Het verkeerskundig
laboratorium
voor studenten

ITS EDU LAB

# Network Performance Degeneration in Dynamic Traffic Assignment

With Applications to Evacuation Modelling

Wouter J. Schakel

August 2009

Rijkswaterstaat

TU Delft

Ministerie van Verkeer en Waterstaat

Rijkswaterstaat

# Network Performance Degeneration in Dynamic Traffic Assignment

**With Applications to Evacuation Modelling**

Wouter J. Schakel

August 2009

**MSc Thesis Transport & Planning**
**Delft University of Technology**

## Colophon

**Published by:** ITS Edulab, Delft
ITS Edulab is a cooperation between the Rijkswaterstaat centre for Transport and Navigation and the Delft University of Technology

**Information:** Henk Taale
Email: Henk.Taale@rws.nl

**Author:** Wouter J. Schakel
Delft University of Technology
Master Transport & Planning

**Graduation committee:**
Prof. Dr. Ir. Serge P. Hoogendoorn
Committee chairman
Delft University of Technology
Faculty of Civil Engineering and Geosciences
Ir. Olga Huibregtse
Delft University of Technology
Faculty of Civil Engineering and Geosciences
Prof. Dr. Ir. John Stoop
Delft University of Technology
Faculty of Aerospace Engineering
Ir. Marco Schreuder
Rijkswaterstaat, Centre for Transport and Navigation
Ir. Ydo de Vries
Rijkswaterstaat, Centre for Transport and Navigation

**Date:** August 2009

**Status:** Final report

## Summary

EVAQ is a traffic model for ex-ante evaluations of evacuations plans. The model is still in development and it is uncertain whether EVAQ will accurately model network performance degeneration. At the same time it can be said that accurate network performance degeneration is very important for evacuations. The research of this thesis identifies that there are phenomena that contribute to network performance degeneration that are not modelled. Two important phenomena that are not included are the flow degeneration as soon as links become congested and the constraints that nodes (intersections) themselves have. Several general ideas were thought up to implement these phenomena. A selection was made on the basis of accuracy.

Flow degeneration as soon as a link becomes congested has to do with the link model. The link model determines what number of vehicles can potentially enter and leave the link within a time step. In order to accurately determine these, the framework of Cell Based Queuing is developed. It represents the queue on a link as a set of cells that are related to successive time steps in the past. The theory of kinematic waves is applied which explains that in congestion the traffic states move upstream. Traffic states in the cells can thus be determined using the link outflow from the past. Link inflow is determined by the remaining storage capacity on the link. As an addition to the theory of kinetic waves, the cell at the end of the link is governed by saturation flow rather than kinematic waves. This implicitly applies a capacity drop.

A newly developed node model evaluates constraints on the nodes. The new node model is a combination of these constraints and the constraints by link inflow that are already evaluated. The node model exists out of several sub models that are used for different node types. The controlled intersection model deals with combined use of conflict areas and the effect of green phases. The uncontrolled intersection model is based on a capacity formula that determines the capacity for a minor flow based on a major flow. The formula is used in a framework that relates all flows on the intersection. For roundabouts an existing model by Cetur (1986) is used. A similar framework is put in place to relate all flows over the roundabout. The model is adapted to work on lane level rather than link level for turbo roundabouts. For weaving sections, on-ramps and off-ramps a new model is developed that looks at lane specific demand.

The new model needs more calculation time but produces more precise capacity estimations. Significant changes are found for the MFD and for queue lengths (spillback). The latter now resembles results from the

microscopic model VISSIM quite closely and needs both the new link and node model.

The new node and link model are part of the Dynamic Network Loading model of EVAQ. This model has been the centre of most changes performed for EVAQ and can be used in any other Dynamic Traffic Assignment model. Furthermore, the model is theory based and can thus be used for reversed engineering and more extensive analysis of bottlenecks, also for evacuation schemes.

## Nederlandse samenvatting

EVAQ is een verkeersmodel voor ex-ante evaluaties van evacuatie plannen. Het model is nog onder ontwikkeling en het is onbekend of EVAQ nauwkeurig Netwerk Prestatie Degeneratie kan modelleren. Tegelijkertijd kan gezegd worden dat nauwkeurige Netwerk Prestatie Degeneratie erg belangrijk is voor evacuaties. Het onderzoek van deze thesis wijst uit dat er fenomenen zijn die bijdrage aan Netwerk Prestatie Degeneratie die niet gemodelleerd worden. Twee belangrijke fenomenen welke ontbreken zijn de degeneratie van doorstroming bij congestie en de randvoorwaarde welke knopen (kruispunten) zelf hebben. Een aantal ideeën is bedacht om de fenomenen te implementeren. Een selectie is gemaakt op bases van nauwkeurigheid.

Degeneratie van doorstroming zodra een link congestie krijgt heeft betrekking op het linkmodel. Het linkmodel bepaalt het aantal voertuigen dat potentieel the link in en uit kan binnen een tijdstap. Om deze aantallen nauwkeurig te bepalen is het raamwerk van CBQ (file gebaseerd op cellen) ontwikkeld. Het representeert the file op een link als een aantal cellen gerelateerd aan opeenvolgende tijdstappen in het verleden. De theorie van kinematische golven, welke verklaart dat verkeersstaten stroomopwaarts verplaatsen, is toegepast. Verkeersstaten in de cellen kunnen dus worden herleid met de link uitstroom uit het verleden. Link instroom wordt bepaald door de overgebleven opbergcapaciteit van de link. Naast de theorie van kinematische golven is de cel aan het einde van de link onderhevig aan saturatie doorstroming. Dit past impliciet een capaciteitsverval toe.

Een nieuw ontwikkeld knoopmodel evalueert randvoorwaarden op de knopen. Het nieuwe knoopmodel is een combinatie van deze randvoorwaarden en de randvoorwaarden van link instroom welke al worden geëvalueerd. Het knoopmodel bestaat uit meerdere submodellen welke voor verschillende knooptypes worden gebruikt. Het model voor kruispunten met verkeerslichten beschouwt het gecombineerde gebruik van conflictgebieden en het effect van groenfasen. Het model voor niet gecontroleerde kruispunten is gebaseerd op een capaciteitsformule welke de capaciteit van een ondergeschikte stroom bepaald aan de hand van een voorrangsstroom. De formule wordt gebruikt in een raamwerk waarin alle stromingen op een kruispunt gerelateerd worden. Voor rotondes wordt een bestaand model van Cetur (1986) gebruikt. Een soortgelijk raamwerk wordt gebruikt om alle stromingen over de rotonde te relateren. Het model is aangepast om op rijstrookniveau in plaats van linkniveau te werken voor turbo rotondes. Een nieuw model is ontwikkeld voor weefvakken, opritten en afritten welke naar rijstrook specifieke verkeersvraag kijkt.

Het nieuwe model vergt meer calculatietijd maar produceert ook preciezere capaciteitsinschattingen. Significante veranderingen zijn te vinden voor het macroscopisch fundamentele diagram en voor filelengtes. De overeenkomst van filelengte met de filelengte van het microscopische model VISSIM is nu veel groter en heeft zowel het nieuwe link en knoopmodel nodig.

Het nieuwe link- en knoopmodel zijn onderdeel van het DNL (dynamische netwerkbelading) model van EVAQ. De meeste veranderingen hebben plaatsgevonden in dit model welke gebruikt kan worden in elk ander DTA (dynamische verkeerstoedeling) model. Daarnaast is het model gebaseerd op theorie en kan het zodoende gebruikt worden voor 'reversed engineering' en het extensiever analyseren van flessenhalzen, ook voor evacuatieschema's.

# Preface

This master thesis marks the end of my study at the Department of Transport & Planning of the Delft University of Technology. The research was performed at the ITS Edulab which is a cooperation between the Department of Transport & Planning and Rijkswaterstaat - Centre for Transport and Navigation. It offers great opportunities for students to work on real-life problems during their master thesis. I am very grateful for the opportunity to perform my master thesis at the ITS Edulab.

My gratitude goes to the committee members from Rijkswaterstaat, Marco Schreuder and Ydo de Vries, for their time and effort to read my report and give constructive criticism. Additionally I would like to thank Henk Taale, as he has also been involved in the project. I would also like to thank the committee members from the Department of Transport & Planning, Olga Huibregtse and Serge Hoogendoorn. They have shown great enthusiasm for the technical details of the models, which has certainly increased my own enthusiasm. Also my gratitude towards John Stoop from the Faculty of Aerospace Engineering at the Delft University of Technology for the focus on safety related issues.

I would also like to thank my fellow students and colleagues at Rijkswaterstaat for providing distraction with table soccer. Special thanks go to Xiaoyu Qian for sharing ideas about MFD's.

During my study I have had much support from both my parents, Piet and Beppie Schakel, and my fiancée Priscilla den Dekker. I thank you greatly for the emotional and financial support.

Wouter Schakel
Delft
August 28th, 2009

## List of common symbols

General

| | |
|---|---|
| $\overline{Q}$ | cumulative quantity (Q is a dummy) |
| $t$ | time step number |
| $\Delta t$ | time step size |
| $\tau$ | number of time steps related to a travelling distance |

Link model

| | |
|---|---|
| $C$ | link capacity |
| $X$ | number of vehicles |
| $L$ | link (part) length |
| $U$ | link inflow |
| $U^{\max}$ | maximum link inflow |
| $Q$ | queue inflow |
| $V$ | link outflow |
| $V^{pot}$ | potential link outflow |
| $\vartheta^{\max}$ | maximum (free flow) speed |
| $^{f}$ | related to free flow part |
| $^{q}$ | related to queue part |
| $_{a}$ | related to link *a* |

Cell Based Queuing

| | |
|---|---|
| $V(g)$ | link outflow related to cell *g* |
| $K(g)$ | density inside cell *g* |
| $W(g)$ | speed inside cell *g* |
| $L(g)$ | length of cell *g* |
| $T(g)$ | travel time through cell *g* |
| $S(g)$ | storage capacity of cell *g* |
| $V'^{pot}$ | link outflow respecting queued cell states |
| $T^{q}$ | queue travel time |
| $X'^{q}$ | queued vehicles that can reach the link head within $\Delta t$ |
| $X'^{f}$ | free flow and queued vehicles that can reach the link head within $\Delta t$ |

# List of common abbreviations

| | |
|---|---|
| CBQ | Cell Based Queuing |
| DNL | Dynamic Network Loading |
| DTA | Dynamic Traffic Assignment |
| EVAQ | Evacuation of Vehicles using Assignment with Queuing |
| LCM | Lane Choice Model |
| MFD | Macroscopic Fundamental Diagram |
| NPD | Network Performance Degeneration |

# Table of contents

# 1.Introduction

## 1.1 DTA, evacuation modelling and network performance

### 1.1.1. Dynamic Traffic Assignment

Dynamic Traffic Assignment (DTA) models are a class of models that can be used to model dynamic traffic processes. The term dynamic means that the state of model elements may change over time. For instance queues may change in length, route choice may change, pricing can be time dependant etc. DTA models use small time increments. For each time increment a new model state is calculated based on previous model state(s). Model components that are often found in traffic models that use DTA are demand modelling, route choice modelling and network loading. Demand modelling determines the number of people that wants to go from one place to another and which mode (car, bike, public transport, etc.) they will choose. Route choice modelling determines what route over the network people will take. Network loading is the simulation of traffic over the road network.

### 1.1.2. Evacuation modelling using EVAQ

A particular dynamic traffic process is the process of an evacuation. Both the network and the hazard are continuously changing. In the past few years a lot of research has been done on evacuations in the field of transport planning and traffic flow modelling. Recently a model by Pel, Bliemer and Hoogendoorn (2008) called EVAQ has been developed. EVAQ stands for Evacuation of Vehicles using Assignment with Queuing. The demand modelling determines when people will start their evacuation from their home. This results in a growing number of vehicles through time that wants to leave a certain area. Route choice modelling entails the choice of destination (save haven) and the route to take. Usually the destination choice is part of the demand modelling. For evacuations the destination is however not fixed, as the goal of the trip is not destination specific. The network loading simulates traffic on the network. Outcome is the network state (flows, speed, densities) through time. In EVAQ the DNL model from Bliemer (2007) is used. It avoids the use of link travel time functions (that result in the wrong location of queues) and explicitly model queuing and spillback. For evacuations it important that the location of vehicles is accurate as the hazard may strike certain areas at a different time than other areas. This has consequences for the number of casualties. EVAQ is used to evaluate the result of evacuation plans for which the number of casualties is very important. Another important criterion is evacuation time. For both criteria travel time plays an important role. Although EVAQ does not explicitly model travel time, it does model quantities that are related. Travel time can thus be an output of the model. An important modelled quantity is flow, which is the number of vehicles that can pass a certain location on the network within some time span.

#### 1.1.3. Network Performance Degeneration

By combining the flow of multiple locations of a network one can speak of network performance. There are multiple ways to combine flows such as weighted by link length, weighted by number of vehicles on a link and not weighted. In any case, network performance is a network wide quantity that explains the throughput of vehicles on the network. Because of interaction between vehicles on the network, network performance may be negatively influenced with an increasing number of vehicles on the network. This is called network performance degeneration (NPD). Processes that contribute to NPD have to do with interaction between vehicles. Many interactions exist such as blocking at intersections, lane changes, changes in speed etc. Chapter 3 will elaborate more on this.

## 1.2 Problem definition

In any model it is desirable to have a high level of accuracy. For evacuations this is difficult to achieve as traffic models cannot be calibrated and validated to actual data from evacuations as such events are (luckily) rare. It is therefore important to put much effort in accurately implementing existing knowledge of transport planning and traffic flow modelling. One important aspect of a road network is the possible degeneration of traffic performance under certain circumstances, as this may have large impacts on capacities and travel time. Most traffic models are concerned with travel time, money, emissions and safety. With evacuation modelling, human life is concerned as it is threatened by the hazard additional to traffic accidents. To preserve life, evacuations plans can be made that optimise evacuation time of given areas. Such evacuation plans depend largely on travel time and therewith on NPD. It has not been investigated whether EVAQ accurately models processes that contribute to NPD, leading to the problem definition of this research:

Problem definition

Evacuation plans are assessed largely by evacuation time for a given area. Evacuation time is highly dependent on network performance degeneration. It has not been investigated into what extend network performance degeneration is accurately modelled in EVAQ. This has to be investigated and if needed, EVAQ has to be changed and/or extended.

