actions such as taking derivatives of the functions used in these models, which are often used in (local) optimization algorithms. Efficient approaches have been developed to convert non-smooth functions to smooth functions. Examples of non-smooth functions include min/max-statements, floor/trunk functions and if-statements. Non-smooth models cannot use derivatives to find the minimum, therefore most global optimization algorithms use a systematic search over the domain for all the optimization variables.
Smoothing the model and approaching the model by a convex function/model reduce the accuracy of the original model. To see the influences of reducing the accuracy, three global optimization algorithms and one local one which uses smooth functions have been selected to be used in this thesis. All methods have options for stopping criteria, which should be te same for all the methods for a fair comparison. Corresponding details will be specified in chapter 4.
Matlab has a toolbox for the three global optimization algorithms (GA, PS and SA), while the code for the RPROP method has been build from scratch for this thesis. The working principle for these algorithms are listed below [1].

[1]The help-directory of Matlab contains information about the optimization techniques, as well as data on what all the different options do and why one might change them.

3-1 Genetic algorithm (GA)

The genetic algorithm (GA) [17] has been developed based on the DNA characteristics. The method assumes two or more parents with certain properties. Some properties of one parent will be given to the offspring, while the other characteristics will be obtained from the second parent, this is called crossover. This creates a new generation of potential parents that can be used for the algorithm in the next time step. From this new population, only the best performance or properties will be selected as the new generation. The GA uses random numbers to select the offspring which will be used for the next generation, where better performing offspring will have a higher chance to be selected. Some new properties, which neither of the parents have had can be obtained by mutation inside of the DNA/data string.
By randomly selecting the new generation of parents from the offspring, as well as the mutations that can happen within the creation of the offspring, the algorithm can avoid being stuck within local minimums. By using these offspring again within the new generation, and after crossover and/or mutation, the new offspring can have a value near a new local minimum, which lies far away from the current minimum.

The main process for this optimization algorithm is the creation of new offspring by crossover and mutation. In crossover, some characteristics from the first parent and some from the second parent are selected to create an offspring. The selection procedure can be designed by the user. In our specific problem, which searches for optimal values for green times in an urban traffic network, the length of the vector which contains our optimization data cannot change. Therefore the amount of optimization variables after crossover should be equal to that for each parent. The option Matlab provides are single point selection and two point selection. For single point selection, a single point is chosen at random, after which all the optimization variables before that point are taken from the first parent and all the variables after this point are taken from the second parent. For the two point selection algorithm not one but two points will be chosen, where after the second point the variables are again taken from the first parent.

Mutation is the change of optimization variables within an individual, this is independent of the parents. Usually mutations are rather small, but this can again be altered by the user of the GA. The optimization variable that will change is chosen at random as is the sign of the small number specified that will be added to this variable. The chances of mutation inside an individual is usually kept rather low, so cross-over will have a larger impact on the new generation than mutation, however this can be changed by the "crossover fraction" option provided by the Matlab toolbox. The default value is 80% crossover and 20% mutations to ensure the optimization is converging to an minimum, but will not be stuck inside a local one.

Other options that should be specified for GA are the population and the parent size. A higher number of parents can create more variation in the offspring depending on how the parents are selected. However, a very high number of parents may reduce the effectiveness of the selection algorithm while increasing the computation time. As for the population, larger populations will result in more offspring and thus more chances of a desired DNA/data string, but will increase the computation time.

GA has the option of implementing constraints. However, due to the nature of the algorithm it does not work well with constraints. As mentioned crossover and mutation will randomly select variables and change them, which could result in violations of the constraints. For our urban traffic network, only two green times are needed for each controlled intersection. Therefore the second green time can be substituted by the cycle time minus the first green time, resulting in less control variables but also removing the equality constraints. A larger intersection with more green times could influence the effectiveness of the GA.

3-2 pattern search (PS)

Pattern search uses a smart search procedure over the domain of the optimization variables. The algorithm will search from an initial starting point for other points which can reduce the realized value of the cost function provided to the algorithm. Different strategies for searching the domain can be chosen. The most commonly used strategy is to search in all directions for all the variables, both in the positive and negative directions. An adaptation to this pattern is to only use the positive direction and to consider one more vector consisting of all the negative values. This second option reduces the number of directions from $2 \cdot N$ to $N+1$, where N is the number of optimization variables. For a problem with three optimization variables, this will result in the following search vectors:
option 1 ($2 \cdot N$) : [1 0 0] [0 1 0] [0 0 1] [-1 0 0] [0 -1 0] [0 0 -1]
option 2 ($N+1$): [1 0 0] [0 1 0] [0 0 1] [-1 -1 -1]

The most important decision to be made, is whether "full search", from the Matlab toolbox options, should be executed or not. A "full search" will compute all the new costs and will select the best performing set of optimization variables found in the current iteration as the new search point. Without a "full search", as soon as a better cost is found the algorithm stops and uses that new point for the next iteration.

Pattern search will alter the step size of the algorithm. If a new and thus lower cost value is found, the step size will be increased to search over a larger domain. When a lower cost value cannot be found, the step size is reduced to narrow the search domain, leaving the initial starting point unchanged for the next iteration step. Pattern search does not have any strategy for exiting the local minimums. Therefore, a multi-start approach must be implemented to increase the chance of finding the global minimum.

3-3 Simulated annealing (SA)

Simulated annealing (SA) [7] has been developed inspired by the cooling processes of metals used in industry. At the beginning of the cooling, the temperature drop is quite high, it is gradually reduced to zero. This inspired the idea of taking large steps at the beginning and gradually reducing the step size until a minimum is found. After some iterations the algorithms can be "reheat" to increase the searching step again which allows the algorithm to avoid local minimum. This process is called the "re-annealing interval" in the Matlab toolbox and can be set by the user.
Simulated annealing does not work with all types of constraints. Only lower and upper bound constrains on the optimization variables can be used. Therefore, substituting the second green time for the cycle time minus the first green time will cause the constrain to be implemented into the problem statement. This also reduces the number of optimization variables to half, which improves the computation time of the algorithm.

Matlab has different options for the SA algorithm, with the most important being the "temperature curve". This curve is used as to define the step size the algorithm will use. As stated, a high temperature corresponds with a large step size which lowers in value as time passes. The possible choices for this curve in the Matlab toolbox include a linear, an exponential, a logarithmic or an user defined function. The default option is the exponential function which creates the following temperature curve:

$$T(k) = T_0 * 0.95^k \tag{3-1}$$

where T is the temperature at time step k and T_0 the initial temperature. Note that after re-annealing, the initial temperature is changed and the time step is reset to zero.

To avoid getting stuck in a local minimum, the SA algorithm has an acceptance function for each new obtained value, which works with a probability of accepting the current value for the optimization variables for the given iteration. If this updated value is lower than the previous value, the algorithm accepts the new value with a probability of 1. However, if the new value is larger than the previous one, the probability P for accepting the current value is equal to:

$$P = \frac{1}{1 + \exp(\frac{\Delta}{\max(T)})} \tag{3-2}$$

$$\Delta = \text{cost}(i) - \text{cost}(i-1)$$

where cost is the value of the cost function, depended on i which denotes the iteration number. By accepting not only the values that decrease the cost function but also some values which increase it, the SA algorithm can potentially move to the global minimum after finding a local minimum.

3-4 Resilient backPROPagation (RPROP)

Resilient backpropagation (RPROP) [2] is a method that will be used together with the theory for Pontryagin's Minimum Principle (PMP) [11]. The theory from PMP considers the following discrete-time representation for a dynamical system:

$$x(k+1) = f(x(k), u(k)), \tag{3-3}$$

where x is the state variables (n and q in the S-model) and u is the control input (the green time in the traffic control problem). The cost function is defined by:

$$J_{\text{tot}} = J_{\text{final}} + \sum_{1}^{k_{\text{final}}-1} J_s(k) \tag{3-4}$$

where J_{tot} is the total cost over the entire prediction horizon, J_{final} is the final cost, and $J_s(k)$ is the stage cost at time step k. The subscript final for variable k refers to the last time step, which in our case is one value larger the prediction window of the MPC ($Np+1$).

Using these equations, a new function is defined as:

$$H(x(k), u(k), \lambda(k+1), k) = J_s(x(k), u(k), k) + \lambda^T(k+1) \cdot f(x(k), u(k), k) \tag{3-5}$$

which has to following conditions which ensure optimalization:

$$\begin{aligned} x(k+1) &= \frac{\partial H}{\partial \lambda(k+1)} = f(x(k), u(k), k) \\ \lambda(k) &= \frac{\partial H}{\partial x(k)} = \frac{\partial J_s}{\partial x(k)} + \frac{\partial f}{\partial x(k)} \cdot \lambda(k+1) \\ \frac{\partial H}{\partial u(k)} &= \frac{\partial J_s}{\partial u(k)} + \frac{\partial f}{\partial u(k)} \cdot \lambda(k+1) = 0 \end{aligned} \tag{3-6}$$

These formulas are for models without (in)equality constraints, since we are substituting these in the model they will not be included in this thesis.

The RPROP algorithm uses this function $H(x(k), u(k), k)$ and all the variables inside to compute the optimal value for the control inputs. This method starts at the final time step and works its way back towards the first time step. The process of this method works by using the following steps:

Step 1: Set the iteration index at zero and set a feasible initial sequence for the control input u_0. This value for the initial control input can be chosen based on the states or completely random, the latter often used in combination with multiple starting points for the optimization algorithm.

[2] Information gotten from supervisor, can be found in [8]

Step 2: Use $u(k)$ and $x(k)$ to calculated $x(k+1)$ using (3-3), starting from $k = 1$ to $k = k_{\text{final}}$. After obtaining these values, $x(k + 1)$ and $u(k)$ can be used to obtain λ via backwards integration:

$$\lambda(k) = \frac{\partial J_s(x(k), u(k))}{\partial x(k)} + \lambda(k+1) \cdot \frac{\partial f(x(k), u(k))}{\partial x(k)} \tag{3-7}$$

by starting from $k = k_{\text{final}}$ and using the final condition:

$$\lambda(k_{\text{final}}) = \frac{\partial J_{\text{final}}(x(k_{\text{final}}))}{\partial x(k_{\text{final}})}$$

Step 3: Use the value determined for λ in step 2 to compose the reduced gradient, denoted as G_r, based on the following equation:

$$G_r(k) = \frac{\partial J_s(x(k), u(k))}{\partial u(k)} + \lambda(k+1) \cdot \frac{\partial f(x(k), u(k))}{\partial u(k)} \tag{3-8}$$

Step 4: Apply the RPROP method to get the new control inputs for the next iteration step:

$$u_{i+1}(k) = \text{sat}(u_i(k) + \Delta u_i(k)) \tag{3-9}$$

where sat is a saturation function and given by:

$$\begin{cases} \text{sat}(x) = x & \text{if lb} < x < \text{ub} \\ \text{sat}(x) = \text{lb} & \text{if } x < \text{lb} \\ \text{sat}(x) = \text{ub} & \text{if } x > \text{ub} \end{cases} \tag{3-10}$$

where lb is the lower bound and ub is the upper bound on the variable which is to be saturated. Furthermore $\Delta u_i(k)$ is calculated by:

$$\Delta u_i(k) = \begin{cases} -\text{sign}(G_{r,i}(k)) \cdot \eta^+ \cdot \Delta |du_{i-1}(k)| & \text{if } G_{r,i}(k) \cdot G_{r,i-1}(k) > 0 \\ -\eta^- \cdot \Delta du_{i-1}(k) & \text{otherwise} \end{cases} \tag{3-11}$$

where the subscript i is used for the iteration number and η^+ and η^- are multiplication factors to either increase or decrease the step size with. The value for η^+ should be larger than one and the value for η^- should be between zero and one.