## 1.3 Research objective and questions

To improve the modelling of NPD the following research objective is formulated:

Research objective

To develop modelling solutions that correctly include processes that contribute to network performance degeneration in order to improve the accuracy of EVAQ and other DTA models.

Note that the research objective has a wider scope than the problem definition, as other DTA models are included. This is because improvements of EVAQ are focussed on the DNL module that can also be implemented into other models. Research questions are divided in two phases that are explained in section 1.4. The following questions will be answered in the two research phases to reach the research objective:

Phase one:
- What processes influence network performance degeneration?
- What processes are explicitly modelled in EVAQ?
- What processes are not explicitly modelled, but are an effective part of EVAQ?
- What processes need to be included in order to achieve better accuracy?

Phase two:
- What solutions can be created to include additional processes?
- What assumptions need to be made for these solutions?
- Are these assumptions more realistic considering network performance degeneration than the assumptions they avoid?
- Are the processes indeed significant for network performance degeneration?
- What are the consequences of the solutions on calculation time and memory use?

## 1.4 Research approach

The project will contain two phases in which the first phase is an investigation into EVAQ and NPD. The second phase is the generation and evaluation of solutions.

### 1.4.1. Phase one

EVAQ will first be investigated in order to be able to expand and change the model. Next, it is investigated what NPD is and what the causes are. EVAQ will be assessed and model runs will be performed with a Macroscopic Fundamental Diagram (MFD) as output. An MFD is a relatively new ‘tool’ in transport modelling. The tool is still under heavy research and cannot be used as a direct calibration tool [Daganzo & Geroliminis (2008)]. In other words, trying to fit the MFD of a model to the MFD of measurements will not guarantee accuracy. It however can be used to analyse what happens on a network scale. Model runs will only be performed for voluntary evacuations to filter out effects of evacuation plans. By including a network loading map, modelled processes can be identified. From phase one, shortcomings of EVAQ can be exposed.

### 1.4.2. Phase two

The second phase will be the creation, validation, implementation and evaluation of several solutions. Solutions are synthesized and a few will be selected based on realism. Theory-based solutions are chosen as these minimise the need of evacuation data. The selected solutions will be fully developed in order to implement them in EVAQ. The new model will be compared with VISSIM for voluntary evacuations in order to assess the relevance of the newly implemented solutions. Performance of the new model is assessed for all evacuation types.

## 1.5 Research scope

As many factors play an indirect or direct role in network performance, the scope will be limited to what directly influences it, being the infrastructure and the people, see Figure 1.1. Loosely translated into the model world this is equal to the network and the vehicles. People within the vehicles make many choices. Here, departure time choice, vehicle choice, destination choice and route choice are not considered, as these are an indirect influence. Direct influences are deceleration, acceleration, lane changes, etc. These influences are a narrow definition of 'driving' and are very common in microscopic models. EVAQ is however a macroscopic model. Therefore the behaviour is dealt with in terms of averages. One can for instance think of fundamental diagrams, which are the result of microscopic behaviour.

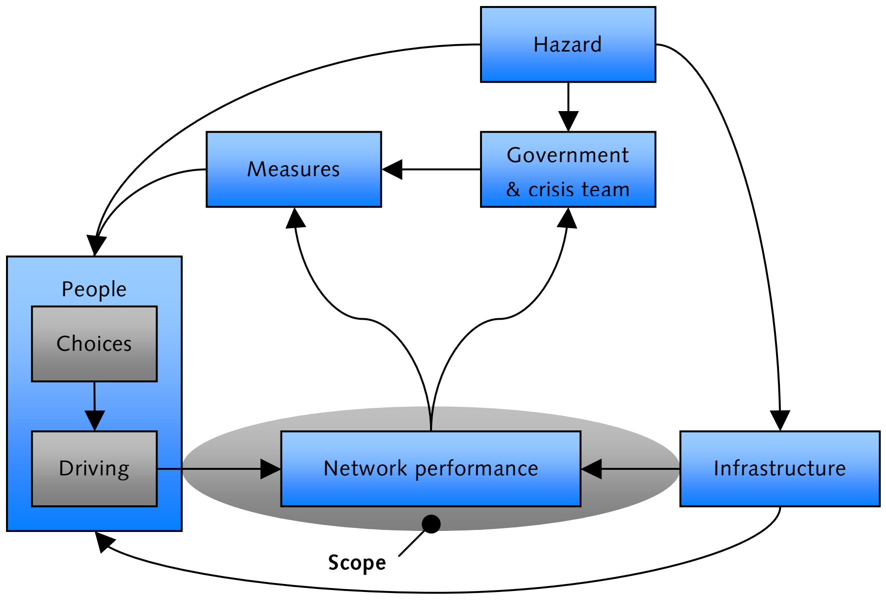


Figure 1.1: Scope of research
Only direct influences are considered

Also important to mention is that focus is on the development of modelling mechanisms rather than calibrating and validating model parameters. The reason for this is twofold. First of all calibration data is difficult to obtain, as evacuations are rare. Also, the time scope of this research is too short to be able to include a descent calibration.

## 1.6 Research relevance

The relevance of this research is subdivided into scientific and practical relevance as listed below.

Scientific relevance:

- The DNL model by Bliemer (2007) is changed and extended such that:
    - Queuing in the link model is dynamically and efficiently modelled with more precision.
    - The node model assesses the capacity of the node itself besides the still valid exit link capacities.
    - The following assumptions are circumvented:
        - Link inflow and link outflow have an equal capacity limit.
        - Every queue is at jam (practically full stop) state.
        - Conflicts at the nodes are insignificant for capacity. Maximum link inflow alone is representative.
- The model has become theory-based. The macroscopic network loading is coupled with microscopic phenomena. Theory about microscopic behaviour is translated to an aggregate level.
- EVAQ can be further developed where model runs that are part of an iterative development cycle can rely on more realistic DNL. Results will be less obscured by errors from the network loading giving a more clear view on the workings of other model components.
- Insight in intersection mechanisms and capacity constraints is gained and made explicit.
- The new weaving model introduces a new weaving theory in which lane choice is an important factor.

Practical relevance:

- The gain in precision enables a better assessment of evacuation plans. Such plans are important for the governmental organisation Rijkswaterstaat.
- Other DTA models can benefit from the new DNL model.
- Using detailed model results enables a process of reversed engineering in which for instance certain movements at intersections may be prohibited during an evacuation in order to minimise delays by flow interaction.
- Similarly for regular circumstances, black-spot analyses can be performed into the constraining elements (such as turn lanes) at intersections. Such elements may receive higher priority or higher capacity.

## 1.7 Reading guide

The structure of this report is displayed in Figure 1.2. The chapters are depicted into three general development phases: analysis, synthesis and evaluation. Note that the usual development phase of calibration is excluded, as explained in section 1.5 about the research scope.

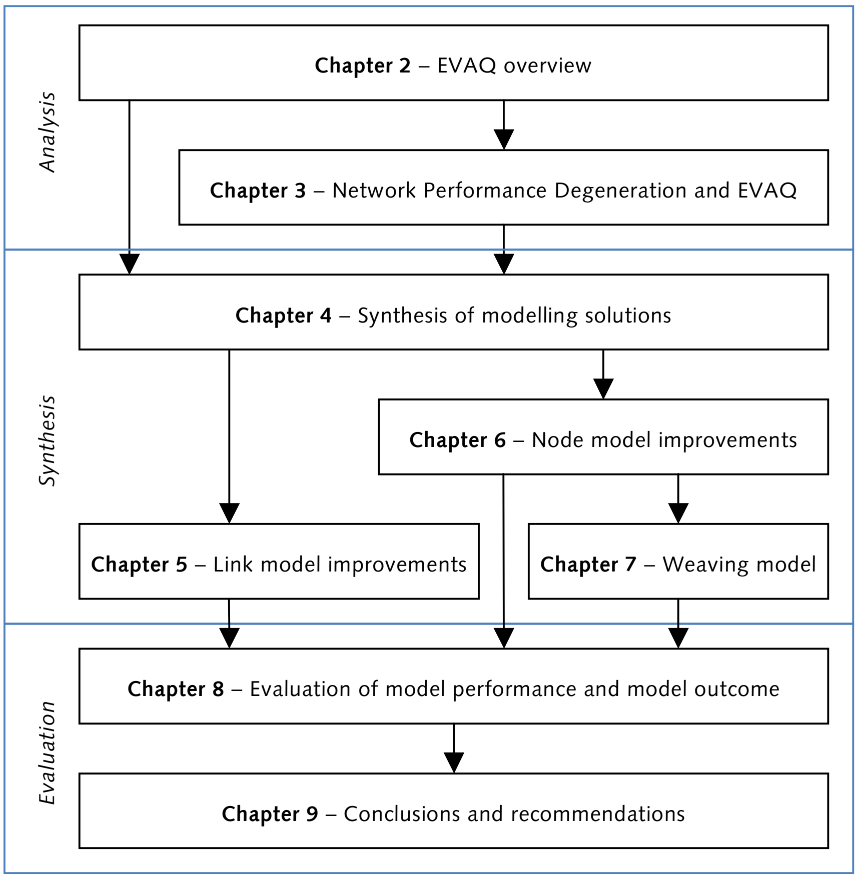

Analysis
Synthesis
Evaluation
Chapter 2 – EVAQ overview
Chapter 3 – Network Performance Degeneration and EVAQ
Chapter 4 – Synthesis of modelling solutions
Chapter 6 – Node model improvements
Chapter 5 – Link model improvements
Chapter 7 – Weaving model
Chapter 8 – Evaluation of model performance and model outcome
Chapter 9 – Conclusions and recommendations

Figure 1.2: Report structure

Chapter 2 – This chapter gives an overview of EVAQ. The model components and model loop are explained. The DNL model, important in this research, is briefly discussed as having two sub models being the link and the node model.

Chapter 3 – Section 3.1 is an investigation to what network performance and the degeneration thereof is. Causes are identified. Section 3.2 describes test runs performed with EVAQ to investigate how these causes are modelled. Important output of these test runs is the Macroscopic Fundamental Diagram (MFD). Following, section 3.3 lists observations from the test runs. Next, observations are made from the model assumptions and mechanisms. Finally section 3.5 answers the research questions of phase one. It is found that the link model can be extended with more detailed queuing dynamics and that the node model has no capacity constraints of the node itself.

Chapter 4 – In this chapter a series of possible solutions is presented. Section 4.1 till 4.5 briefly explain ideas to improve the model. In section 4.6 two solutions are selected on the basis of realism and a new modelling framework is presented in section 4.7. Cell Based Queuing, where each cell explains a part of the queue, is selected for the link model. For the node model it is recognized that capacity constraints of the node itself will be included.

Chapter 5 – This chapter further explains how Cell Based Queuing works. Section 5.1 first explains how the queue is represented. In sections 5.1.1 and 5.1.2 it is explained how the maximum link inflow and the potential link outflow are derived from the new queue representation. Section 5.1.3 elaborates on short links within the new representation, as the model needs to be able to deal with short links. This follows from the fact that nodes will need to resemble actual intersections rather than aggregated intersections. A numerical example is given in section 5.1.4. Section 5.2 will give some conclusions.

Chapter 6 – In this chapter the various node type specific sub models of the new node model are explained. In section 6.1 a few general changes to the node model framework are discussed after which section 6.2 presents the Lane Choice Model (LCM) that is the basis of the node type specific sub models discussed in this chapter. Sections 6.3 till 6.5 are about sub models for controlled intersections, uncontrolled & priority intersections and roundabouts. All these models are based on existing models and formulas.

Chapter 7 – This chapter describes the sub node model for weaving, merging and diverging sections at highways. It is treated separately as it is a new model, does not use the LCM and needs to be calibrated, as the model parameters are unfamiliar to traffic engineers. First, section 7.1 discusses existing weaving models and theory. Section 7.2 explains the new model that is based on lane choice of the road users. A numerical example is presented in section 7.2.4. The calibration, based on FOSIM data, is given in section 7.3.

Chapter 8 – Various aspects of the new model will be evaluated in this chapter. Section 8.1 is a qualitative assessment of the new link model (CBQ). It is shown that both free flow and congested traffic show the correct kinematic waves over the link. A sensitivity analysis is performed that shows sensitivity to the capacity and saturation flow parameters. Section 8.2 lists many qualitative observations of the various sub node models that are based on movies that display the free flow and congested traffic states of the links. Behaviour is as expected. Next a quantitative comparison is shown with VISSIM. Generally the intersection capacities are good but some link specific capacities show rather large errors, especially for the controlled intersection and the weaving section models. Section 8.3 shows the model significance of both the new link and node model with respect to the old model and the VISSIM results. Section 8.4 discusses the performance of the new model. Calculation time is largely increased for scenarios that are quickly calculated but only slightly increased for scenarios that are not quickly calculated. Finally section 8.5 discusses the applicability of the new model.

Chapter 9 – Conclusions and recommendations for both modelling and implementation are given in this chapter.

# 2.EVAQ

This chapter will elaborate on EVAQ and its components. A general overview will be given and the role of separate components will be explained. This chapter is not a detailed coverage of EVAQ but enables one to understand the general mechanism of EVAQ. Appendix A: EVAQ Algorithm Overview is a full technical explanation of EVAQ. The appendix can be used as a reference.

## 2.1 An overview of EVAQ

EVAQ is a traffic model aimed at evaluating plans for an evacuation. Various hazards can be investigated, as long as a time-spatial pattern can be described. The hazard not only creates casualties, it also influences the network by changing link parameters such as maximum speed and capacity. The model is dynamic in the sense that both traffic flows and the network change over time. The dynamics are described by three model components: demand modelling, route choice modelling and network loading. A route generation method from Bliemer & Taale (2006) and travel time estimation are used for the route choice modelling.

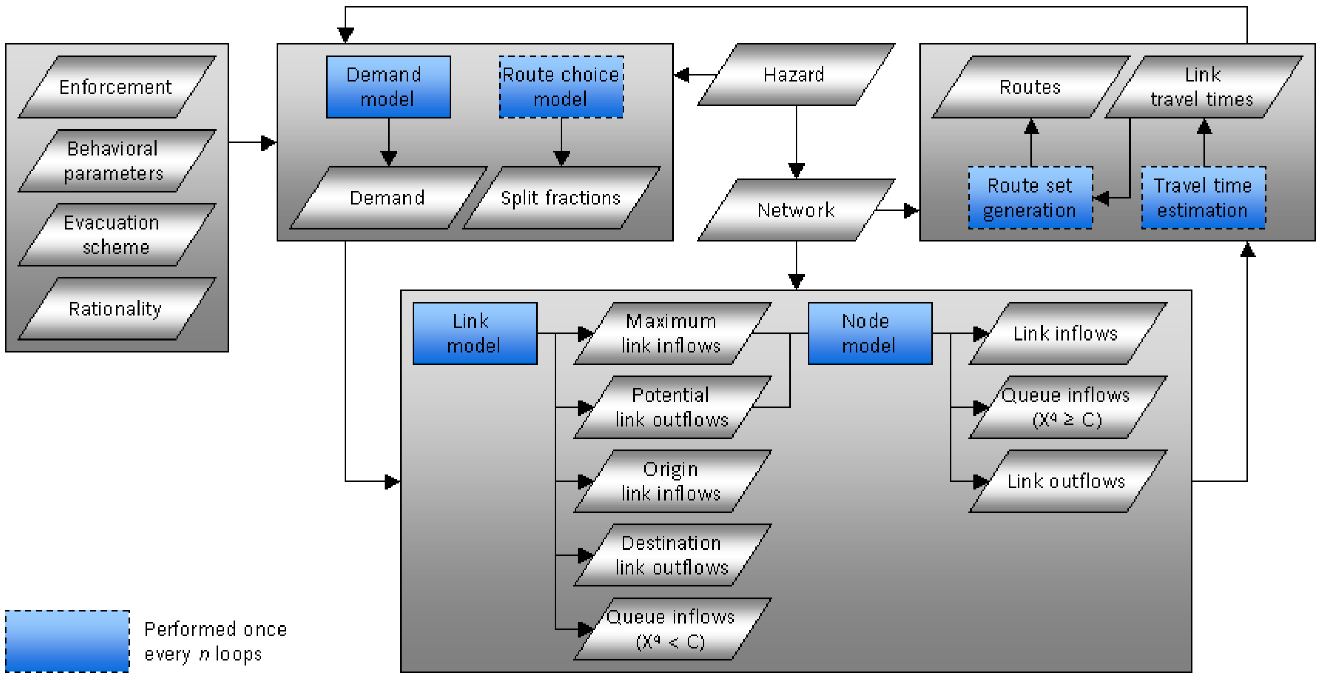


Figure 2.1: EVAQ Framework

The scheme in Figure 2.1 displays a typical loop of the EVAQ model for a single time step. The travel time estimation, route set generation and route choice model are not performed every loop as this significantly reduces calculation time with minimum loss of accuracy. Instead, split fractions of earlier time steps are used.

## 2.2 EVAQ modules explained

### 2.2.1. Demand model

The loop starts at the demand model where the decision to start evacuating is modelled. This decision depends on the time until the hazard strikes and possibly the given evacuation instruction. Instructions are included for mandatory or recommended evacuations. Voluntary evacuations have no evacuation instruction. For recommended evacuations a level of enforcement is included forcing people with different amounts to leave according to the evacuation instruction. The rationality of the road user is included with an aggregated parameter that adapts the utility to leave a given origin. It follows from the used logit model that this allows a distribution that can be anywhere between fully rational and fully irrational (indiscriminate). The demand model results in an increasing number of vehicles per origin that will enter the road network if possible.

### 2.2.2. Route choice model

The route choice model determines split fractions at each node. Split fractions depend on the available routes and the expected travel time of these routes. Also the possible route instruction and rationality are factors similar as in the demand model. It is assumed that people have knowledge about the location of queues. With this assumption the travel time estimation component can determine expected link travel time. Links damaged by the hazard get infinite travel time. Multiple routes are generated for each node with increasing stochastic variation for link travel time. This captures differences in human perception of expected travel time. Based on the deterministic travel time, the route choice model divides flows from each node over the connecting links based on the generated routes. The distribution results in split fractions. The route choice model, route set generation and the travel time estimation are not performed each loop. To reduce calculation time the same split fractions can be used for a few successive time steps without large consequences for accuracy.

### 2.2.3. Network loading model

The DNL model from Bliemer (2007) is used. Two sub models, the link and the node model, determine the exact way in which vehicles transverse the network.

*Link model*

The link model splits links into two parts, the free flow and the congested part, as in Figure 2.2. Vehicles transverse the free flow section at free flow speed. At the edges of the parts, cumulative flows are tracked. Cumulative link inflow, queue inflow and link outflow are the central quantities that the network loading is based on.

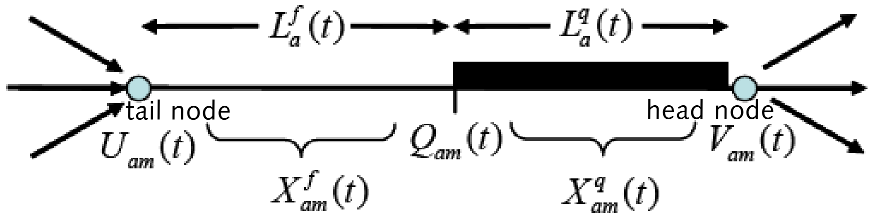


Figure 2.2: Link model [Bliemer (2007)]

The task of the link model is to determine what number of vehicles can enter and leave the link within a time step. Potential outflow is determined by vehicle supply (inflow patterns) limited by capacity. Maximum inflow is determined by the remaining link storage capacity limited by capacity. The number of vehicles in queue determines the queue length, assuming a single and fixed queue density. Based on the queue length, the queue inflow is determined from the inflow pattern and the current expected link travel time can be estimated for the route choice model.

*Node model*

The node model applies split fractions to the potential link outflow of the link model. If this results in a potential link inflow that exceeds maximum inflow, all flows over the node are reduced accordingly. This is how congestion is initiated either by the limit of capacity or remaining storage capacity. The latter may be limiting with long queues, creating a spillback mechanism.

## 2.3 Conclusions

EVAQ is a model that captures the traffic process of an evacuation. Many assumption are contained within the model about departure choice, route choice combined with destination choice and how vehicles transverse the network. Mainly the DNL model is of interest within the scope of this research. Central in the DNL model are the cumulative flows. The next chapter will investigate NPD using a macroscopic fundamental diagram, which is related to link flow. An assessment will be performed to see if the DNL model of EVAQ deals with the degeneration of network performance as expected.

# 3.Network Performance Degeneration

This chapter will start with an investigation into what network performance is and the degeneration thereof. Next, causes to this phenomenon are identified in order to evaluate if EVAQ is able to model the degeneration of network performance. In section 3.2 some test runs with EVAQ are described that have a macroscopic fundamental diagram (MFD) as output, from which observations are made in section 3.3. From chapter 2 it can be learned what EVAQ models and what assumptions are made. Based on this, section 3.4 observes shortcomings in EVAQ with respect to the causes of NPD. Finally some conclusions are drawn from the observations.

## 3.1 Network performance

Performance of a network is a rather general term that can be interpreted in several ways. The main indicator within this document is macroscopic flow. Macroscopic flow also has many definitions; here a non-weighted summation of flow over all nodes is used. This measure is used as links have multiple flows due to the link model. By the law of flow conservation, node flow is equally valid. Note that origin and destination nodes have no flow 'over' the node. Only en-route nodes are thus considered. Other definitions of macroscopic flow exist, such as weighted flow by length or pcu, but here the exact definition is not important. Of importance are the conclusions that can be drawn when looking at macroscopic flow. Such conclusions are easier to derive from non-weighted flow. Travel time may seem a more intuitive indicator, but it is less directly connected to the model. A problem is also that travel time cannot be determined within a single time step. Any vehicle arriving at its destination will have a travel time dependant on previous time steps, obscuring abrupt changes such as the start of congestion for a certain link. Interpreting how the model responds and works can be assessed more directly by using flow.

Macroscopic flow can be plotted against the number of vehicles (or pcu) that is on the network in total. This is called accumulation. By subtracting the cumulative outflow from the network from the cumulative inflow into the network, the net amount of vehicles can be determined for any time step. How exactly the relationship between the number of vehicles in the network and the macroscopic flow should look like, is subject of extensive research, see Qian (2009). However, some is known, the main thing being that from some point, network performance should degenerate [Daganzo & Geroliminis (2008)], see also Figure 3.1. The following sections summarize processes that contribute to this decrease in flow. Flow decrease with density increase (congested branch of fundamental link diagram), capacity drop and spillback/gridlock are mentioned.

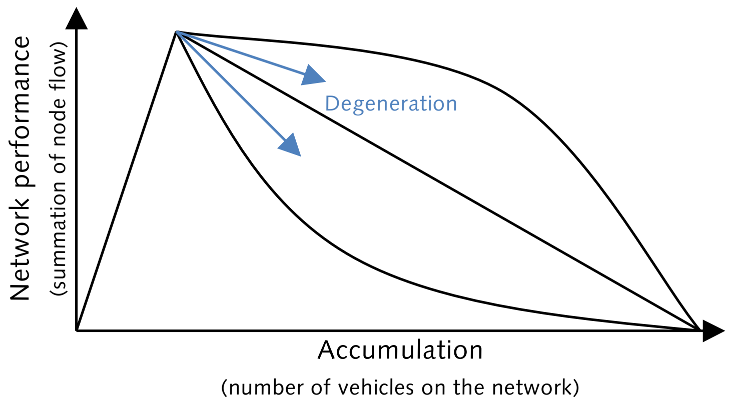


Figure 3.1: Example macroscopic fundamental diagram

### 3.1.1. Flow decrease with density increase

From fundamental link diagrams (density vs. flow) it is known that if density is higher than capacity density, there should be a decrease in flow as density increases [Syllabus CT4821]. This decrease is a highly stochastic process resulting in various definitions of fundamental diagrams. One thing however is omnipresent in all; average flow decreases gradually up to a point where flow becomes impossible, jam density. An example of a fundamental diagram (density vs. flow) is given in Figure 3.2.

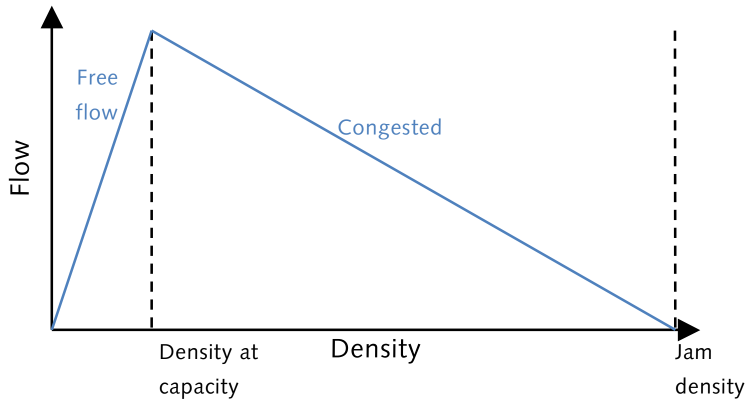


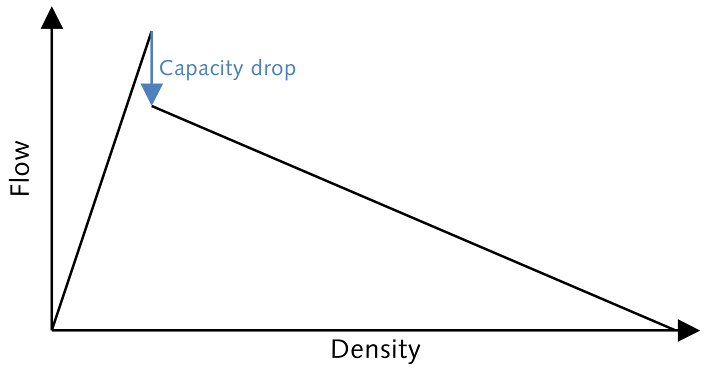
Figure 3.2: Example fundamental k-q link diagram

The decrease in flow is the result of less efficient driver behaviour. As density increases, the speed decreases more than by ratio. During evacuations it may be expected that drivers behave differently, resulting in a different shape of the fundamental link diagram. The reduction of flow will however remain, as this is inherent to driver behaviour. As the accumulation of a network increases, more and more links will become congested. Total flow will thus start to decrease at some level of accumulation.

### 3.1.2. Capacity drop

Many traffic flow researches assume the fundamental link diagram shows a capacity drop [Ning Wu (2001), H. M. Zhang (2000), Syllabus CT4821]. Different causes for this capacity drop can be assumed such as driver behaviour and maximum capacity downstream. In terms of driver behaviour, H. M. Zhang (2000) assumes different states where drivers can be in. Drivers are relaxed, anticipating or balanced. The result is several highway capacities. Figure 3.3 shows an example fundamental link diagram with capacity drop.

Figure 3.3: Example fundamental k-q link diagram with capacity drop



The effect of the capacity drop is a reduction of maximum flow at the head side of congestion. For example at traffic lights this value, the saturation flow, is about 1800 pcu/h. This is lower than regular capacity values. Also in other cases such as a bottleneck on a highway where vehicles are driving maybe at 50 km/h, it may be expected that drivers are unable to achieve maximum outflow equal to capacity. The effect of the capacity drop in fundamental link diagrams on MFD's is uncertain. Daganzo 2008 for instance does not show any kind of capacity drop in his proposed MFD.

### 3.1.3. Spillback and gridlock

Bottlenecks (of any form) put a limit on the maximum flow of upstream links. This includes intersections that are influenced by spillback from upstream links. Through the intersection, the queue effectively influences links further upstream. Note that the bottleneck itself has no direct influence in macroscopic flow reduction. It puts a limit on flow and any queue behind it suffers from further flow decrease only due to stochastic congestion processes. The resulting flow might even be lower than maximum flow in the bottleneck.