Step 5: Calculate the updated value of the cost function and verify if the stopping criteria is reached, i.e.:

$$|J_{tot,i} - J_{tot,i-1}| / J_{tot,i-1} < \sigma, \tag{3-12}$$

$$\text{where } \sigma << 1$$

If not, start a new iteration from step 2 again using the updated values for $u_i(k)$.

Although no constraints are added in these equations, with the bounds on the control input listed inside the saturate function, this method can be adapted with more terms to include (in)equality constraints. As stated with the other optimization algorithms, substituting the constraints into the model will result in less parameters to be optimized as well as removing the equality constraints the algorithms has to ensure.

3-4-1 Creating a smooth model

In order to use a smooth optimization algorithm, such as RPROP, the derivatives of the functions describing the model are needed. Hence, a smooth version of the model should be provided. The models chosen for this thesis, the S-model and the VT-micro emission model, have none smooth functions. The S-model consists of minimum statements in the equation for the leaving flow rate (equation 2-5). The model created to change the macroscopic traffic flow model to the microscopic emission model uses if-statements to know if the traffic flow model is saturated, under-saturated or over-saturated. These if-statements create selector variables, which have a value of zero or one. On the transition from zero to one and vice versa the function for this selector variable is discontinues, which makes the overall model it is used in non-smooth. A smooth function for these selector variables can be described as:

$$\mathrm{S}_{var} = \frac{1 \pm 0.5 \cdot (1 + \exp(\alpha/2))^{-1}}{1 + \exp(-\alpha \cdot (x - x_{\text{bound}}))} \tag{3-13}$$

which will create a function that transitions from zero to one or one to zero, depending on using a plus or a minus sign in the numerator. The value of α can be altered to increase or decrease the smoothening effect on the function. The value for x_{bound} can be use to define where the transition from one to zero has to occur.

The minimum statement can be approached by the following approximation:

$$\min(x_1, x_2, x_3) = \frac{-1}{\alpha} \log(\exp^{-\alpha \cdot x_1} + \exp^{-\alpha \cdot x_2} + \exp^{-\alpha \cdot x_3}) \tag{3-14}$$

which removes the sharp edges where two functions cross, such that a derivative is defined for each point on the function. The value of α can be used to increase or decrease the smoothening, where a low value will result in high smoothening and a high value in no almost no smoothening at all. One problem that might arise is that by using this equation, the function will get negative values when the terms inserted into this equation are close to zero.
Other non-smooth function were already removed from equation (2-3), which were the "rem" and "floor" functions. These functions can also be made smooth, but is not in the scope of this thesis. These smooth functions were already applied in [8](chapter 7), which can be used for more information.

3-5 Summary

In this chapter, we have discussed the optimization algorithms we are going to use for optimizing the green times of our urban traffic network. We have discussed these algorithms, considering two categories: global and non-global (local) optimization algorithms. Global optimization algorithms search the entire domain to find a global minimum. They underlying procedure is based on broadening the search domain after a few iterations. By using a multi-start approach, multiple minimums can be found for both global and local optimizers, which can be used to determine the lowest value among the minima and use that minimum as the global minimum.

Where global optimization algorithms can use any model, local optimization algorithms can be based on derivatives which require a smooth function to ensure a continues derivative over the entire domain. Methods to smooth a model can be used to ensure the model can be used with local optimization algorithms. Local optimization algorithms can only use convex models, such that the found minimum is the only and therefore global minimum. When using non-convex models in combination with local optimization algorithms, adaptations can be made to ensure the success of this combination. First of the model can be approached as a convex function, reducing the accuracy of the model, but will ensure only one minimum can be found. The other solution discussed here, is by using multiple starting locations for the solver to find multiple minima. The lowest value of this minima can be considered the global minimum, assuming enough random starting points have been selected.
The (in)equality constraints for these optimization problems can and will be removed for this thesis by substituting these equations in the model. This will also remove half of the variables that need to be optimized, decreasing the computation time but also increases performance of the algorithms as they are not bounded by (in)equality constraints. The bounds on the optimization variables is still present, however these do not hinder the optimization algorithms as much as the (in)equality constraint do.

Chapter 4

Case study

In this chapter, we propose an urban traffic network to compare the different optimization algorithms discussed in this thesis for optimizing the traffic signals. This urban traffic network, is shown in Figure 4-1, and has three controlled intersections.

This traffic network consists of eleven links, where each link has only one lane. The urban traffic network has four entrances (via links 1, 2, 6 and 7) and three exits (via links 4, 11, and 10). At the end of each link, the vehicles enter the waiting queue from which they leave the link after the light has turned green. At each link a turning rate $\beta_{i,j}$ is defined, where i is the link the vehicles are in the queue and j is the link the vehicles will leave towards. This turning rate describes the number of vehicles that will turn towards link j, as part of the total number of vehicles inside the waiting queue.
As stated the optimization algorithms are supplied with a cost function, which consists of a model with model predictive control. MPC requires a prediction horizon for which the cost function will be defined, which is set to five for this thesis. Vehicles that enter the input link need at least three to four cycles (depending on the length of the link and the free flow velocity) to exit link 11, when also adding congestion a good approach for the prediction horizon would exceed at least this value and therefore was set to five.

To simulate the traffic for the real-time situation, a traffic simulator called SUMO is used. This simulator has its own set of traffic rules to approach the flow of the vehicles, based on car-following models [1]. This simulation will provide the Matlab program with the states (n and q). The vehicle flows for the input links are known to both programs (SUMO and Matlab) and will be provided before the simulation/optimization starts.

The simulations were run on a desktop computer with an Intel Core i7-4790K CPU, 8GB memory and a AMD Radeon R7 200 Series GPU. We used Matlab version R2014b running

[1] "SUMO is mainly a microscopic, space-continuous road traffic simulation. It supports multi-modal and inter-modal ground based traffic. SUMO models individual vehicles and their interactions using models for car-following, lane-changing and intersection behavior." [13]

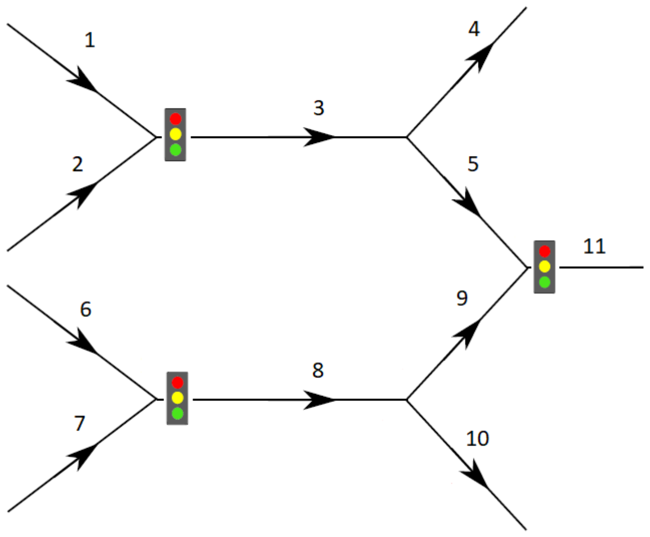

Figure 4-1: The urban traffic network used for the case study in this thesis.

on the windows 10 64bit version of the operating system, which is used to run the optimization algorithms. The traffic simulator program is called SUMO, of which version 0.26.0 was installed.

	l_{veh}	μ	v_{free}	C	a_{acc}	a_{dec}	v_{low}	o
lower bound	5	0.1	8.5	30	0	-5	0.1	1
upper bound	10	2.0	16.0	100	5	0	8.5	80

Table 4-1: Value for the bound of the to be identified parameters

4-1 Parameter identification

Before the simulations are performed, the parameters of the S-model and VT-micro emission model should be identified with respect to the SUMO model. These parameters include:

1. the length of the vehicles l_{veh} [m]
2. the saturation leaving flow of a link [2] μ [veh/s]
3. the maximum velocity/ free flow velocity v_{free} [m/s]
4. the capacity of the links C [veh]
5. the acceleration a_{acc} [m/s] of the vehicles in the urban traffic network
6. the deceleration a_{dec} [m/s] of the vehicles in the urban traffic network
7. the idling velocity v_{low} [m/s] of the vehicles in the urban traffic network
8. the smoothening parameter o used for the RPROP method [-]

The lower and upper bounds for these parameters are given in Table 4-1.

[2]The maximum flow that can leave a link per model sampling cycle.

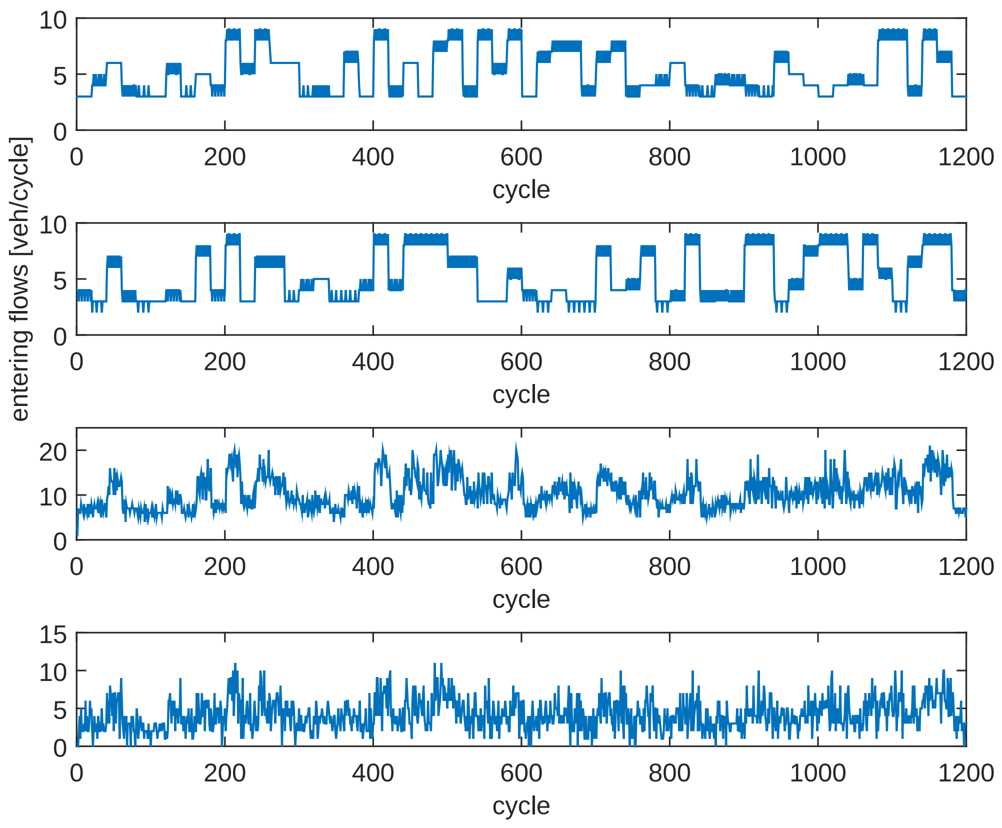

Figure 4-2: The entering flows for the input link used for the identification of the parameters.

	l_{veh}	μ	v_{free}	C	a_{acc}	a_{dec}	v_{low}
only TTS	9.99	0.37	8.54	74.14	/	/	/
only TE	9.98	0.38	14.53	65.45	2.33	-2.50	1.44
only NO_x	9.98	0.37	12.19	66.96	2.49	-2.60	5.63

Table 4-2: Value of identification parameters found after optimizing different cost functions

To identify these parameters, the SUMO model was run for 1200 cycles to collect the data. After that the model (either only the S-model or in combination with the VT-micro emission model) was given this data, which consists of the the number of vehicles on the link n, the vehicles inside the waiting queue q and the entering vehicle flows for the input links (α_{enter} for link 1, 2, 6 and 7). Using this data, the model was able to calculate the cost function for all the cycles. This model was fed to an optimization algorithm, together with the cost-value gotten from the SUMO traffic simulator. This optimization algorithm tried to find the optimal value for the parameters given above, in order to reduce the error between the cost-value for SUMO and the cost-value obtained from the model.