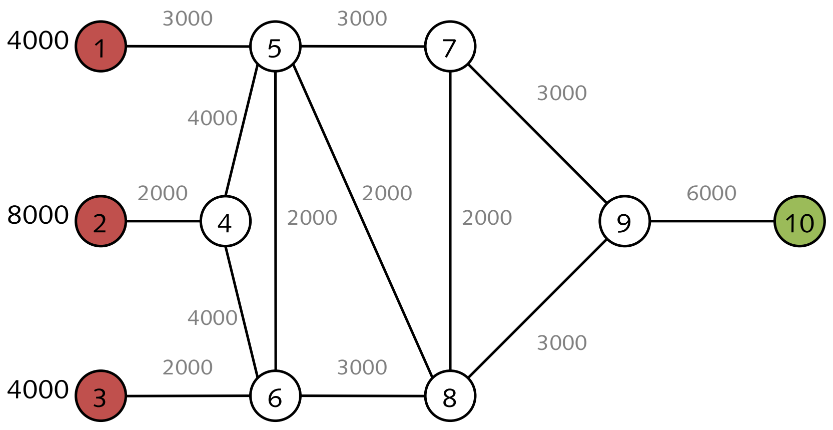
Links that are influenced by spillback will have reduced outflow, resulting in NPD. Vehicles that want to go to the link with spillback block vehicles that do not want to go to the link with spillback. All vehicles on a link are thus influenced and all flow is reduced. If vehicles enter the intersection while they cannot clear it, they block the entire intersection and all upstream links suffer from outflow reduction. If a set of links and nodes form blocking circles, gridlock has occurred. No vehicle can leave the intersection as no vehicle can enter the links. No vehicle can enter the link as the links are fully congested and no vehicle can leave the links. Gridlock is a very real phenomenon.

For network performance, spillback and gridlock can have severe effects. Especially if drivers have only few route options, the effect will be strong.

## 3.2 EVAQ test runs with an MFD as output

In order to find out how the chosen DNL method performs, a small test network was used, see Figure 3.4. There are three origins (nodes 1, 2 and 3) where a flood will strike. Population is listed at the origins. There is one save haven (node 10). Capacities are shown at the links in pcu/h. By adapting the capacity of the entrance links (1-5, 2-4 & 3-6), the effective demand on the network can be adjusted. The congestion on the entrance links themselves plays no role as only en-route node flow is considered. By adjusting the capacity of the exit link (9-10) the amount of congestion and spillback can be adjusted. In this way both dimensions of the MFD are covered.

Figure 3.4: Test network



Several runs were performed in which the following factors were used on the original capacity of the entrance links: 1/3, 2/3, 1, 4/3, 5/3 & 2. Note that vehicles do not leave spread evenly throughout an hour. Actually, full capacity of the entrance links in the first time steps is reached, even with a factor of two on capacity, since 98,2% of the inhabitants wants to leave immediately. This is because the hazard will strike in one hour, for which it is assumed (based on a logarithmic departure pattern) that this percentage of people has started their evacuation. The entrance capacities were used twice with different exit capacities of 2000 or 6000 pcu/h creating severe and minor spillback. Table 3.1 shows the parameters that were used in the runs. All runs are voluntary evacuations to filter out influences of an evacuation scheme.

Table 3.1: Test run capacities [pcu/h]

| | Run | Entrance capacities | | | Exit capacity |
|---|---|---|---|---|---|
| | | 1-5 | 2-4 | 3-6 | 9-10 |
| □ | 1 | 667 | 1333 | 667 | 6000 |
| ◊ | 2 | 1333 | 2667 | 1333 | |
| ○ | 3 | 2000 | 4000 | 2000 | |
| * | 4 | 2667 | 5333 | 2667 | |
| × | 5 | 3333 | 6667 | 3333 | |
| + | 6 | 4000 | 8000 | 4000 | |
| □ | 7 | 667 | 1333 | 667 | 2000 |
| ◊ | 8 | 1333 | 2667 | 1333 | |
| ○ | 9 | 2000 | 4000 | 2000 | |
| * | 10 | 2667 | 5333 | 2667 | |
| × | 11 | 3333 | 6667 | 3333 | |
| + | 12 | 4000 | 8000 | 4000 | |

Voluntary evacuations let drivers select their own destination and route. Usually, the closest destination and the shortest route are chosen. For gridlock to occur, there must be diverse route choice behaviour. For this reason, a separate test network was made as in Figure 3.5. The network represents an urban network with a ring road. More important however is that for this network, the route choice model was adapted. Split fractions are made equal for all links, creating diverse traffic. Off course this is not realistic route choice behaviour. The purpose of this run is however to find out if the DNL model will produce gridlock if circumstances are right.

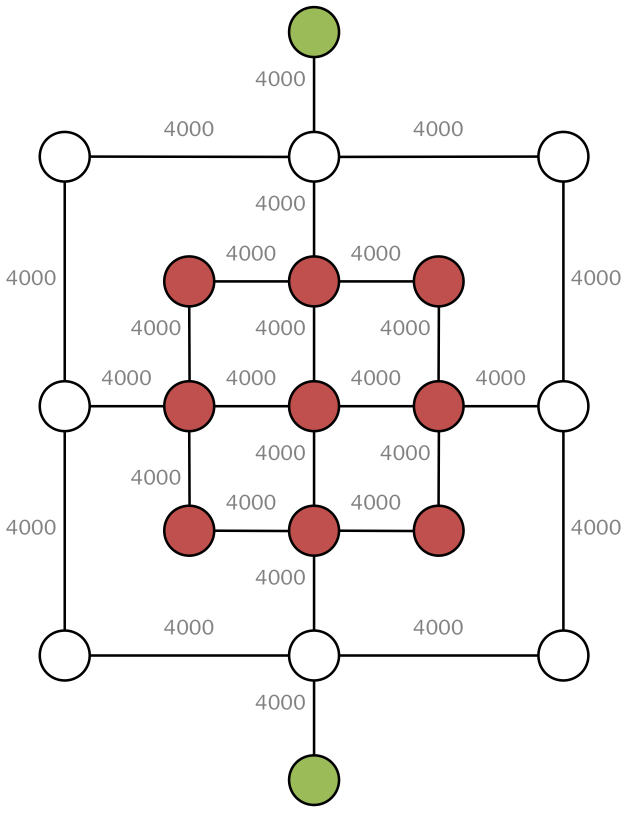


Figure 3.5: Test network for gridlock

## 3.3 Observations from the test runs

As described, MFD's will be used to find out what phenomena are present in EVAQ with respect to network performance. Each run produces a certain shape as can be seen in Figure 3.6 and Figure 3.7. The different runs are indicated with the symbol as in Table 3.1. Each dot represents a time step.

Some key observations can be made from the resulting plots:

1. There is a more or less linear free flow part.
2. There are several horizontal parts.
3. Jumps are visible in between the horizontal parts.
4. In case of a severe bottleneck, performance degeneration is visible.
5. The runs follow a clock-wise path more or less shaped as an italic '*p*'. This is similar as found by Qian (2009).
6. Additional to the steps, there are points in between with no apparent reason at first sight.

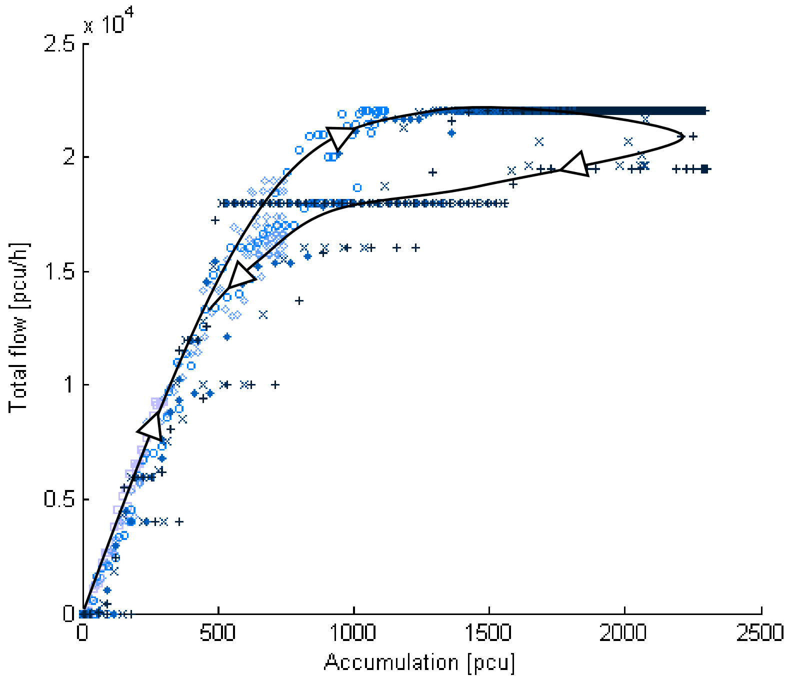


Figure 3.6: Macroscopic fundamental diagrams with $C_{9\text{-}10}$ = 6000

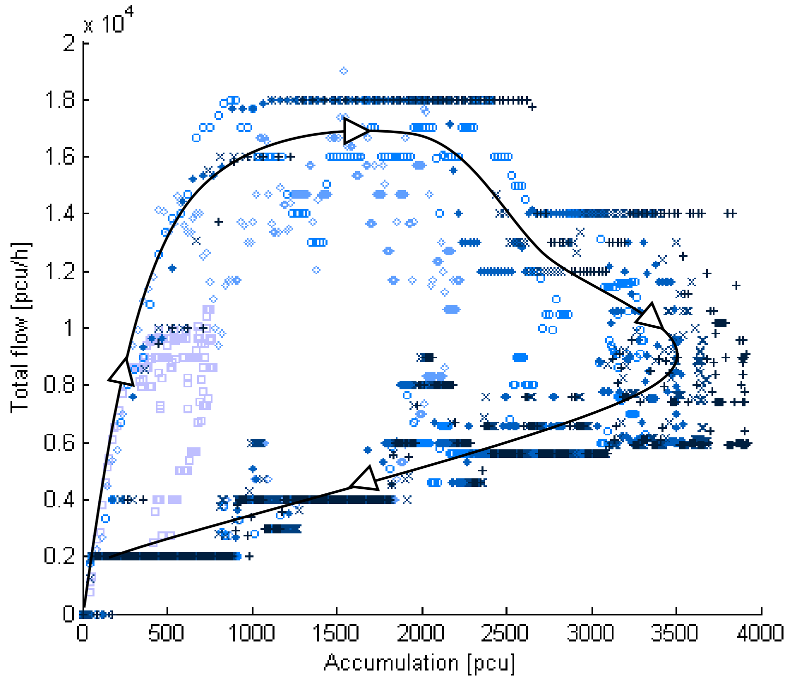
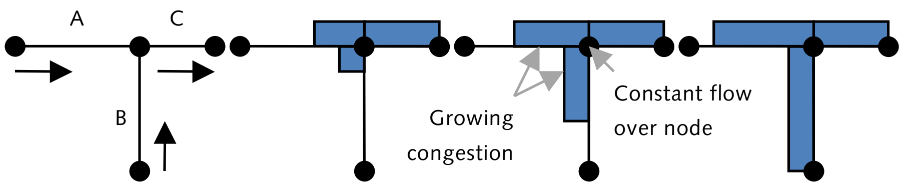


Figure 3.7: Macroscopic fundamental diagrams with $C_{9\text{-}10}$ = 2000

The linear free flow part is as expected. No NPD should occur for free flowing traffic, nor does it actually arise.

The horizontal parts are quite remarkable. The flow remains perfectly equal for a range up to 1000 till 2500 vehicles on the network. This is not at all comparable to a fundamental link diagram. The node model applies the same restrictions on link outflow during such a range, due to spillback. To explain, Figure 3.8 shows a small part of a network where links *A* and *B* merge into *C*. The second image shows that link *A* and *B* are affected by spillback from link *C*. Assuming that flow out of *C* remains constant, flow into *C* also remains constant. The third image shows that congestion on both links *A* and *B* grows. Flow over the merge node however remains constant. In other words, there are more and more vehicles on the network while flow remains constant. Only as soon as link *B* creates spillback will the macroscopic flow change, as in the fourth image.

Figure 3.8: Spillback



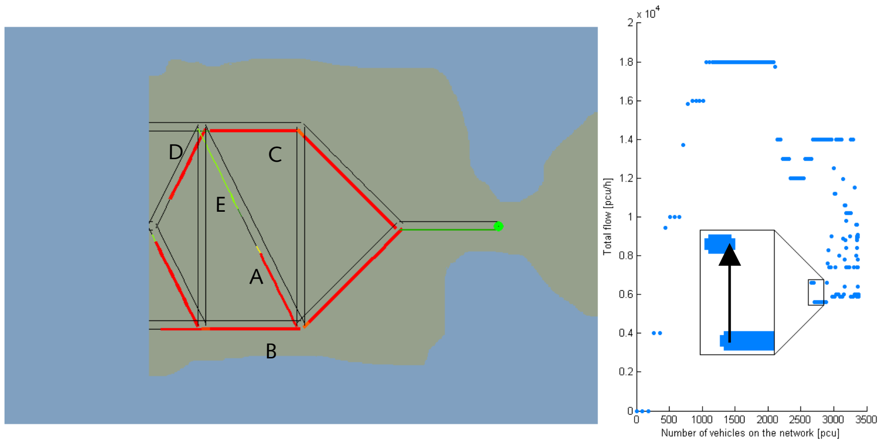
The steps between the horizontal parts can be explained in relation to the spillback. As soon as a link is spilling back, flows of upstream links will be reduced. The opposite happens if spillback disappears. Flows of upstream links will then increase if demand is still present.

A fifth observation that can be made from the results is a path that each run makes through the plots. It first rises up to a point where the network is filled enough for spillback to limit or reduce flow. Then there is a stepwise reduction of flow, where each step is a new link that is affected by spillback as explained earlier. After a while there is no traffic from the origins anymore, resulting in a stepwise reduction of flow. The steps now represent links that become *un*affected by spillback. Flow could go up, but actually goes down because there is no demand left. This is why spillback disappears in the first place. The path is strongly related to the demand pattern and the size of the network. What is important to note from this is the fact that if a congested network depletes, it will not follow the same path as when it got filled. This is different than generally assumed for fundamental diagrams of a link. Qian (2009) observes a similar cause for the shape of the MFD in the macroscopic DTA model MARPLE and the microscopic model VISSIM.