The SUMO files given by my advisor [3] included code to obtain the value for q, however when the traffic light turned green the queue was set to zero, since the vehicles were moving and thus not waiting any more. This approach does not work for the S-model I was using, therefore the average queue was taken and used as input for the S-model. This also resulted in no longer using the value of q in the cost function for this optimization approach. The identified values of the parameters are given in Table 4-2.

The smooth model, needed for the RPROP algorithm, has an extra parameter for identification, namely the smoothening parameter o. Since the impact of this value on the overall S-model is not as large as the other parameters, only the order of magnitude has been taken into account. To decrease the computational burden of the optimization algorithm, this parameter was identified separately. The identified values of the parameter identification for the smooth model are given in Table 4-3

After testing, values for o which were too large gave error in the code, either by having a negative and infinite flow of leaving vehicles or giving mathematical errors, which most likely were caused by dividing by a very small (almost zero) number. I tested the values of 1, 10 and 100 for the smoothening parameter and only the value of 1 did not gave any problems for the scenario used for identification. It is still possible that due to smoothening the equations some vehicle flows become small negative numbers, however the next time step should include these vehicles again, such that no vehicle is lost.

Table 4-2 and 4-3 show that some of the identification parameter approach the bounds, either the lower or upper bound. The values for the velocity as well as the vehicle length tend to go towards the bounds and depending on the cost function, the value for the free flow velocity

[3]The traffic model for SUMO for the given network was already implemented, this included the code that determined the states n and q from the data available from the SUMO simulation.

RPROP	l_{veh}	μ	v_{free}	C	o
only TTS	5.00	0.37	16.00	40.80	1

Table 4-3: Value of identification parameters found after optimizing for the smooth model

does as well. Although the physical meaning of these parameters is lost after the identification procedure, the bounds are needed to ensure the found optimal values are at least near acceptable values in reality (a velocity of 50 m/s is not reasonable in urban traffic networks, thus should not be included in the search).

4-2 Scenarios

For the case study, four scenarios have been considered. These scenarios vary to congest different parts of the network, such that the effect of the controller can be seen. The SUMO model was run first for a couple of cycles (31 cycles for the first scenario and 14 cycles for the other three scenarios) in order to create some traffic on the links. The number of cycles the simulation is run without control depends on the amount of time is needed to create enough congestion with the current vehicle flow inputs as well as the green times given before control. To ensure some of the links become congested the green times are set such that at least one of the two links of the intersection are congested.

One of the main problems of using this network resides in the fact that the links which are not input links can be emptied within a few cycles using the standard control of half the cycle time for both of the links. After checking the SUMO simulation, it appeared as though twenty vehicles can pass through a traffic light for a given cycle time. For the S-model this is dependent on the saturated vehicle flow rate μ which is roughly 0.37 [veh/s], which is about 22 vehicles for a given cycle. Considering only 60% of the vehicles go from link 3 to link 5 and 40% from link 8 to link 9, the number of vehicles reaching the second controlled intersection (link 5 and 9 flowing towards link 11) can be removed within a few cycles.
Due to this characteristic of the network, it is logical to let the scenarios be run for 30 control sampling cycles. After this time interval, the congestion is mostly gone at the intersection near the end of the network. The input links (at least some of them) are over-congested in most of the scenarios such that control is always needed.

The entering flows for the scenarios are given in Figures 4-3 to 4-6. The green times of the uncontrolled cycles were set to improve the congestion on some of the links. For each scenario the network in SUMO will be described before control starts (for reference check Figure 4-1):
For scenario 1, link 1 and 5 are fully congested, link 2 and 6 are filled with vehicles till around halfway through the link and link 7 and 9 are almost empty.
The second scenario fully congests links 1, 2 and 6. link 7 is almost empty. Further in the system link 5 is congested where as link 9 is almost empty.
The third scenario fully congests links 2, 6 and 7. Link 1 is almost empty. Further in the system link 9 is congested where as link 5 is almost empty.
The fourth and final scenario makes links 1 and 9 filled till halfway through the link. Link 6 has a few vehicles, almost a quarter of the link where the rest is empty. The overall demand of this scenario is higher such that near the end the links will be filled up more.
Link 3 and link 8 cannot be congested without fully congest either link 5 or link 9 respectively. The output links (4, 10 and 11) cannot be congested in the current model, as all vehicles that enter can immediately leave the link after entering the waiting queue.

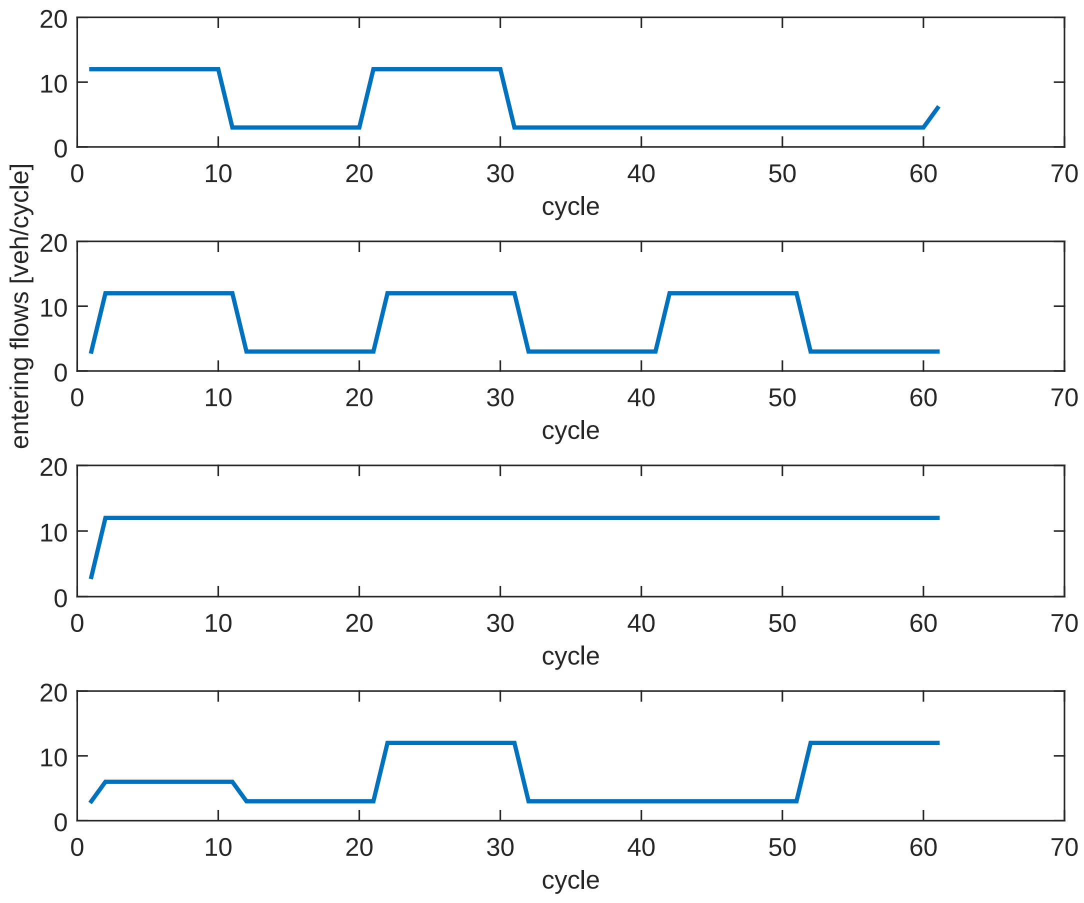

Figure 4-3: The demand profiles for the first scenario

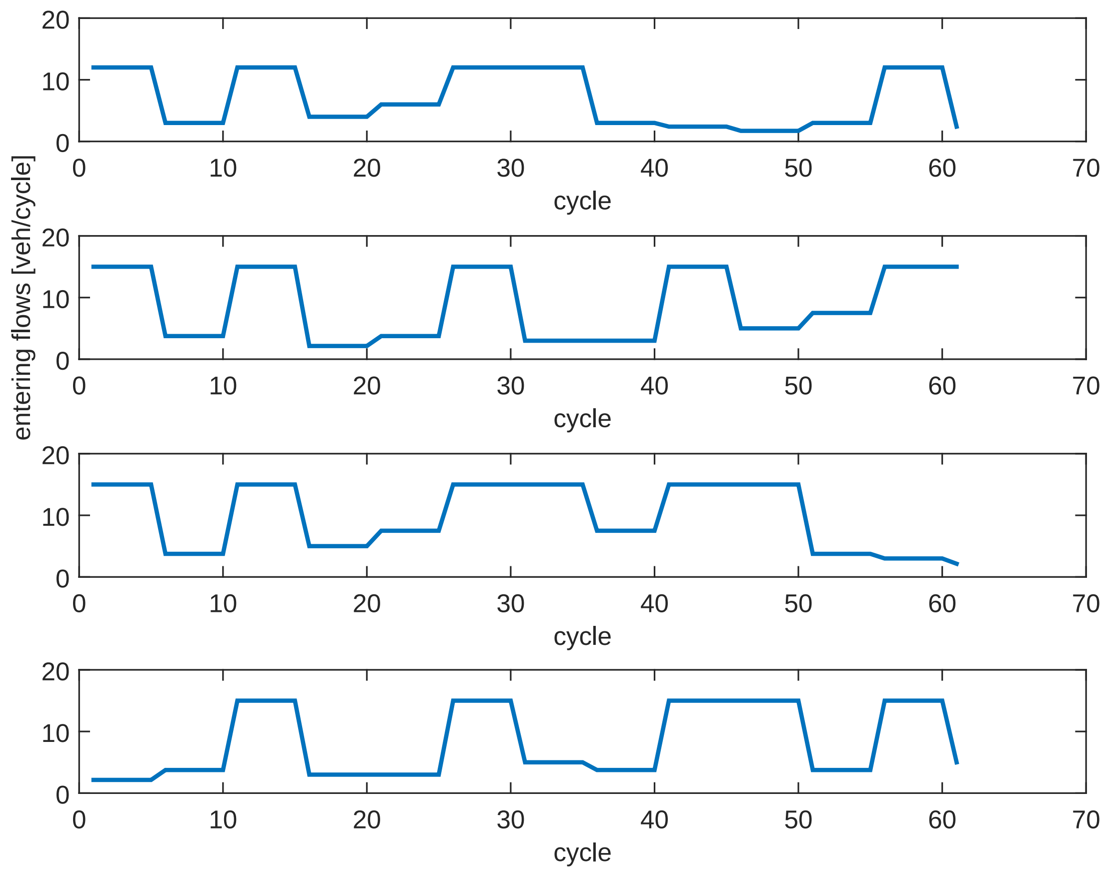

Figure 4-4: The demand profiles for the second scenario

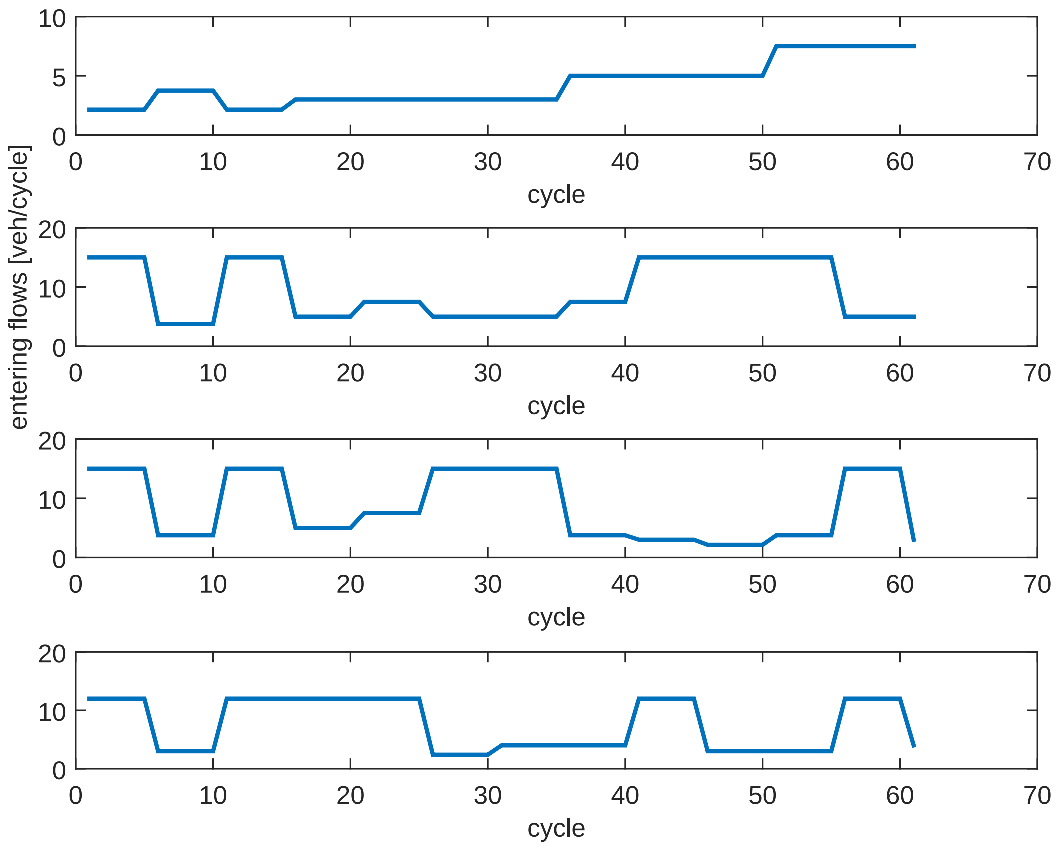

Figure 4-5: The demand profiles for the third scenario

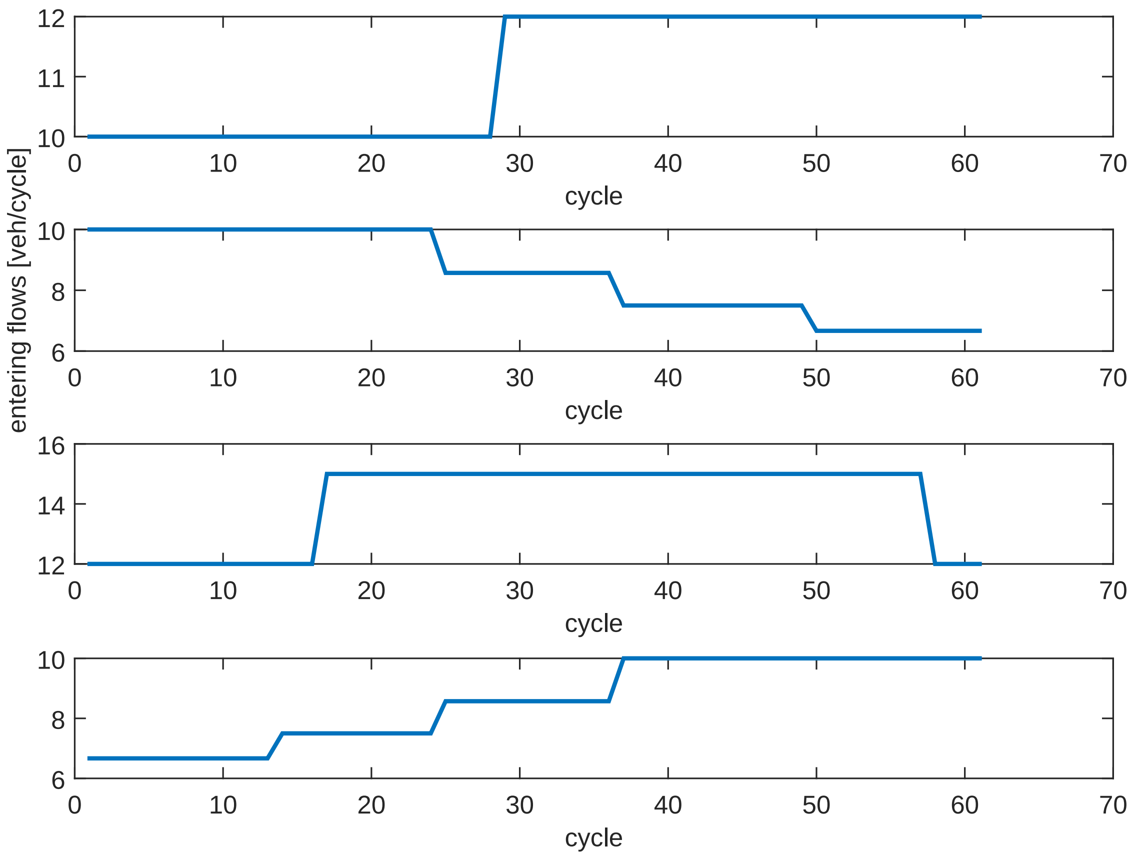

Figure 4-6: The demand profiles for the fourth scenario

4-3 Adaptations for performance improvement

When testing the models (S-model and VT-micro emission model), some errors occurred in the optimization algorithms but also some non-intentional behaviour was spotted. Therefore some adjustment or adaptations were made to the model(s) to improve the models and/or optimization.

Since the input links will be over-saturated, overflow might happen as the entering flow added to the existing vehicles on a link might exceed the capacity. To prevent this from happening, origin queues can be added, virtual links which an infinite capacity which stores these vehicles. The vehicle flow leaving the origin queues will be the same as for the other links given by (2-5), which contains the capacity of the next link into consideration.
Based on the experiments done for this thesis, we observed that for relatively smaller prediction horizons (in this case smaller than five), considering the origin queues did not make a big difference. The main reason is the inability of the model of removing the entire congestion on a link together with the entering flow of vehicles in the given prediction horizon of only five cycles. However, we recommend considering the origin queues for longer prediction horizons (at least for the models used in this thesis).

The next adaptation was made regarding the arriving flow rates for the S-model given by (2-4). The leaving flow rates given by (2-5) depend on q and α_{arrive}, but not on n. This caused the optimization algorithm to optimize the leaving flow rates based on the entering flow rates and the queue, but did not take the other vehicles on the link into consideration.

To solve this issue, the formula for the arriving vehicle flow (2-4) was altered. From the definition, the number of vehicles on a link n at a given time step, consist of the vehicles inside the waiting queue q and the vehicles that arrive at the end of this waiting queue α_{arrive} within one cycle time. This implies that the vehicles that entered the link in the previous time step and did not reach the end of the waiting queue are the remaining vehicles still on the link aside from the vehicles inside this waiting queue, which is equal to $n - q$. The total arriving flow for the current cycle consists of the vehicles still on the link and travelling towards the end of the waiting queue $(n - q)$ as well as the number of vehicles that enter the link in the current cycle and can reach the end of the waiting queue within that same cycle. To get vehicle flows, the value of $(n - q)$ should be divided by the cycle time. Therefore the equation for the arriving flow of vehicles was modified as follows:

$$\alpha_{\text{arrive}}(k) = \frac{t_{\text{cycle}} - \tau(k)}{t_{\text{cycle}}} \cdot \alpha_{\text{enter}}(k) + \frac{(n(k) - q(k))}{t_{\text{cycle}}} \tag{4-1}$$

This alteration to the arriving flow formula made the arriving flow equation dependent on n, and since the value for the arriving flow is used within the leaving flow equation, made this dependent on n as well. After one time step the old formula for the arriving flows will be equal to this new equation, since the states, both n and q will be updated accordingly. The RPROP algorithm did not function properly without this addition, since the values for all the variables need to match or the approach based on derivatives do not work. Therefore equation (4-1) was used for all the models used in the optimization algorithms. Since the S-model was changed, re-identification on the model parameters should have been done, however do to time limitations was not.

Regarding the arriving flow a different problem was seen during the simulations. The model assumes that all the arriving vehicles are present inside the waiting queue when the light turns green, which is not the case. Recall from (2-5) that the leaving flow consists of a term that contains the arriving vehicles and the vehicles inside the waiting queue. This could result in a higher number of vehicles that can pass the intersection if the light is put to green than the number that will leave in reality. The system is trying to remove vehicles from that link that are still not in the queue and will not arrive before the light turns red. This issue needs to be tackled in future work.

The last adaptation needed is for the formulation of the cost function. Currently the TTS is used, which is the sum of all the vehicles on all the links combined multiplied by the cycle time. The main issue with formulating the cost function in this way occurs for the congested scenarios. Independent of the green time, the number of vehicles leaving towards the next link will be equal to the saturated leaving flow rate μ, assuming no congestion on the next link. Therefore the green time does not affect the TTS as the number of vehicles flowing will always be equal to this saturated leaving flow rate, independent of from which link these vehicles leave.
During the simulations the optimization algorithms prioritise removing all vehicles from one of the links first, after which the control input will have an effect again on the TTS. This implies one link will congest even further, which could propagate back towards the previous link. When this happens the green time will also have an effect on the TTS. However at this point the optimization algorithm is already too late to react.
Therefore a new cost function is suggested, which still tries to reduce the number of vehicles inside the network but tries to empty out the congested links more evenly. By first squaring the value of the number of vehicles on a link and then summing all these values, the importance is shifted to emptying out the individual links rather than emptying out the overall network. This implies using altering the cost function from:

$$\text{cost} = t_{\text{cycle}} \cdot \sum_{k=1}^{\text{N}_p} \sum_{i=1}^{11} n_i(k) \tag{4-2}$$

to the following cost function:

$$\text{cost} = \sum_{k=1}^{\text{N}_p} \sum_{i=1}^{11} (n_i(k)^2) \tag{4-3}$$

Since this is no longer related to the TTS, the multiplication with the cycle time loses its value and as such can be removed. Table 4-4 shows that depending on the number of vehicles that are on the links, the value for the TTS cost function does not change. Therefore the green time does have any effect on over-congested links, as the TTS does not change unless one of those links is almost empty. The new cost function does however change depending on the load of the links, which will try to even out the number of vehicles on all the links (only two for this thesis) at the same intersection. This problem can be neglected for the global optimization algorithms by using MPC with a large enough control horizon. Methods, such as

$n_1(k)$	$n_2(k)$	cost function value old (TTS)	cost function value new
10	50	216000	2600
20	40	216000	2000
30	30	216000	1800

Table 4-4: Values of the cost functions based on the load of the links

RPROP, that use derivatives tend to get stuck on these problems, even with the help of MPC.