A last observation is chaotic (at first sight) points in between the steps. To analyse what this is, and also to verify the preceding statements, the macroscopic diagram of run 12 was plotted next to the loaded network into a movie. In this way the macroscopic diagram can be related directly with traffic phenomena. It appeared that short bursts of different route choice behaviour explained the chaotic points. The

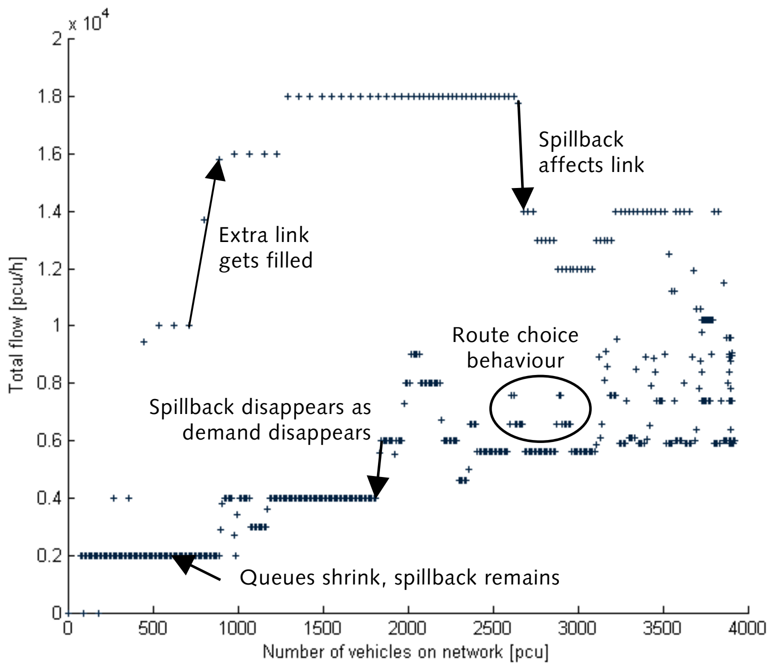
Wardrop principle [Wardrop (1952)] applies, making people temporarily use other links. Figure 3.9 shows this principle. Queues at *A* and *B* share a downstream link. The queue at *A* will therefore have lower outflow than the queue at *C* as both downstream links have equal capacity and thus flow governed by the network exit link. Vehicles from the queue at *D* will therefore mostly choose the route via *C*. At some point however, the queue at *A* is shorter than the queue at *C* and a few vehicles from *D* will change their route via *E*. Quickly the queue at *A* becomes longer than the queue at *C* and *C* is again the better alternative. Flow *E* increases macroscopic flow for a small time span, resulting in the gain as seen in the close-up of the plot (note that the path goes from right to left).

Figure 3.9: Route choice close-up



Similarly to the route choice phenomena, all previous statements were also visually verified. Figure 3.10 shows all observed phenomena.

Figure 3.10: Observed phenomena



### 3.3.1. Gridlock

The gridlock test run did indeed result in gridlock as Figure 3.11 shows. The nine central origins are all blocked. Vehicles on the nodes are unable to clear the nodes as each node has a fully congested link connected to it. EVAQ applies a single reduction factor on an intersection thus all flows are blocked. The congested links have no outflow, as the nodes allow no flow. There are in fact 2 blocking cycles and one link ending into a cycle. In the MFD we can observe that although there are about 8600 vehicles, there is no flow whatsoever.

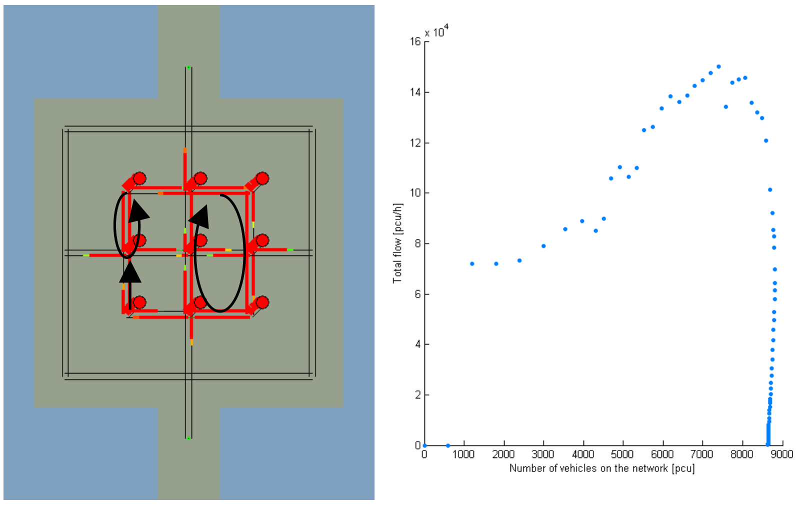


Figure 3.11: Gridlock test run

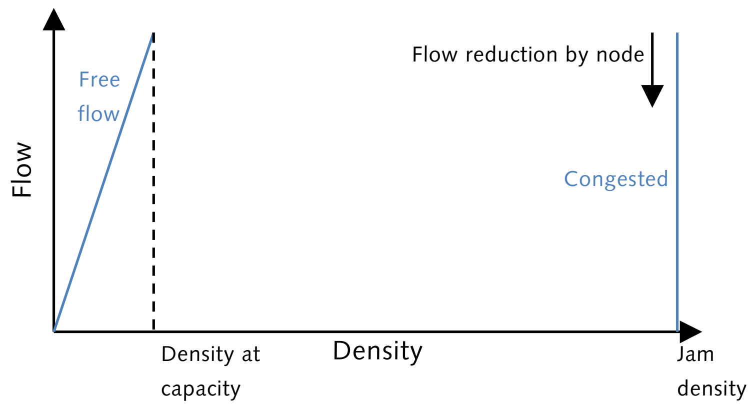
## 3.4 Observations from the dynamic network loading

The previous section has listed observations made from the MFD. By looking at the DNL model additional observations can be made. This section will explain the assumptions about the queued traffic state of the link model and the capacity constraints of the node model.

### 3.4.1. Queued traffic state

In EVAQ, some assumptions are made for the queued traffic state. The most prominent assumptions are a maximum outflow equal to capacity and a fixed density. These assumptions effectively assume the fundamental diagram as shown in Figure 3.12. The free flow branch is normal for a fundamental link diagram and needs no further explanation. The congested branch is far from normal, see also Figure 3.2. The cause of the vertical branch is the assumption that the queue is at a fixed density, being jam density. One could argue that there is a fundamental difference between the traffic state that an individual vehicle can be in and the average traffic state that the link can be in. Note however that fundamental diagrams are derived for cross sections and only apply to the entire link as the link is thought to be homogeneous. In other words, the fundamental link diagram relates to any cross section of the link, but is not about an aggregation of link length. Within the modelling framework, cross sections will mostly produce data points on the fundamental diagram of Figure 3.12. Only if both free flow and congested traffic arise at a cross section during the aggregation period will there be points in between.

Figure 3.12: Fundamental diagram according to EVAQ

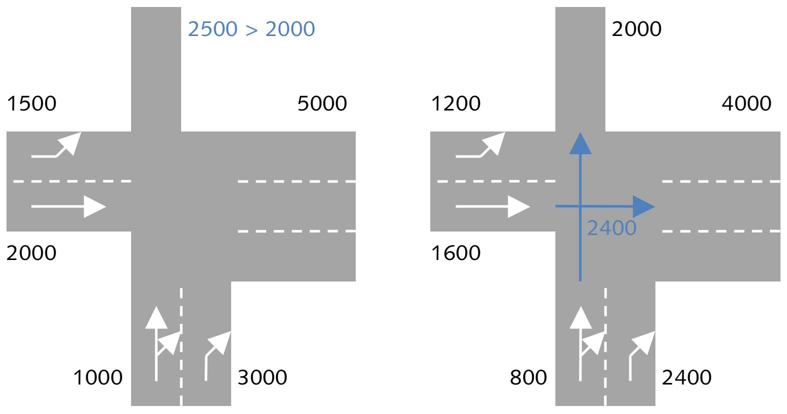


The fundamental diagram in Figure 3.12 explains to a large extend what happens in the test runs. The horizontal parts are explained by the fact that as long as queues are present and node constraints (spillback) remain equal, flow will not change. Clearly these assumptions are ignorant to the gradual relation between congested flow and density present in fundamental link diagrams, see also 3.1.1. Such a fundamental diagram might result in flow even lower than the limit by spillback. Qian (2009) found similar fundamental link diagrams in the MARPLE DNL model that uses a similar horizontal queuing framework. Densities between density at capacity and jam density are also found, as the *average* link density is related to link outflow. Qian identifies that the spatial separation of density and flow made the model unsuitable for the generation of MFD's.

### 3.4.2. Constraints in the node model

From the test runs it follows that the node model propagates congestion over the nodes and spillback is correctly modelled. It can be questioned if the amount of spillback is correct, as only link capacity is included as a constraint on flow. Intersections themselves pose capacity constraints on flows and can possibly lead to smaller flows and larger queues. As an example of the current node model an intersection is given with 2 incoming links, 2 outgoing links and 4 flows, see Figure 3.13 (left). The link to the north is a limiting constraint. It is shown that capacity of 2000 is exceeded. By reducing all flows by a factor of 0.8, the flow is reduced down to capacity (see the right image).

Figure 3.13: Current node model



So far there seems to be no problem. Note however that there is a conflict point between the two through movements that has a demand of 2400 pcu/h. According to the Syllabus CT4822 the following capacities apply to a conflict:

- *Uncontrolled intersection*
  Depending on the flow ratio, the capacity of two crossing flows is anywhere between 1600 and 2000 pcu/h.
- *Controlled intersection*
  A base saturation flow of 1800 pcu/h is utilized during the share of green time. Many factors can by applicable to the saturation flow, many of which tend to lower the saturation flow.

Apparently, in this case the capacity constraints of the intersection itself are more limiting than the link capacity. Constraints of the intersection should thus be included for a correct assessment of capacity.

## 3.5 Conclusions

Using the results presented in this chapter, the research questions of phase one can be answered.

*What processes influence network performance degeneration?*
These processes are discussed in section 3.1 and are flow decrease with density increase (fundamental link diagram), capacity drop for congested traffic, spillback and gridlock. All these processes originate from congestion. From section 3.4.2 it may also be concluded that capacity constraints of both the links and the nodes can trigger congestion.

*What processes are explicitly modelled in EVAQ?*
From chapter 2 it follows that the node model has two causes for congestion that both relate to the maximum link inflow. Either the link capacity is exceeded or the link is fully congested. Congestion is thus explicitly triggered and spillback is explicitly modelled.

*What processes are not explicitly modelled, but are an effective part of EVAQ?*
Related to spillback, gridlock is also an effective part of EVAQ. It has been shown in section 3.3 that the single reduction factor enables blocking cycles.

*What processes need to be included in order to achieve better accuracy?*
Remaining processes that are not covered by EVAQ are flow decrease with density increase, capacity drop and capacity constraints of intersections.

From the answer to the last question it may be concluded that EVAQ, and especially the DNL module, can be improved. A more detailed representation of congested traffic will be needed to vary the density of queues. A capacity drop should be imposed on congested traffic and the node model will have to be extended with capacity constraints of the node itself.

The next chapter will present several solution directions to implement the above changes. The three following chapters will work out selected solutions into more detail.

# 4. Solution directions

The previous chapter has shown that EVAQ omits certain processes contributing to NPD. This chapter will explore some general ideas to include these processes. Some of the solutions are then selected for further development. The next three chapters will elaborate on the selected solutions in more detail. The solutions discussed in chapters 5 & 6 for the link and the node model respectively are extensions of existing theories, models and formulas. Chapter 7 elaborates on the weaving model, which is part of the new node model. This model is entirely new. For the link model, the solutions will try to capture queuing into more detail. The node model solution will elaborate on flow interaction at the node, possibly triggering congestion. The current node model will not change as imposing flow reduction whenever vehicles cannot clear the node nicely covers spillback and gridlock. This holds for all node types as soon as congestion is significant.

Sections 4.1 till 4.3 will describe new ways to represent congestion. Solutions are; using an average congested state, directly implementing a fundamental diagram that relates outflow with the density of the entire queue and Cell Based Queuing (CBQ), which also relates outflow with densities in the queue but includes shockwave theory. Section 4.4 will elaborate on imposing a limit on outflow for congested traffic. Section 4.5 will generally explain how capacity constraints of nodes can be included. Next, a selection of the solutions will be made and a new modelling framework is presented. Finally, conclusions will be drawn.

## 4.1 Average congestion state

A simple change to the model, without introducing any additional computation time or complexity, would be to use different values for congested traffic. Instead of using a queue density equal to jam density ($k^{jam}$), and a maximum queue outflow equal to capacity ($C$), one could opt for more conservative and average values. One could for example represent congested flow by the median congestion values:

Equation 4.1

$$k^{queue} = k^{capacity} + \frac{k^{jam} - k^{capacity}}{2}$$

$$q^{queue} = \frac{C}{2}$$

$$u^{queue} = \frac{q^{capacity}}{k^{capacity}}$$

Note that the head node might limit the flow. This would also change the density and speed of the queue according to a fundamental link diagram. The current model does not deal with this. Values could be calibrated to represent congested traffic as close as possible. Data from evacuations is however hardly available. A new fundamental diagram arises from these assumptions as shown in Figure 4.1.