4-4 Results

Simulations were run for the different optimization algorithms and each scenario, depending on the algorithm used, other options may have been altered and tested as well. These changes will be discussed for the given algorithm and why they were changed.
As previously stated, local optimization algorithms need to be run from multiple starting point to find several local minimums. After finding these minimums, the minimum with the lowest value will be selected as the (sub)optimal solution for the given optimization problem. Even global optimization algorithms, although being able to find multiple minimums, can have increased performance for using multiple starting points for a single iteration. The use of multiple starting points will be referred to as using a multi-start (approach) from this point forward.
For each of the algorithms the use of a multi-start approach has been simulated to compare with the scenario run without this multi-start. For multi-start, only twenty different starting points have been taken, to keep the computation time low. This is not an accurate means to test the overall effectiveness of the multi-start approach. However, during the simulations, the minimums seemed to no be far apart. A better representation to check the effect of multi-start approaches on this model has been discussed in chapter 5 in the future work.
The results of the simulations are given for all the algorithms, in Table 4-9 and 4-10 and compared in section 4-4-5.

GA	multi-start average time	multi-start maximum time	no multi-start average time	no multi-start maximum time
scenario 1	48.39	55.19	2.36	2.57
scenario 2	66.19	73.83	3.05	3.81
scenario 3	47.55	51.78	2.28	2.52
scenario 4	47.71	50.07	2.37	3.98

Table 4-5: The average and maximum computation time needed to compute the solution for GA to the optimization problem for each scenario

4-4-1 Genentic Algorithm (GA)

After running the four scenarios for the four different optimization algorithms, the following results were found for GA. Figure 4-7 shows the realized value of the TTS for the four traffic scenarios considering both single and multiple starting points. Note that the multiple starting points have been generated in a random way, which might have an influence on our relatively small amount of multiple starting points of twenty.
The starting population for GA in these experiments was set to 50, which were already randomly selected. Therefore an assumption was made that the multi-start approach would have a small effect on the overall value of the TTS. From Figure 4-7 the difference between the performance of GA with and without multiple starting points seems negligible. The fact the algorithm without a multi-start approach performs better in some cases is because of a better starting point and Random Number Generator (RNG) compared to the algorithm with a multi-start approach. Taking more starting points for the multi-start should even out the RNG and give similar or better results.
Considering the negligible difference in TTS value, the computation time is the next factor to investigate. As expected, running the algorithms twenty times will also increase the computation time by at least the same amount. Given the results obtained from these four scenarios, a multi-start approach of twenty starting values is not needed for GA if computation time is of any concern. Simulations that run for a longer time period in combination with a larger number of starting points for each iteration might create larger differences in performance, but these effects need to be investigated in future work.

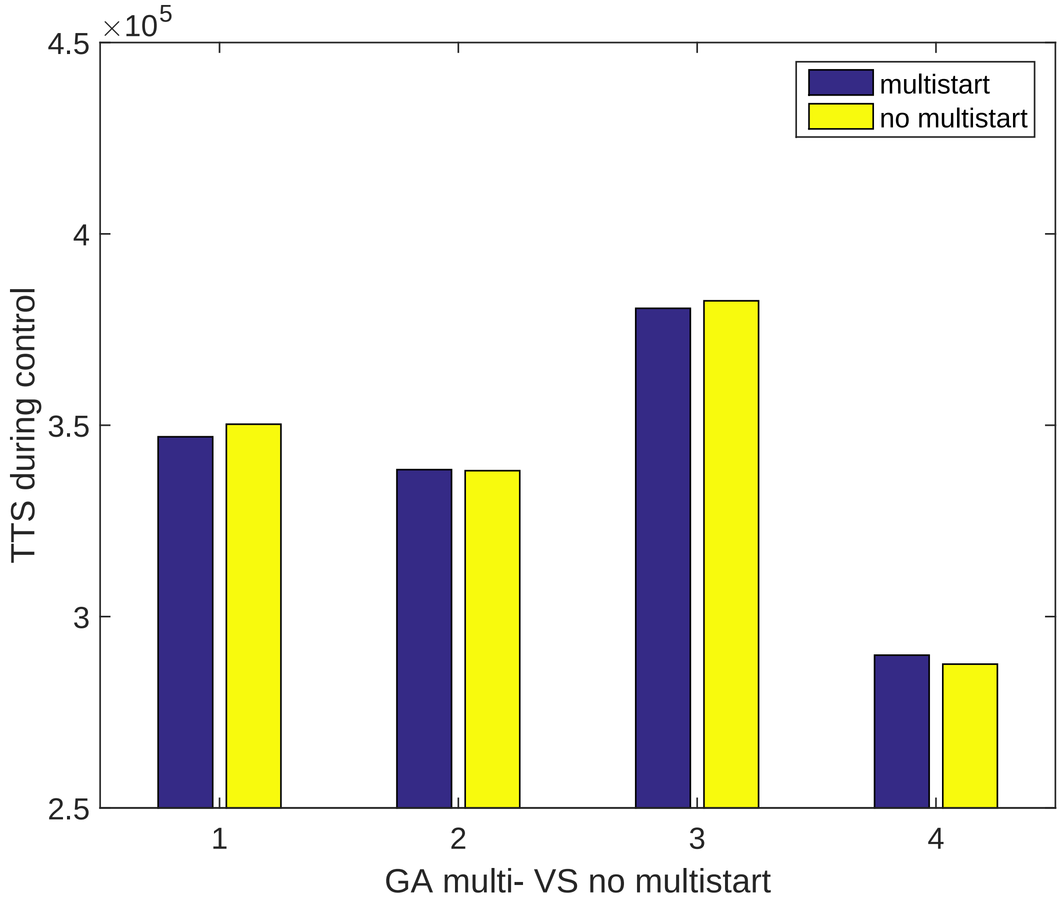

Figure 4-7: The difference in performance for GA with and without a multi-start approach for the different scenarios

	multi-start		no multi-start 1e-3		no multi-start 1e-1	
SA	average	maximum	average	maximum	average	maximum
scenario 1	741.64	762.00	62.57	75.53	37.71	46.91
scenario 2	766.39	812.78	71.90	83.20	45.51	50.82
scenario 3	744.58	777.02	50.51	75.14	37.25	39.01
scenario 4	785.61	847.41	70.72	85.09	38.59	42.82

Table 4-6: The average and maximum computation time needed to compute the solution for the SA algorithm to the optimization problem for each scenario

4-4-2 Simulated Annealing (SA)

The results for the realized value of the TTS for SA are shown in Figure 4-8. In our experiments, SA showed to be the slowest algorithm. To reduce the computation time of the algorithm, two alterations were made. The first was the increase of the re-annealing time from 100 to 300 iterations. This gave the algorithm more iterations to narrow down the solution and trigger the stopping criteria before the step-size would be increased again from the re-annealing. The second alteration was the increase in the stopping criteria on the change in function value from $1 \cdot 10^{-6}$ to either $1 \cdot 10^{-3}$ or $1 \cdot 10^{-1}$. A higher stopping criteria will be reached faster, which terminates the optimization, reducing the computation time. The multi-start approach was run only with the stopping criteria of $1 \cdot 10^{-1}$.

From Figure 4-8, it can be seen that the multi-start approach performs best for all scenarios, although depending on the scenario, the differences between using or not using the multi-start approach can be neglected (around 1-2% between multi-start and no multi-start).
The main problem of this algorithm resides in the computation time, which was the highest of all the optimization algorithms tested. Table 4-6 lists the computation time for the SA simulations run. The algorithm which used the multi-start approach, took over an hour each cycle to compute the (sub)optimal solution. Considering real-time applications, this cannot be implemented for the computation time is too large to compute the solution in time. The performance of the simulations without the multi-start approach is worse, however the computation time might be decent enough to be used in real-time applications (assuming an optimized process on a chip works faster than the computations in Matlab). Considering the negligible differences between the tested options for the SA algorithm, we would recommend to implement the fastest option for real-time application.

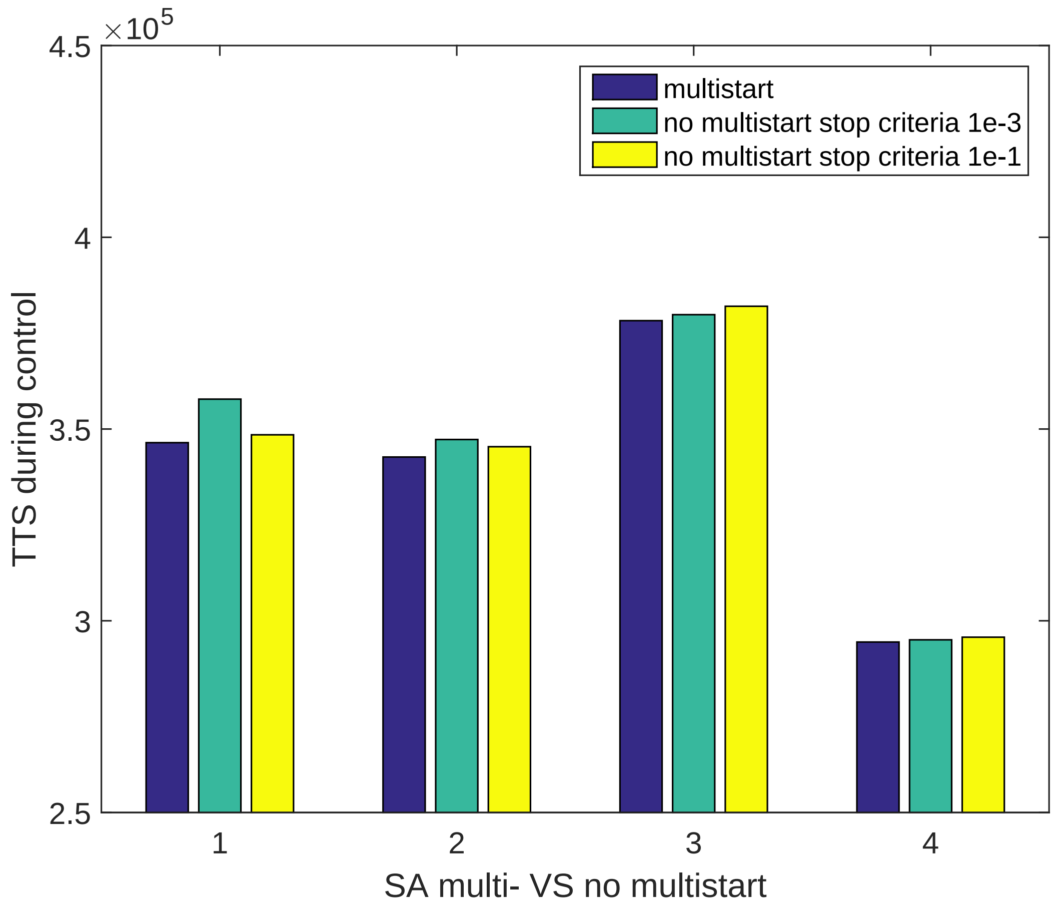

Figure 4-8: The difference in performance for SA with and without a multi-start approach for the different scenarios

	multi-start full		no multi-start full		multi-start no full		no multi-start no full	
PS	average	maximum	average	maximum	average	maximum	average	maximum
scenario 1	22.89	51.70	1.21	3.15	17.59	33.72	0.83	1.42
scenario 2	26.45	38.85	1.56	5.01	21.25	43.52	1.12	2.67
scenario 3	11.19	20.22	0.53	1.49	9.33	20.25	0.42	0.98
scenario 4	21.00	31.46	1.09	1.94	15.70	23.23	0.86	1.52

Table 4-7: The average and maximum computation time needed to compute the solution for the PS algorithm to the optimization problem for each scenario

4-4-3 Pattern search (PS)

The results for the realized value of the TTS for the PS algorithm are shown in Figure 4-9. For PS, the difference between a multi-start approach and only one starting point was also investigated. Since the accuracy for not using a multi-start approach can drop, the option to do a "full search" was also included in the simulations. When "full search" is being used, the PS will check all the changes to all the variables, after which the lowest value for the realisation of the cost function is chosen. Without this option, PS checks the variables until a lower value is found when compared to the previous cost value. When found, the iteration is stopped immediately and the next iteration starts. Since only fifteen control variables are available in this thesis, a "full search" was assumed to take less time than using a multi-start approach, but the effect on the realisation of the TTS value had to be investigated further.

From Figure 4-9, the difference between using using a multi-start approach or not is again negligible. The difference between using a "full search" can be neglected, as there is no viable improvement from these four scenarios that were tested.
Table 4-7 shows the computation time for the PS algorithms that were tested. As seen for the other optimization algorithms, the use of a multi-start approach increases the overall computation time, in this thesis by at least twenty times. The use of a "full search" does add computation time, but significantly less than the use of the multi-start approach. One should note however that only fifteen optimization variables are used in these scenarios, when the number of variables increases, so will the added computation time for using a "full search".

The overall conclusion for the PS algorithm When considering both the value for the TTS as well as the computation time, the PS algorithm should be run without a multi-start approach and, depending on the maximum computation time allowed for real-time applications, can either be used with or without the "full search" although this option does not seem to have any significant value.

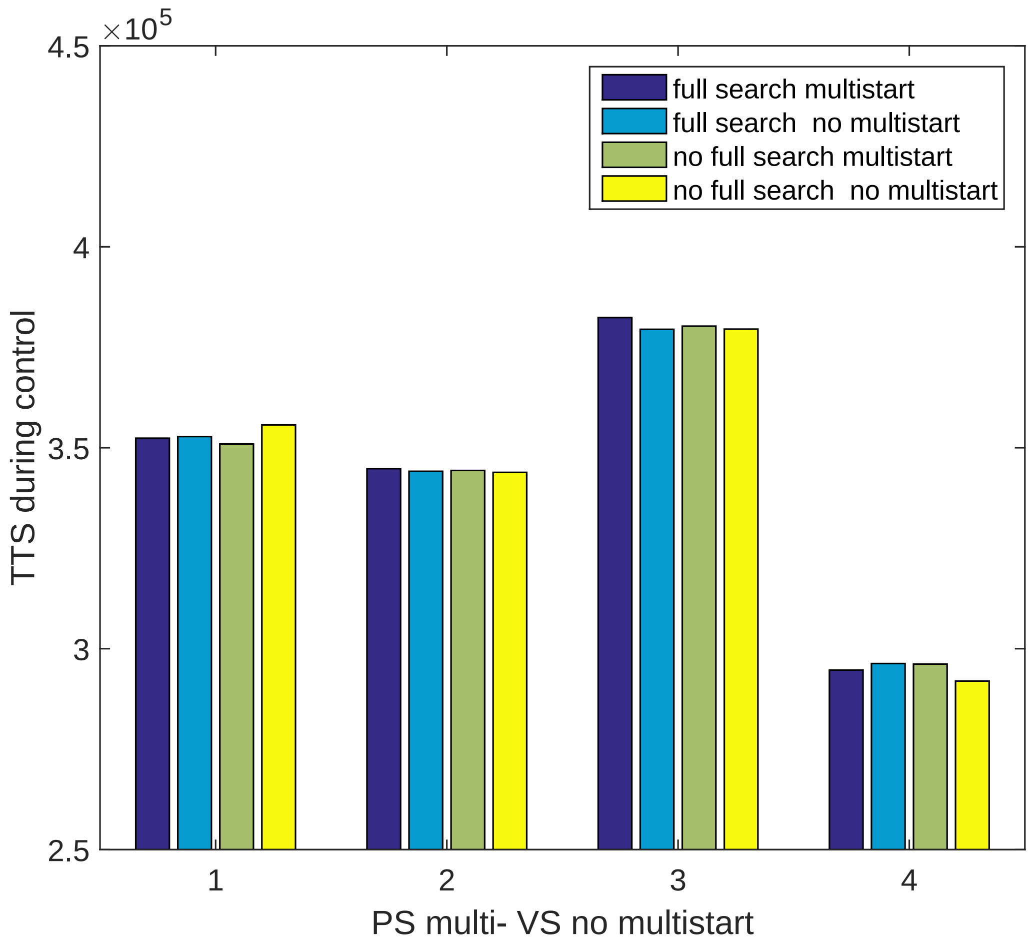

Figure 4-9: The difference in performance for PS with and without a multi-start approach for the different scenarios

	multi-start new cost		no multi-start new cost		multi-start old cost		no multi-start old cost	
RPROP	average	maximum	average	maximum	average	maximum	average	maximum
scenario 1	126.30	286.73	5.68	12.45	52.62	128.91	2.05	5.87
scenario 2	120.10	211.74	4.62	7.63	57.94	101.29	2.89	6.94
scenario 3	111.25	160.61	4.82	14.33	77.00	117.78	3.08	8.63
scenario 4	133.10	150.15	5.26	11.99	28.12	40.57	1.05	2.07

Table 4-8: The average and maximum computation time needed to compute the solution for the RPROP algorithm to the optimization problem for each scenario

4-4-4 RPROP

The results for the realized value of the TTS for the RPROP algorithm are shown in Figure 4-10. As already stated in the section about adaptations, the TTS cost function does not work as good for a derivative based approach such as the RPROP method. Especially in the fourth scenario, the last intersection became congested, causing link 5 to fully congest before the controller tried to empty that link again. Therefore, the RPROP algorithm was also run with the new cost function that was discussed in the adaptations section. To compare both cost functions, after the simulation, the definition for the TTS was used on the state variable n such that both have a similar cost value.
After checking the simulation data in SUMO, the original cost function (TTS) congested link 5, where as the new cost function did not. The main difference in performance in the first three scenarios can be explained by looking at the input links. In those scenarios, at least one of the input links is congested past the capacity. As stated, no input queues were added to the model. This causes the new cost function to remove the vehicles from the over-congested link, which afterwards just fills back up. The original cost function will try to empty out the other link first, after which control will have effect again on the TTS value, ultimately causing less vehicles to be counted. By adding input queues we would expect to get somewhat similar results for both cost functions, at least for the first three scenarios. The last scenario did not have over-congested links, which caused the new cost function to perform better than the old cost function.

Figure 4-10 shows quite varying results. The difference between the original (old) cost function and new cost function was just explained, however new simulations with input queues should provide new data before conclusions can be drawn from the data. The best comparison between the two cost functions would be scenario 4, which does not have over-congested links, however this is only one scenario so no conclusions can be made from this.
The value for the cost function varies heavily between using and not using a multi-start approach, however it varies in both ways. For the original cost function, the multi-start approach seems to be needed, looking at scenario 1 and 4, however the number of initial points should be increased when looking at scenario 3. For the new cost function, the multi-start approach seems to perform worse than not having a multi-start, suggesting the number of initial points needs to be increased for the multi-start approach.

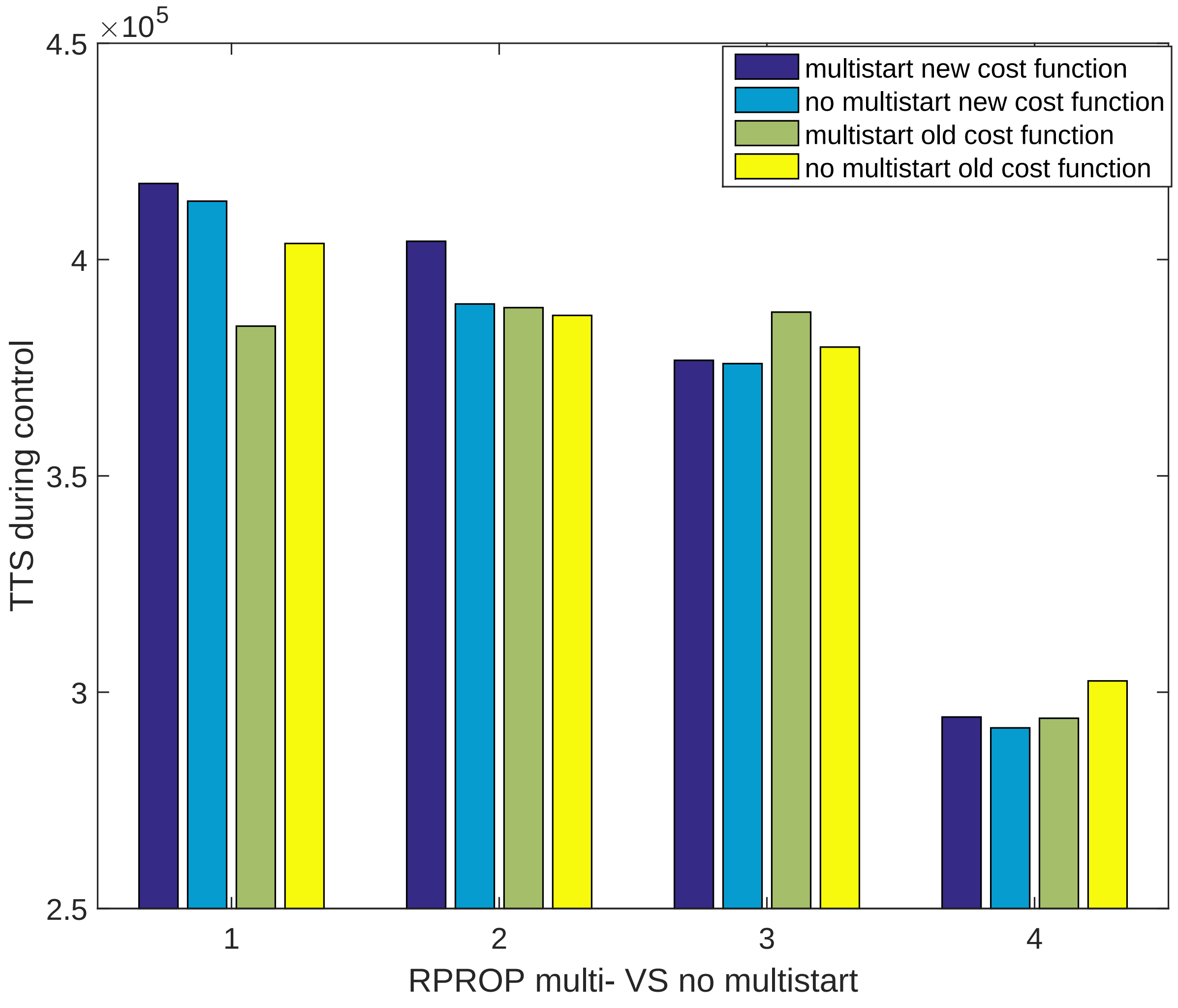

Figure 4-10: The difference in performance for RPROP with and without a multi-start approach for the different scenarios

4-4-5 Comparison between the algorithms

The discussion given below are based on the results obtained with and without multiple starting points for all the algorithms. As shown in Figure 4-11 and 4-12, all the optimization based algorithms perform better than the fixed time (FT) controller. The RPROP algorithm uses the original cost function to make sure the TTS can be compared to the other optimization algorithms, which use the same cost function.