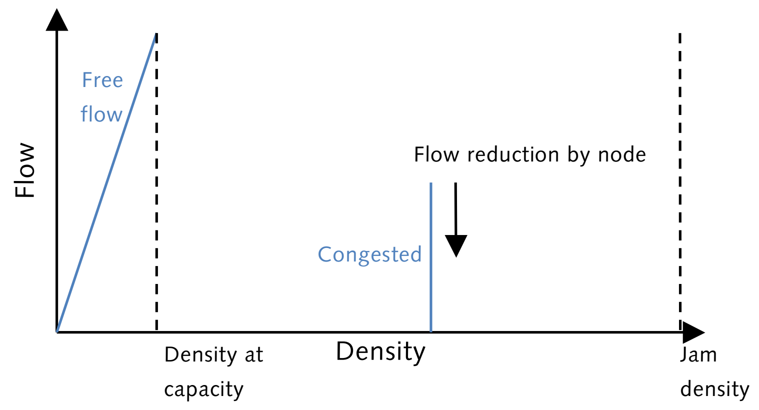
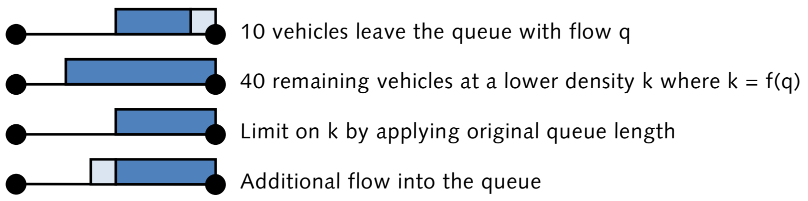


Figure 4.1: Fundamental link diagram with average congestion state

## 4.2 Direct implementation of a fundamental diagram

Another way to represent the queue is a new concept that uses a fundamental link diagram to derive a single queue state. For a given (out)flow, a fundamental diagram assumes a certain density and speed. The density could be applied to the entire queue, assuming instantaneous kinematics. This could potentially lead to vehicles effectively going back. In order to prevent this, a limit should be applied on the density. For example if 50 vehicles stand in a queue of 500m and 10 vehicles will leave within a time step, then the density of the next time step can never be lower than 40 vehicles over the same 500m. This limit keeps the last vehicle in queue at time *t* on the same spot for time *t+1*, see also Figure 4.2.



10 vehicles leave the queue with flow q
40 remaining vehicles at a lower density k where k = f(q)
Limit on k by applying original queue length
Additional flow into the queue

Figure 4.2: Direct implementation of fundamental link diagram

Density itself might limit outflow for the next time step. Applying this with the same *k-q* relationship makes it impossible for density to get below the defined *k-q* line while the limit of original queue length makes it possible for *k* to be larger than the defined *k-q* line (Figure 4.3, B). For a representative *k-q* noise (Figure 4.3, A) either the *k-q* relation needs to be defined lower than average (Figure 4.3, C) or different *k-q* relations should be defined for *k = f(q)* and *q = f(k)* (Figure 4.3 D).

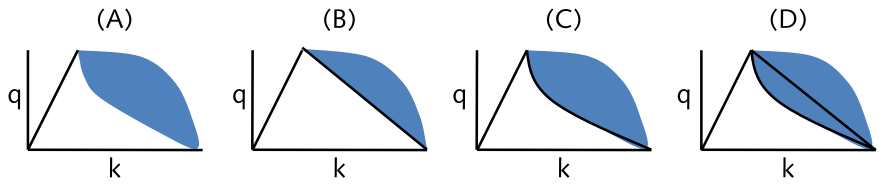


Figure 4.3: *k-q* relations with noise

## 4.3 Cell Based Queuing

In some DNL models [Daganzo (1993), Yperman, Logghe, Tampere & Immers (2005)], kinematic wave theory is applied on either cells or links. Different traffic states (flow, density, speed) are assumed to move along a link with a certain wave speed. Assuming a triangular fundamental diagram entails a single wave speed for free flow traffic and a single negative wave speed for congested traffic ($w$). Cumulative vehicle numbers now are related to cumulative vehicle numbers in the past. The time that is looked back is dependent on the link or cell length and the kinetic wave speed to be used. A problem with cell based models is that they are CPU demanding as dynamics are evaluated for each consecutive pair. A new modelling framework is thought up that prevents calculations per cell but rather uses all cells on a link at once. This framework is called Cell Based Queuing (CBQ).

To capture the theory of kinetic waves we can split the link into cells, but here only for the congested part. Cells will have a length that is equal to what a congested kinetic wave transverses in a time step. This is different from Daganzo (1993) where each cell has a length related to free flow speed. Each cell holds the average traffic state, initiated in a particular time step. The cell states are initiated by link outflow. As congested waves move upstream, so do the cell states. In this way the congested kinematic waves are represented in the queue.

*Link inflow*
In the current model, maximum inflow of a fully congested link equals the remaining storage, assuming jam density. Kinematic wave theory however demands that maximum inflow equals the outflow some time ago [Yperman, Logghe, Tampere & Immers (2005)]. The time that needs to be looked back equals $L_a/w$ or the time it takes for a congested kinematic wave to transverse the link. This phenomenon is represented by the cell states as the upstream cell actually holds this traffic state. The maximum inflow is still the remaining storage, but the storage is based on densities from the cell states.

*Queue inflow*
It makes no sense to apply a similar principle on queue inflow. Instead of limiting the queue inflow of a fixed point, all potential queue inflow should be accepted into the queue and the queue length should be adjusted accordingly, as is practice in the current model. Note however that the queue length is determined by the densities from the cell states.

*Link outflow*
Having the queue respecting the cell states also imposes speeds in the cells. Because of these speeds, there may be a limited number of vehicles that could actually reach the link end within a time step, as can be seen in Figure 4.4 where $T$ is the travel time towards the link head and $S$ is the cumulative number of vehicles. The link is split into cells as indicated by the dash-dot lines. For free flow traffic a limit on potential outflow is already achieved from link inflow following wave speeds

where the speed is simply the maximum speed resulting in $\tau$, which is the (non integer) number of periods relative to $t$ that vehicles now entering the queue, entered the link. Note that ‘entering the queue’ equals ‘leaving the link’ if no queue exists. A linear interpolation over link inflow is performed to determine the number of vehicles that can reach the end of the link within the time step.

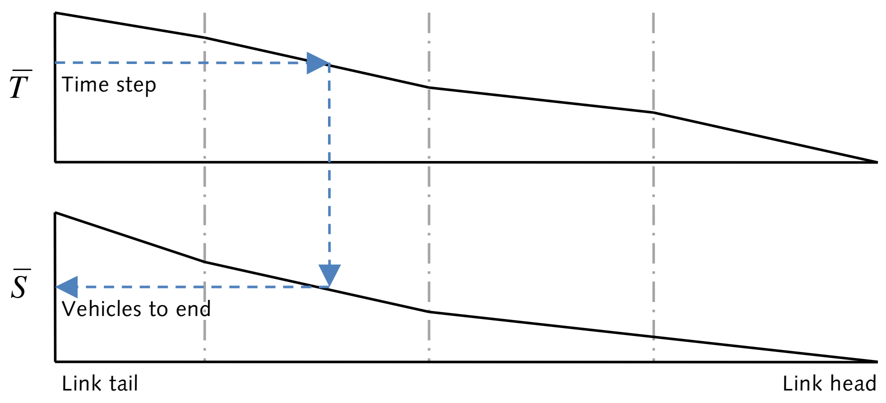


Figure 4.4: Queued vehicles that can reach the link end within a time step.

Different from existing models is the fact that the cell states are aggregated into cumulative quantities per link, providing an easy and fast way to derive momentary maximum link inflow and potential link outflow. Dynamics for each cell pair do not need to be assessed.

Besides the free flow and congested waves described, there are also standing waves. Lenz, Sollacher & Lang (2001) indicate that these waves are usually formed by local speed control. They are the result of stop and go waves that are flattened out. In order to include this phenomenon, local speed control would have to be modelled. This would be very complex, as cell length will probably not coincide with the gantry locations where standing waves are initiated. Lenz, Sollacher & Lang (2001) however also indicate that highway sections with a standing wave have (almost) constant flow both over the section and over time. Using a fixed fundamental diagram does not allow the differences in speed around a standing wave at an equal flow. In reality, standing waves contribute to the scatter often witnessed in fundamental diagrams. Using a representative fundamental diagram takes away the need to include standing waves.

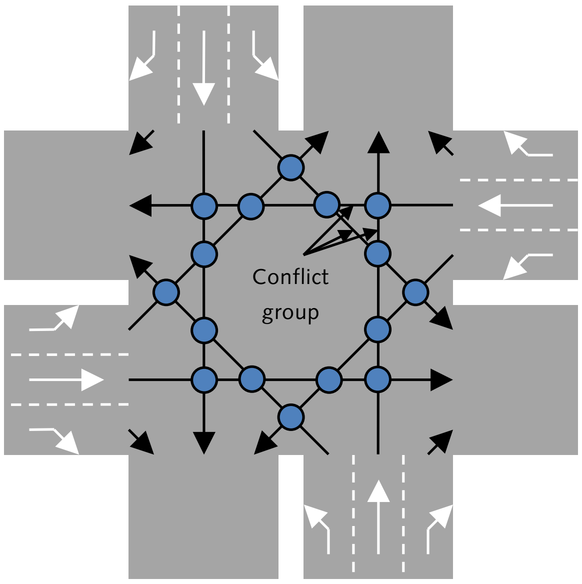
## 4.4 Congestion outflow limits

In EVAQ it is assumed that outflow on a congested link can be as high as the capacity of the link. Bliemer (2007) explains that capacity in this context should be defined differently from capacity as in a fundamental link diagram. However, in EVAQ all links have a single capacity determining both the maximum in- and outflow. In congested state, this seems an overestimation of capacity. Examples to contribute to this argument are the base saturation flow at traffic lights (1800 pcu/h/lane [Syllabus CT4822]) and the congestion discharge rate at highways [Ning Wu (2001)] which might be related to capacity drop [Syllabus CT4821]. Reducing queue outflow will degenerate network performance to a larger extend than using capacity flow. Congested outflow capacity might be dependent on the type of link as people behave differently on for example an urban road or a highway.

## 4.5 Constraints in the node model

The node model will need to be extended in order to include capacity constraints of the node itself. Extra constraints can help to limit the use of conflict points to the capacity of such a point. This capacity is dependent on the type of node. Also the way in which flows interact is different for different nodes. Similar to traffic light groups, there can be multiple flows crossing all other flows in the same group. Figure 4.5 shows an example of this. Constraints should thus be defined for groups, and not just for pairs. These constraints can potentially play a large role in NPD. Focusing on this, a method for evacuation routing is proposed by Cova & Johnson (2002) that minimizes the number of conflict points since most traffic delays actually occur at intersections [Southworth (1991)].

Figure 4.5: Conflict group



## 4.6 Selection of solutions

Five different implementations for proper NPD were discussed. Some of these will be worked out in the following chapters and eventually implemented into EVAQ. Four out of five solutions cover the link model, one covers the node model. The solution for the node model is however very general. What it comes down to is additional capacity constraints. For the link model a solution must be selected. Theory based solutions are preferred as these minimize the need for calibration data, which is difficult to obtain for evacuations. To recap, the link model solutions were:

1. Average congestion state
2. Direct implementation of a fundamental diagram
3. Cell Based Queuing
4. Congestion outflow limits

The first solution is attractive as it is simple. But the level of realism is low. Different queues may have very different average traffic states, depending largely on the available outflow capacity. This can potentially have large consequences for travel time and queue length and therewith the number of evacuees. The second solution may produce more realistic results, but this is very doubtful. Solution two assumes limited instantaneous kinetics, which is a far cry from reality. There simply is no theoretical background, not for the instantaneous kinematics nor for the limit on density by original queue length. This makes the solution unreliable.

The third solution has an inherent precision of queuing dynamics that corresponds with the time step of dynamic traffic models. At the same time it prevents cell-to-cell flow calculations. The solution covers both in- and outflow using kinematic wave theory. Due to the theoretical background and the full coverage of traffic dynamics that are currently not modelled in EVAQ, this solution is selected for further development.

The fourth solution is valid, but only covers a single aspect of queuing (the outflow limit). This solution is also selected for further development and will actually be a part of CBQ.

## 4.7 New dynamic network loading modelling framework

The new modelling framework of the DNL will remain similar as in Figure 2.1 and is given in Figure 4.6. The link model is still based on splitting the link into a free flow and a congested part. Potential outflow and maximum inflow are still determined from the state of the link but the congested traffic states are deduced by the intermediate step of CBQ. The node model checks capacity constraints, that now also include constraints of the node itself, and determines actual flows. Determination of the origin and destination flows has moved from the link model to the node model, as the node model determines the actual flows. The link model implicitly uses the outcome of the node model by calculating the number of vehicles on a link and in the queue. As CBQ is based on past outflows, there is now also an explicit feedback from the node model to the link model.

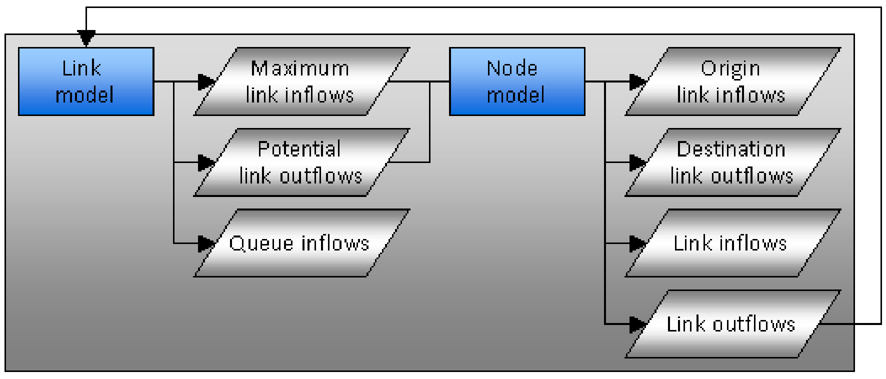


Figure 4.6: Dynamic Network Loading framework

The complexity of the framework has reduced, but the individual modules will increase in complexity. The link model defines the queue as multiple cell states based on past outflows instead of a single constant state. The node model will additionally check constraints on the node.

## 4.8 Conclusions

In this chapter several solutions have been presented that include processes that cause NPD into EVAQ. A selection has been made based on precision and a preference for theory based models. The link model will be expanded with CBQ in order to represent the queue with more realism. Both link inflow and outflow can be influenced by this representation. Additionally, a limit on congested outflow will be introduced within the CBQ framework. For the node model, node type specific sub models are needed that evaluate the capacity of conflict groups. The new DNL framework has also been presented in which there is an explicit feedback from the outflows to the link model as CBQ depends on the outflows.