Looking at Figure 4-11 first, no multi-start approach is used. GA is performing best for scenario 2 and 4, giving similar results compared to the other global optimization algorithms in the other two scenarios. SA performs best in scenario 1 and for scenario 3 both PS and RPROP give the best response, with only minor difference between the two algorithms. Overall, the GA is performing best. The reason for this resides in the initial population of 50 for the GA, which already has a good spread over the domain compared to the other starting points used in the other optimization algorithms. The SA algorithm has the "re-annealing" inside its algorithm, which increases the step size after a few iterations. This can lead to escaping local minimums and finding a different minimum. PS has no build in algorithm to escape from local minimums, however if the step size is large enough and the minimums close to each other it might move between different local minimums.
As discussed in the RPROP method section, the chance of ending in a suboptimal minimum is quite large, as it does not empty the fullest link, but rather just empties out one link while the other gets congested. This is caused by the derivatives which are calculated, as some are quite close to or even equal to zero on some intervals of the function, rather than in one point. When the link is almost congested the third term of equation (2-5) will become visible in the derivative. This term does not appear when the link is only halfway congested, as the derivatives are created for a local point. When less congestion is available, such as in scenario 3 and 4, the RPROP method performs quite similar compared to the other methods. This is the main reason the RPROP method performs poorly compared to the other three optimization algorithms, although it can achieve similar results depending on the scenario.

Figure 4-12 shows the results for using a multi-start approach for the given optimization algorithms used in this thesis. A similar shape can be seen when compared to Figure 4-11. The values for the TTS are listed in Table 4-9. Below each scenario the percentage increase or decrease will be listed compared to the TTS of the GA. The GA has been taken as a bench mark since it performed best in most scenarios.
Table 4-9 shows that GA performs best for scenario 2 and 4 again, with SA performing only slightly better for the other two scenarios. Where the RPROP method was one of the better performing algorithms for scenario 3 without the use of multi-start, it is quite far above (1.9%) the other methods used. The results for the fourth scenario however shows that the used of a multi-start approach made the RPROP perform similar when compared to the SA and PS algorithms. The RPROP method still relies on the local derivatives which cause it to function poorly compared to the other optimization algorithms. A suggestion was already made for a new cost function which ensures links to be emptied out evenly so over-congestion will not happen. However since this cost function was not needed for the other methods, the original cost function was used, knowing the limitations it would cause the RPROP method.

TTS with multi-start($\cdot 1e5$)	FT	GA	SA	PS	RPROP
scenario 1	4.92	3.47	3.46	3.52	3.85
	41.7%	0%	-0.2%	1.6%	10.9%
scenario 2	4.89	3.38	3.43	3.45	3.89
	44.5%	0%	1.3%	1.9%	14.9%
scenario 3	4.00	3.81	3.78	3.82	3.88
	5.1%	0%	-0.6%	0.5%	1.9%
scenario 4	3.75	2.90	2.94	2.95	2.94
	29.4%	0%	1.6%	1.6%	1.4%

Table 4-9: The average and maximum computation time needed to compute the solution for the RPROP algorithm to the optimization problem for each scenario

Table 4-10 shows the average time that was needed for the algorithms to compute the (sub)optimal solution for a single cycle. Since the FT controller only implement a fixed ratio based on the cycle time to each traffic light, it has no calculations and the time needed was set to zero. The pattern search seems the fastest, completing the computations in under half a minute in all scenarios on average. The GA is second with roughly a minute for each cycle, as is the RPROP method, although varying results are seen for RPROP. Since the RPROP algorithm was build from scratch, no optimization was done on the code based on lowering the computation time of the algorithm. Optimizing the code could lead in an overall reduction of computation time, but will probably not explain the variation in performance for the given scenarios. The SA is performing poorly based on computation time, even with the reduction of the stopping criteria, taking well over an hour to complete.

Based on the data obtained from these simulations, one might conclude that the global optimization algorithms perform better than the current implementation of the RPROP algorithm used in this thesis. Overall the GA performs best on reducing the cost function, followed by SA and then PS. Based on computation time, PS performs outstanding, followed by GA and RPROP. SA cannot be used in real-time applications based on the high computation time and should be avoided. Given the current data obtained, we recommend the GA for lowering the cost function in all cases. As Figure 4-7 shows, a multi-start approach is not mandatory for GA, which keeps the overall computation time low enough for real-time applications.

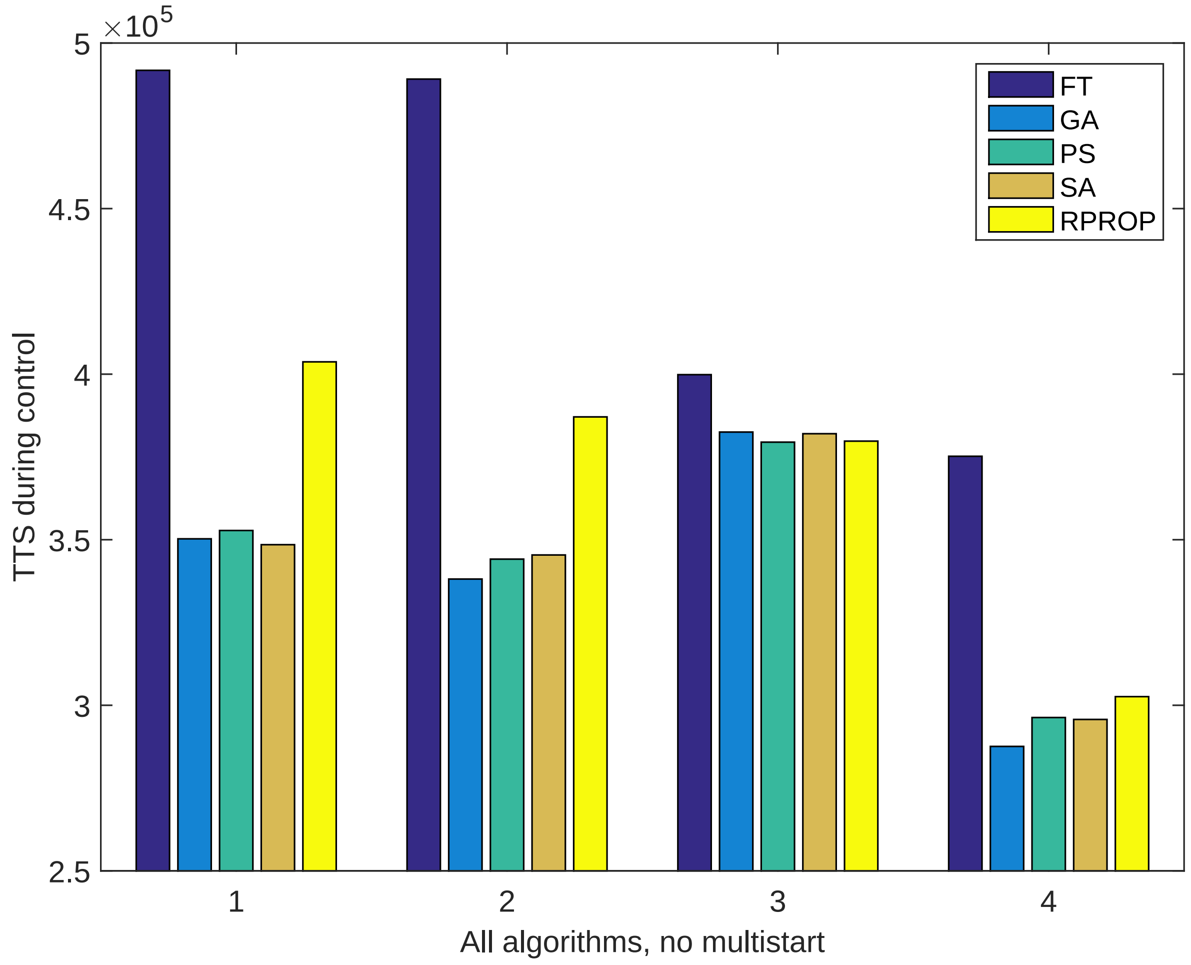

Figure 4-11: The difference between algorithms without the use of a multi-start approach

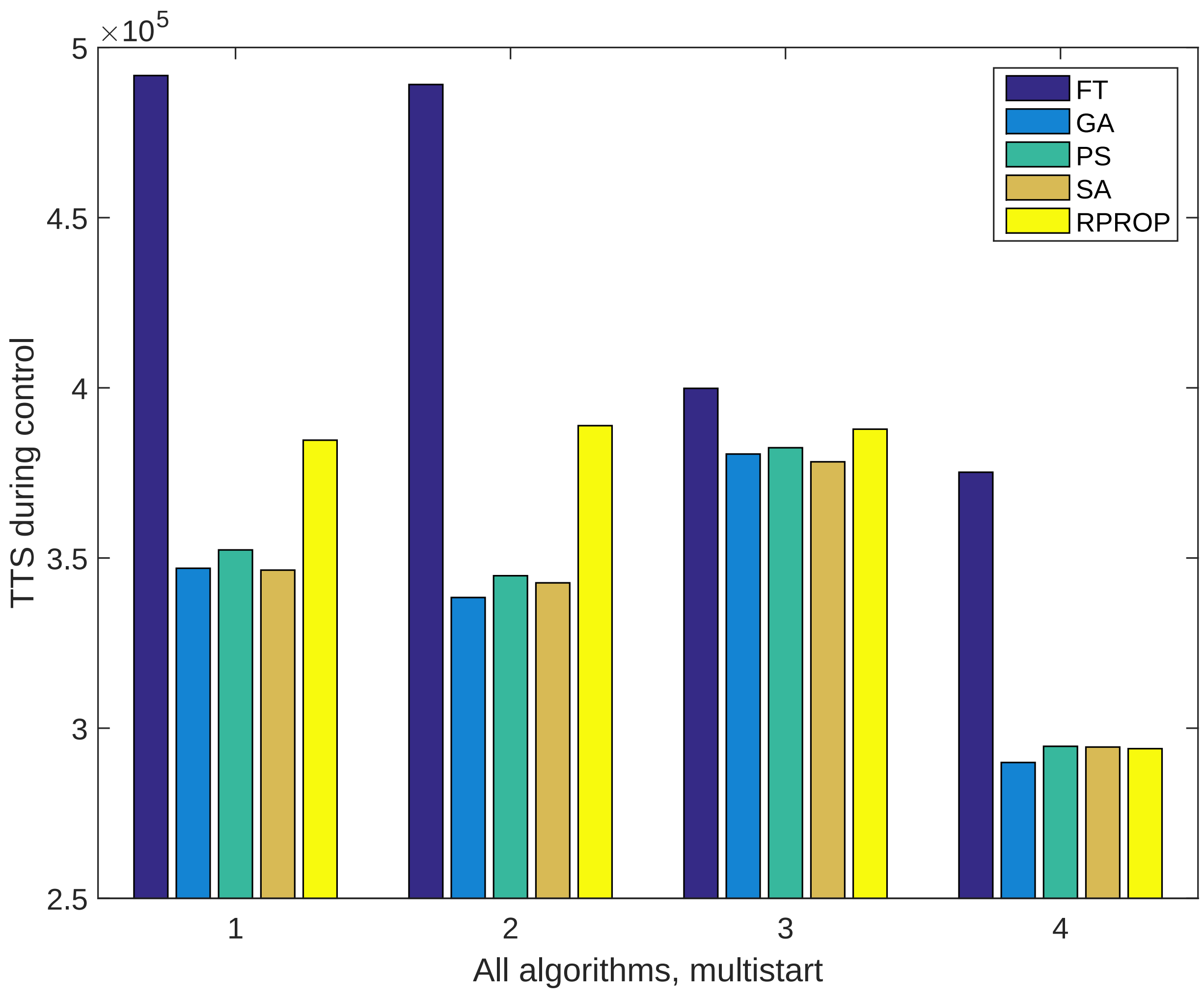

Figure 4-12: The difference between algorithms with a multi-start approach

computation time [s] with multi-start	FT	GA	SA	PS	RPROP
scenario 1	0	48.39	741.64	22.89	52.62
	-100%	0%	1432.5%	-52.7%	8.7%
scenario 2	0	66.19	766.39	26.45	57.94
	-100%	0%	1057.8%	-60.0%	-12.5%
scenario 3	0	47.55	744.58	11.19	77.00
	-100%	0%	1465.9%	-76.5%	61.9%
scenario 4	0	47.71	785.61	21.00	28.12
	-100%	0%	1546.7%	-56.0%	-41.1%

Table 4-10: The average and maximum computation time needed to compute the solution for the RPROP algorithm to the optimization problem for each scenario

Chapter 5

Conclusions and recommendations

Given the simulations and the results discussed in the previous chapter the following conclusions can be drawn from the work done in this thesis. Moreover, recommendations proposed for future work are included in a separate section.

5-1 Thesis results

The main contribution of this thesis work was the use of the RPROP method combined with the S-model. This model uses derivatives to compute the (sub)optimal value for the optimization problem. These derivatives are local, which can ensure that some characteristics of the model further away are not included in this local derivative. This may lead to finding only a local minimum which can be far apart from the global minimum. In the simulations, congested intersections were often optimized for a local minimum and the global minimum could not be reached.
In the uncongested case, most vehicles arriving at the intersection could leave with an even distribution for the green times. This caused control to have no effect and creating similar results by using a Fixed Time (FT) controller. Therefore only scenarios which have congestion have been simulated. If the intersections were congested, but over time less vehicles arrived than vehicles would leave, a solution could be found to reduce the vehicles of both incoming links of the intersection.
This issue was mostly caused by the cost function that was being optimized. The sum of the number of vehicles on all the links multiplied by the cycle time was equal to the Total Time Spent (TTS). This cost function however sums up all vehicles, independent of on which link the vehicles are located. If the intersection was fully congested all optimization algorithms would try to empty one of the links, such that afterwards controlling the green times would have an effect.

During simulations an observation was made that having a multi-start approach would not benefit the optimization algorithm much. To prove this, a 20 point multi-start approach was compared to not using a multi-start approach at all. After observing the results of the simulation, one could say that for 20 multiple starting points the results are roughly equal, but not similar. Although 20 starting points is quite low for a multi-start approach, the results of a single starting point would sometimes be better. This can be explained by a better starting point or different random numbers being used in the algorithm. Choosing 200 or even 1000 starting points should create a similar or better performing control sequence, but these might exceed the computation time span allowed for real-time control. In the interest of real-time control, a trade-off has to be made between good control performance and computation time. Given the results of the simulations, not having a multi-start approach is performing good for the global optimization algorithms when compared to having a 20 multi-start approach of the same algorithm. As discussed, without the multi-start approach the RPROP method is not functioning that well in removing vehicles from the network and it will often need the use of a multi-start approach given the model and cost function used in this thesis.

From the results for the multi-start approach the following conclusion can be drawn. Over the four scenarios which were being run, the Genetic algorithm (GA) performed best in reducing the TTS for the multi-start approach. The Simulated Annealing (SA) algorithm performed slightly worse in half of the cases and slightly better in the other half, but the overall difference is a few percent at best. SA performs similar to the Pattern Search (PS) algorithm and performs slightly worse than GA. The RPROP method performance shows different performance depending on the congestion on the traffic network. With light congestion its performs on par with the other methods where for the first two scenarios the performance was respectably 11% and 15% worse than the GA. All methods are still performing better than the Fixed Time (FT) controller, which had a TTS value over 40% larger than the GA in some of the scenarios.

The results for the computation time show different results. The GA took twice the amount of time the PS algorithm needed, if not more. The RPROP method had varying results compared to the GA, which should be reinvestigated after the proposed fixes have been applied. The SA algorithm took a long time to compute the results, needing ten to fifteen times the amount of time the GA needed, which was already twice as slow as the PS algorithm.

The results for the scenarios with only one starting point, show similar trends as with multiple starting points. The FT controller is behind in all scenarios with the RPROP methods second to last in all but one of the scenarios. For the computation time, only the time for the SA algorithm is large using Matlab.

The main conclusions, with respect to using a multi-start algorithm, would be:

- GA,PS and SA perform similarly w.r.t. decreasing the cost function
- GA and PS yield low computation times, with RPROP having varying results
- Choose GA for the better performance of the four algorithms
- Choose PS if the computation time is an issue for real-time control

5-2 Topics for future work

In this thesis some adaptations were described but not yet implemented due to time constraints. In the future these changes could be made. A number of these adaptations are mentioned below.

The most important element is adding the emission model with the correct formulas. Reducing emission gasses has become a popular research topic so adding this to the models would be interesting for modern day control. The emission models work with vehicle velocity and acceleration, which are not present in the S-model. Average velocity assumptions will have to be made, further reducing the accuracy of the model. However, it could be interesting to see how vehicle behaviour (slowing down, going from zero to the free-flow velocity) affects the emissions and how this can be controlled. This could work against lowering the TTS and thus a balance/trade-off has to be found between reducing the TTS value and reducing the emissions.

The cost function will need to be adapted or changed. Currently, it will lower the total number of vehicles inside the network but this is independent on the links they are residing on. Lowering the number of vehicles per link and in total would be a better alternative for a cost function. First squaring the value of the number of vehicles for each link and then summing these values up should already give a better result. This should keep the number of vehicles on each link as low as possible or in the case of congestion, will even out the number of vehicles waiting on each incoming link of the intersection.

One should notice that the algorithms are searching for green times expressed as real numbers, where as for a real-time application integers will probably be used. Using integer search is a restriction on the optimization algorithm and can hinder some of the algorithms used in the overall optimization algorithm. However, rounding the values obtained will have its effects on the TTS, most probable increasing its value. If the cost function can be lowered by rounding the values to the nearest integer it would be likely for most of the algorithms to have found this as one of the solutions for the optimization problem. Since integer search can hinder the optimization algorithms, we suggest simulations are run with rounding the value before feeding them back to the SUMO simulator to investigate these effects. Simulations should point out the overall differences between rounding the values obtained up, down or having integer search for both TTS and computation time.

The RPROP method will require some work, most importantly on the code. The code is working but in no means has this been optimized for faster computation speeds. The algorithms for GA, SA and, PS were already implemented in Matlab. They have a range of options and they have been optimized for reduced computation time of the algorithm. The RPROP code could also change such that parameters can be changed more easy such as stopping criteria or number of iterations. Implementing the optimization algorithms on a chip rather than in Matlab will also improve the computation time. Research can be done to see the computation time/span of the optimization algorithms on a chip. This information can be used to find the correct setting (multi-start, stopping criteria, etc.) for each algorithm such that high performance can be obtained within the time span that is allowed for finding the correct control inputs.

Lastly a more in-depth analysis could be done for the multi-start approach. With only 20 starting points the current analysis was a bit limited, but gave some constructive results at least. However, no generalization can be made on this small set of data. A larger number of starting points could be chosen for again four or five scenarios, preferably more.

A larger view for future work can maybe found in in the adaptation of the models (traffic flow model and the emission model) to new developments in the vehicle/automotive industry. Although small now, the shift towards electric vehicles will be made to decrease the emission gasses released into the environment by gasoline and diesel vehicles. By using sustainable energy to charge electric vehicles, the emission gasses released for using the vehicles in transit are removed.
These vehicles still use an energy source, which can be depleted. Therefore the emission model might be replaced by a new model, used to reduce the amount of energy the vehicles use. Vehicle behaviour will be the main topic, as some actions (accelerating, etc.) will take more energy than other actions (standing still). Some electric vehicles can even generate energy by braking, which can be stored inside the battery for later use. One of the main problems might be the ever changing technology for these kind of vehicles, as this kind of technology is still in its infancy.
Over the years, vehicles are equipped with more digital systems and sensors. If the data on these can be obtained and sent to the traffic flow controller, more information can be used for computing an optimal solution. One of the problems in the model is the inaccurate flow of vehicles that enter the waiting queue, since an average of an entire cycle is taken. Having data on vehicle velocities might increase the accuracy of this flow, although the set-up has to be tested to see if the computation time is not an issue. In this thesis all the work was done on a PC, where in real-time parallel processing might be an option to ensure a larger model might work for real-time traffic control.

Bibliography

[1] K. Ahn; H. Rakha; A. Trani; M. Van Aerde. Estimating vehicle fuel consumption and emissions based on instantaneous speed and acceleration levels. *Journal of transportation engineering*, 128(2):182–190, 2002.

[2] N.E. Ligterink; R. de Lange. Refined vehicle and driving-behaviour dependencies in the versit+ emission model, 2009.

[3] US environmental protection agency (EPA). Mobile model (on-road vehicles). https://www3.epa.gov/otaq/mobile.htm, 23-03-2016.

[4] D. Helbing. Gas-kinetic derivation of navier-stokes-like traffic equations. Technical report, Institute of Theoretical Physics, University of Stuttgart, 1998.

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

[6] S. Lin; B. De Schutter; Y. Xi; H. Hellendoorn. Integrated urban traffic control for the reduction of travel delays and emissions. *IEEE transactions on intelligent transportation systems*, 14:1609–1619, 2013.

[7] L. Ingber. Simulated annealing: Practice versus theory. *Elsevier, Mathematical and Computer Modelling*, 18:29–57, 1993.

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

[9] H.Y. Sutarto; R.K. Boel; E. Joelianto. Parameter estimation for stochastic hybrid model applied to urban traffic flow estimation. *The institution of Engineering and Technology*, page 9, 2015.

[10] K. Aboudolas; M. Papageorgiou; E. Kosmatopoulos. Store-and-forward based methods for the signal control problem in large-scale congested urban road networks. *Transportation Research Part C*, 17:163–174, 2009.

[11] H. SchÃďttler; U. Ledzewicz. *Geometric Optimal Control Theory, Methods and Examples.* Springer, 2012.

[12] M. Brackstone; M. McDonald. Car-following:a historical review. *Transportation research part F: traffic psychology and behaviour*, 2:181–196, 1999.

[13] DLR Institute of Transportation Systems. Sumo wiki online documentation (faq). http://sumo.dlr.de/wiki/FAQ#What_kind_of_a_traffic_simulation_is_SUMO.3F, 23-11-2017.

[14] R. Smit; R. Smokers; E. Rabé. A new modelling approach for road traffic emissions: Versit+. *Transportation Research part D: transport and environment*, 12:414–422, 2007.

[15] N. Geroliminis; J. Haddad; M. Ramezani. Optimal perimeter control for two urban regions with macroscopic fundamental diagrams: a model predictive approach. *IEEE transactions on intelligent transportation systems*, 14:348–359, 2013.

[16] L. Ntziachristos; Z. Samaras. Copert iii computer programme to calculate emissions from road transport. Technical report, European Environment Agency, 2000.

[17] L. M. Schmitt. Fundamental study theory of genetic algorithms. *Elsevier, Theoretical computer science*, 259:1–61, 2001.

[18] H. Rakha; K. Ahn; A. Trani. Development of vt-micro model for estimating hot stabilized light duty vehicles and truck emissions. *Transportation Research part D: transport and environment*, 9(1):49–74, 2004.

[19] university of California riverside (UCR). Comprehensive modal emission model (CMEM). http://www.cert.ucr.edu/cmem/, 23-03-2016.

[20] S.K. Zegeye. *Model-Based Traffic Control for sustainable Mobility.* PhD thesis, Delft University of Technology: TU Delft.