Chapter 5 explains in detail how CBQ works. Chapter 6 will cover the node model and sub models for controlled intersections, uncontrolled & priority intersections and roundabouts. All these sub models use a Lane Choice Model that is also explained. Chapter 7 presents another sub model of the node model, the weaving model. It is dealt with separately as it does not depend on the Lane Choice Model and is not based on existing models or formulas and thus also needs a calibration.

# 5. Link model improvements

The selected solutions of chapter 4 will now be explained in detail. This chapter will elaborate on CBQ. Section 5.1 will explain changes for the link model. The link model will be adapted by implementing CBQ. The current model will stay in place, separating the free flow and queue part. Also the main quantities, link inflow, queue inflow and link outflow, remain. This framework is simple but functions well. The queue part is however represented by cells, which increases the complexity. An intermediate step is added that translates past outflows to traffic states in the cells. For the cell at the downstream side of the link, a limit on congested outflow will be put into place. This will be explained in section 5.1.2. As can be learned from chapter 6, the node model will require nodes that resemble actual intersections. A consequence of this is that the link model will need to have the ability to deal with short links. This is explained in section 5.1.3. A numerical example will be given to illustrate the new link model after which conclusions are presented.

## 5.1 Cell Based Queuing

CBQ is a new way to represent queuing dynamics in a discrete manner. The discretisation is based on the time step used in EVAQ and a fundamental k-q diagram. The fundamental diagram can be assumed to be triangular [Yperman, Logghe, Tampere, Immers (2005)], resulting in a constant congested wave speed. Such a congested wave consists of a traffic state moving upstream on a link as vehicles force each other to decelerate, or let each other accelerate. As this wave speed is constant, the distance covered within a time step is also constant. A link can be divided in multiple cells with equal length. Only the last cell can have a different length because of the link length not being an integer multiple of cell length. Note that the order of cells (first to last) is in the direction of the congested wave, which is opposite of the driving direction. The cells form an intermediate step to derive potential outflow and maximum inflow of the link. This is the main difference with existing cell based models where in- and outflow is determined for each cell. With CBQ, the states in the cells together determine the number of vehicles that can be stored on the link, and the number of vehicles that can reach the end of the link. The states can be derived from previous link outflows via the fundamental *k*-*q* diagram. In Figure 5.1 a link with four cells is displayed. The first cell has a state based on link outflow of the previous time step. The second cell represents a congested wave that has travelled two time steps and is thus related to link outflow of two time steps ago. For the other cells the same applies with an increasing number of time steps in the past.

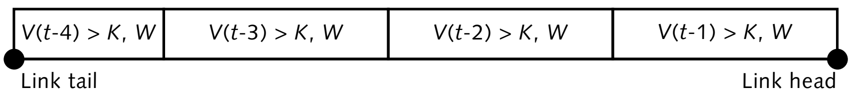


Figure 5.1: Cell states

Generally, the flows related to cells $g$ = 1...$G$ of link $a$ have a link outflow $V$ equal to the difference of cumulative link outflow of two time steps relative to the current time step $t$.

Equation 5.1

$$V_a(g) = \overline{V}_a(t-g) - \overline{V}_a(t-g-1)$$

Knowing the link outflow that governs each cell state, density $K$ can be derived by applying the fundamental k-q diagram.

Equation 5.2

$$K_a(g) = f(V_a(g))$$

With the equation of flow conservation, a speed $W$ can be determined.

Equation 5.3

$$W_a(g) = \frac{V_a(g)}{K_a(g)}$$

So far, each cell has a traffic state based on outflow in the past. These states move upstream by interaction between vehicles. This interaction is hardly present for vehicles at the first cell. In reality it is often observed that even from standing still, vehicles are able to achieve outflows near 1800 pcu/h. This flow is usually called saturation flow. Here saturation flow is viewed in wider terms, also including achievable outflow from speeds above zero. This flow is the basis of limiting congested outflow as described in section 4.4 and is denoted as $q_a^{sat}$. Here it is recognized that indeed such a limit is effective on the first cell. The first cell is always able to reach $q_a^{sat}$ if the node allows. When traffic starts to break down, outflow can be above $q_a^{sat}$. If this is true, vehicles are moving at such a high velocity that it is assumed that the flow related to the first cell can be equal to outflow instead of $q_a^{sat}$. Here it is proposed to use the maximum of both as governing flow for the speed (not density) of the first cell.

Equation 5.4

$$W_a(g=1) = \frac{\max\left(q_a^{sat},\ V_a(g=1)\right)}{K_a(g=1)}$$

Using saturation flow for the first cell also prevents a problem where no flow recovery is possible. If outflow is zero at any time step, the speed in the first cell will be zero. This makes it impossible for any vehicle to reach the link head, again resulting in no outflow.

Cell density and speed can be calculated to cell travel time *T* and cell storage *S* (number of vehicles that fits inside a cell at the cell density) using the cell length *L*.

Equation 5.5

$$T_a(g) = \frac{L_a(g)}{W_a(g)}$$
$$S_a(g) = K_a(g) \cdot L_a(g)$$

Cell travel time and storage are the crucial quantities to derive remaining link storage (maximum inflow) and the number of vehicles that can reach the link head (potential outflow). The first step is to calculate cumulative quantities.

Equation 5.6

$$\overline{L}_a(g) = \sum_{i=1}^{i=g} L_a(i)$$
$$\overline{T}_a(g) = \sum_{i=1}^{i=g} T_a(i)$$
$$\overline{S}_a(g) = \sum_{i=1}^{i=g} S_a(i)$$

These cumulative quantities might look like the graphs in Figure 5.2. Note that cumulative cell length is linear with link length. Also note that travel time and storage show similar patterns. This is because high densities form high travel times because of low speeds.

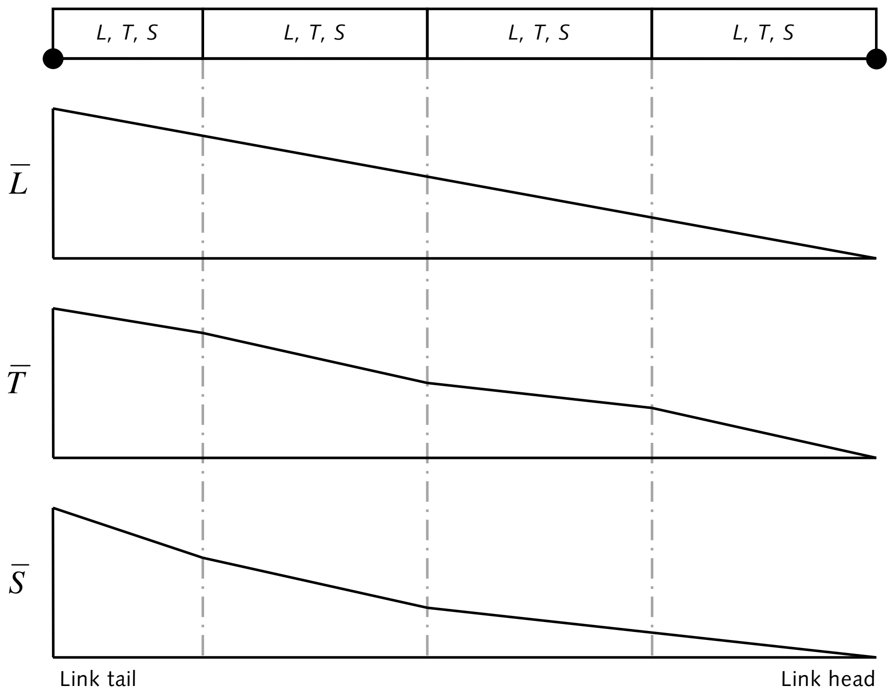


Figure 5.2: Cumulative quantities

Besides these cumulative quantities, the number of vehicles in the queue and on the link is also required.

Equation 5.7

$$X_a^q = \overline{Q}_a(t) - \overline{V}_a(t)$$
$$X_a = \overline{U}_a(t) - \overline{V}_a(t)$$

Based on the number of vehicles in queue and the cumulative cell storage, a queue length can be linearly interpolated.

Equation 5.8

$$L_a^q = \overline{L}_a\left(\overline{S}_a^{-1}\left(X_a^q\right)\right)$$

This interpolation is shown graphically in Figure 5.3.

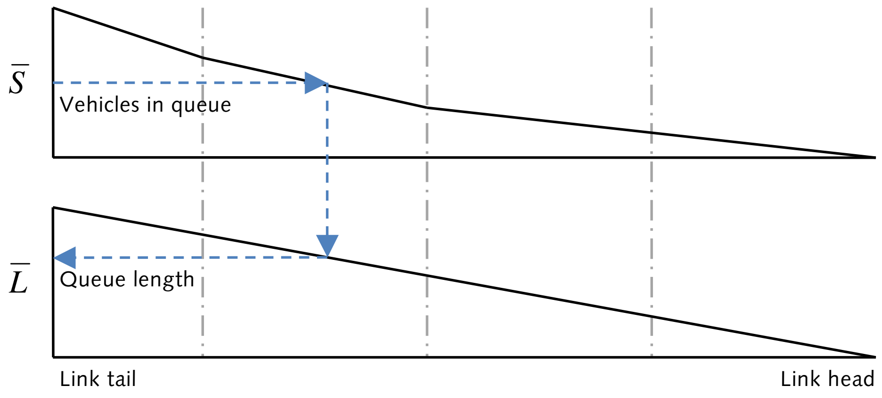


Figure 5.3: Linear interpolation from *S* to *L*

With the queue length, cumulative queue inflow until the next time step (*Q(t+1)*) can be calculated in the same way as in the original model using linear interpolation over link inflow, see Equation A.13. All information to calculate maximum inflow and potential outflow is now gathered. Queue inflow is known, completing the set of flows that make up the link model.

### 5.1.1. Maximum inflow

Maximum inflow has two limits being link capacity per time step and remaining storage. Remaining storage will only be limiting if the queue spans most of the link. This is equal to the old model but the queue densities and therewith the remaining storage will be different. Some models, such as the original DNL model by Bliemer (2007), assume that for fully congested links, maximum inflow is equal to current outflow. Such an approach is ignorant to queuing dynamics and also takes away the possibility for gridlock to occur. For these reasons, remaining storage is used as the theory of shockwaves is implied if the remaining storage is limiting. Remaining storage is defined as the total storage ($S_a(G)$) minus the total current number of vehicles on the link ($X_a$).

Equation 5.9

$$U_a^{\max} = \min\left(C_a \cdot \Delta t,\quad \overline{S}_a(G) - X_a\right)$$

### 5.1.2. Potential outflow

Potential outflow is seen as the number of vehicles that can reach the link end within a time step. If there is no queue ($X_a^q = 0$), potential outflow equals queue inflow as calculated earlier. Queue inflow is cumulative, so a difference between two time steps results in potential outflow. If there is a queue, potential outflow equals the number of vehicles that can reach the link end ($V'^{pot}_a$) obeying the queued cell states. This can be any number of congested vehicles and maybe also a few free flow vehicles if the queue is short.

Equation 5.10

$$V_a^{pot} = \begin{cases} \overline{Q}_a(t+1) - \overline{Q}_a(t), & X_a^q = 0 \\ V'^{pot}_a, & X_a^q > 0 \end{cases}$$

Whether free flow vehicles can reach the link head or not, is not known beforehand. There are two ways to find out if free flow vehicles can reach the link head. The first is to find out if the travel time through the queue is shorter than the time step. The second way is to calculate the number of vehicles that could reach the link head assuming an infinite queue, and then checking if the actual number of vehicles in queue is less. Both require some additional steps to calculate $V'^{pot}_a$ as can be seen in Figure 5.4 and Figure 5.5. All grey boxes in the figures represent linear interpolation steps, that are CPU demanding. Assuming that the queue will mostly have more vehicles than the potential outflow, it is wise to use option two. This option is usually done after two steps and sometimes after three, whereas option one is usually done after three steps and sometimes after two.

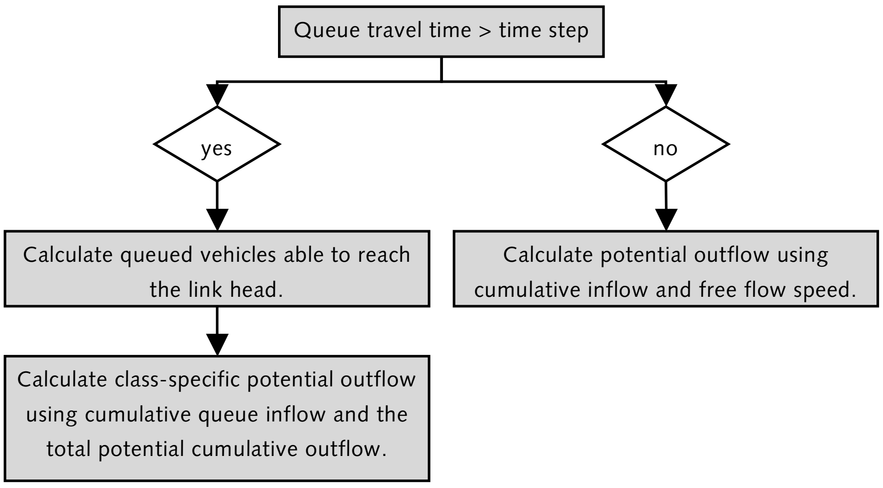
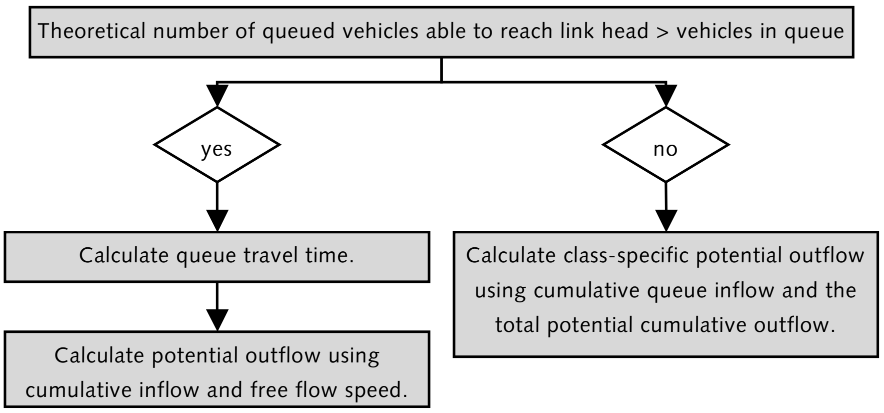

Queue travel time > time step
yes
no
Calculate queued vehicles able to reach the link head.
Calculate potential outflow using cumulative inflow and free flow speed.
Calculate class-specific potential outflow using cumulative queue inflow and the total potential cumulative outflow.

Figure 5.4: Option 1 to calculate $V_a^{q\prime}$.
Usually three steps are performed, sometimes two steps.

Figure 5.5: Option 2 to calculate $V_a^{q\prime}$.
Usually two steps are performed, sometimes three steps.



Theoretical number of queued vehicles able to reach link head > vehicles in queue
yes
no
Calculate queue travel time.
Calculate potential outflow using cumulative inflow and free flow speed.
Calculate class-specific potential outflow using cumulative queue inflow and the total potential cumulative outflow.

Option two is mathematically represented in Equation 5.11. Note that the class dimension (*m*) is now included. The steps will be further explained.

Equation 5.11

$$V'^{pot}_{am} = \begin{cases} X'^{q}_{am}, & X'^{q}_{a} \leq X^{q}_{a} \\ X'^{f}_{am}, & X'^{q}_{a} > X^{q}_{a} \end{cases}$$

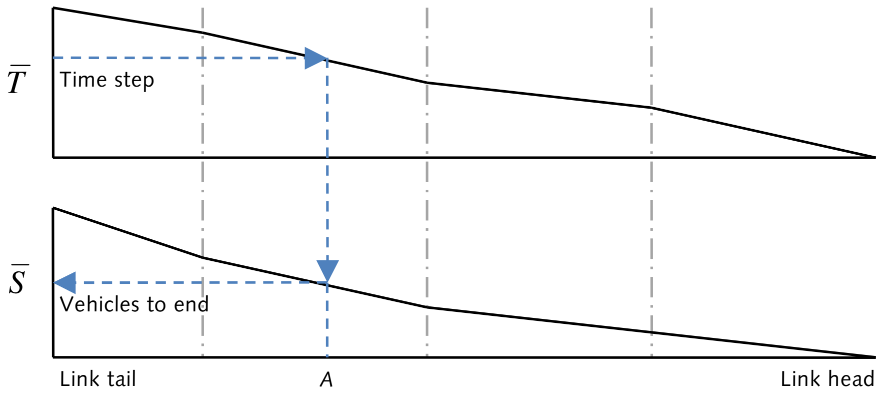
The first step of option two assumes an infinite queue to determine the maximum number of queued vehicles that can reach the link head within a time step ($X'^{q}_{a}$). By linear interpolation, this number can be found.

Equation 5.12

$$X'^{q}_{a} = \overline{S}_a\left(\overline{T}_a^{-1}(\Delta t)\right)$$

This is graphically represented in Figure 5.6.

Figure 5.6: Linear interpolation from *T* to *S*



If the actual number of vehicles in queue is larger, indeed all vehicles beyond location *A* are queued. It is however not known how many vehicles from each class form this flow. To acquire potential outflow for class *m*, $\tau$ should be solved from Equation 5.13 (a) and put into Equation 5.13 (b). Equation 5.13 (a) describes the following:

- Cumulative queue inflow at time *t*-$\tau$ equals cumulative flow at *A* at time *t*, where $\tau$ represents the travel time through the queue up to point *A*. Note that this travel time is related to past queue lengths and can thus not be determined from the current queue length.
- Cumulative flow at *A* equals cumulative outflow plus the number of vehicles between point *A* and the link head ($X'^{q}_{a}$).
- From the previous two statements, cumulative flow at point *A* can be removed, resulting in Equation 5.13 (a).

Equation 5.13 (b) rearranges Equation 5.13 (a) but as all known variables are now class specific, the unknown ($X'^{q}_{am}$) can be made class specific.

Equation 5.13

$$\text{(a)}\quad X'^{q}_{a} + \sum_{m} \bar{V}_{am}(t) = \sum_{m} \bar{Q}_{am}(t-\tau)$$

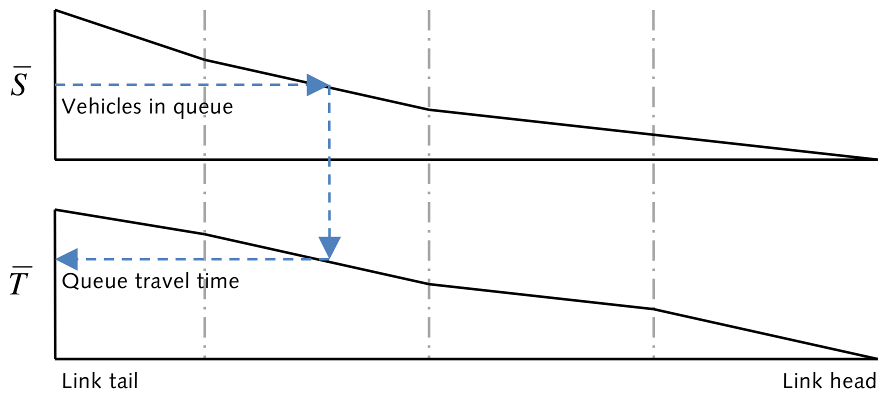
$$\text{(b)}\quad X'^{q}_{am} = \bar{Q}_{am}(t-\tau) - \bar{V}_{am}(t)$$

If the actual number of vehicles in queue is less than $X'^{q}_{a}$, a few free flow vehicles need to be included, giving $X'^{f}_{a}$. These vehicles can travel some distance at free flow speed and then go through the (short) queue to reach the link head, all within a time step. The queue travel time can be linearly interpolated, see Equation 5.14 and Figure 5.7.

Equation 5.14

$$T^{q}_{a} = \bar{T}_{a}\left(\bar{S}^{-1}_{a}\left(X^{q}_{a}\right)\right)$$

Figure 5.7: Linear interpolation from S to T



The time remaining ($\Delta t'_{a}$) for the free flow section is given by the time step minus queue travel time.

Equation 5.15

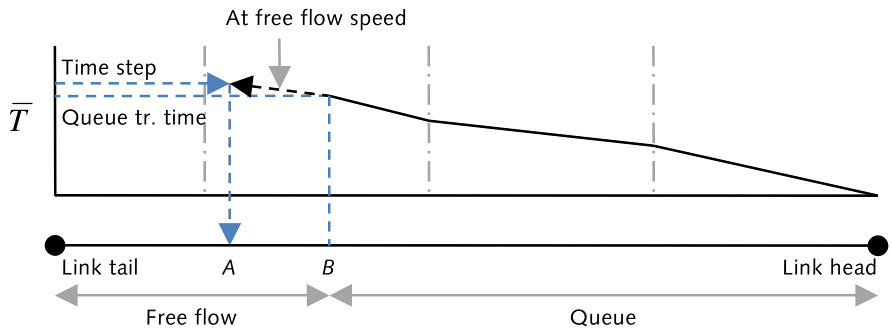
$$\Delta t'_{a} = \Delta t - T^{q}_{a}$$

The free flow section from where vehicles are able to reach the link end within a time step is defined as lying between points *A* and *B* where point *B* is the start of the queue (see Figure 5.8). The length of this section is equal to what vehicles at free flow speed can transverse in the time step remainder $\Delta t'_{a}$.

Equation 5.16

$$L_a^{AB} = \Delta t'_a \cdot \vartheta_a^{\max}$$

Figure 5.8: Determination of location *A*



As the queue length is known, the location of point *A* is now defined. Similarly to cumulative queue inflow at point *B*, a cumulative number of vehicles at point *A* can be determined. The (non cumulative) number of vehicles between point *A* and the link head is equal to the difference between cumulative inflow at point *A* and cumulative link outflow. Using an equation similarly to Equation A.13 and subtracting cumulative link outflow, the correct number of vehicles that can reach the link head is known, see Equation 5.17. Note that $\tau$ is one period extra as we need current cumulative flow at *A* and not cumulative flow for *t*+1.

Equation 5.17

$$X'^{f}_{am} = \overline{U}_{am}(t - \lceil \tau \rceil) + \left[(\lceil \tau \rceil - \tau) \cdot \{\overline{U}_{am}(t - \lfloor \tau \rfloor) - \overline{U}_{am}(t - \lceil \tau \rceil)\}\right] - \overline{V}_{am}(t)$$

Finally, links that are affected by the hazard have an absolute maximum number of vehicles, as all later vehicles are unable to continue. The new cumulative link outflow may not exceed this value.

### 5.1.3. Short links

As can be learned from the next chapter about the changes to the node model, conflicts at intersections are additional constraints that may limit the flow over a node. The evaluation of node conflicts requires nodes that resemble actual intersections. Common with macroscopic networks is to model a cluster of intersections as a single node, such as intersections that are simply very close to one another or intersecting grade separated highways where many weaving sections, on ramps and off ramps may be found. It may be opted to use nodes that represent multiple intersections. Such nodes should be defined as type 'none', conflicts will then not be included. It remains the responsibility of the user to verify that conflicts at the specific intersections are not significant for the desired model outcome. A better option in terms of accuracy would be to model each actual intersection as a node. For the described situations this usually involves short links that are a problem for the DNL. Links that can be traversed within a time step are inconsistent with the relation between link inflow, queue inflow and link outflow. The node model thus desires the possibility to deal with short links. As described by Taale (2008) there are two methods to circumvent this problem. The first is to shorten the time step. This may be undesirable concerning the calculation time of the model. The second method is to virtually lengthen the links so that the travel time is equal to a time step. This would influence the travel time, but as the time step is short, the deviation is small. Taale additionally introduces steps to perform accurate spillback by using a critical link length that is thus different from the virtual link length. A similar approach is performed here. CBQ is performed as for all links. The maximum number of congested cells, hence the maximum inflow and potential outflow, represent the actual link length. However, queue inflow (is link outflow if there is no queue) is interpolated from link inflow using the virtual length to determine the travel time towards the queue or the link head. This thus introduces slight delays but enables the relationship between the flows. Practically the virtual lengthening can easily be implemented by using the current time step as a maximum limit for the time step where inflow is determined. As this is already performed in the link model to accurately model fully congested links, nothing actually has to change. Only an error message about short links has been removed now that the method for short links has been theorized. This creates a dynamic virtual length that is equal to the queue length and the length that can be traversed within a time step at free flow speed together. The dynamic length follows from the use of queue inflow, of which the location is dynamic. Most DNL models only use link inflow and link outflow, which are at fixed locations.

### 5.1.4. Numerical example

A numerical example for one link will now be shown. The link and model parameters are:

| | |
|---|---|
| Time step: | 20 s / 0.005556 h |
| Length: | 0.35 km |
| Maximum speed: | 50 km/h |
| Lanes: | 1 |
| Capacity: | 2000 pcu/h |
| Saturation flow: | 1500 pcu/h |
| Jam density: | 150 pcu/km |
| Wave speed: | 18.1818 km/h |
| Last cell factor: | 0.465 (last cell is not as long) |
| In the queue: | 20 pcu |
| On the link: | 25 pcu |

The link has 4 cells for which the various traffic states are calculated as below. Outflow *V* has been deduced from past link outflow. Density and speed are derived from the triangular fundamental link diagram.

| | | | | |
|---|---|---|---|---|
| *V* (pcu/*Δt*) | 7 | 6 | 5 | 7 |
| *V* (pcu/h) | 1260 | 1080 | 900 | 1260 |
| *K* (pcu/km) | 80.7 | 90.6 | 100.5 | 80.7 |
| *W* (km/h) | 15.6134 | 11.9205 | 8.9552 | 18.5874 |

Note that the first cell (to the right) has higher speed than the last cell although their densities and flows are equal. This is because the first cell has its speed based on the maximum of its flow and saturation flow. From the cell states we can calculate travel time and storage by also using the cell length where the last cell has a different length.

| | | | | |
|---|---|---|---|---|
| $L$ (km) | 0.0470 | 0.1010 | 0.1010 | 0.1010 |
| $T$ (h) | 0.0030 | 0.0085 | 0.0113 | 0.0054 |
| $S$ (pcu) | 3.7905 | 9.1515 | 10.1515 | 8.1515 |

The length, travel time and storage are made cumulative.

| | | | | |
|---|---|---|---|---|
| $L$ (km) | 0.3500 | 0.3030 | 0.2020 | 0.1010 |
| $T$ (h) | 0.0282 | 0.0252 | 0.0167 | 0.0054 |
| $S$ (pcu) | 31.2450 | 27.4545 | 18.3030 | 8.1515 |

To calculate the queue inflow we need the queue length. As there are 20 pcu in queue, the queue length spans the first 2 cells and partially the 3rd cell. From the cumulative length we can interpolate to find $L^q$ = 0.2208 km. With cumulative link inflow the queue inflow can be determined but this is omitted here. Next the maximum inflow is calculated. Capacity equals 11.1111 pcu/$\Delta t$ but the remaining storage is 31.2450 – 25 = 6.245 pcu. The latter is the minimum and thus the maximum inflow. Finally the potential outflow needs to be determined as the number of vehicles that can reach the end of the link. First, all vehicles are assumed to be in queue. The time step is just over the travel time of the first cell and the potential outflow would thus be just over the number of pcu in the first cell. Indeed the interpolation results in 8.2606 pcu. As this is indeed less than the number of pcu in queue, the assumption holds and this is potential outflow.

## 5.2 Conclusions

The queue representation of CBQ has been explained. The derivation of maximum inflow and potential outflow has been presented as being dependant on the cell states. In this way the shockwave theory is implicitly included. The link model as described relies on fundamental diagrams. These are assumed to be triangular. A triangular fundamental diagram is defined by three points, one of which is the origin. The capacity point is described by the capacity value and the free flow speed. The last point is the jam point. Flow equals zero, a jam density needs to be given. The fundamental diagrams are thus given by three parameters per link: capacity, free flow speed and the number of lanes, which will be multiplied with a network wide jam density per lane. Note that an explicit capacity gap is omitted in the fundamental diagram. There is however a limit on congested outflow for the first cell. This limit requires a saturation flow per link. Additional parameters for the link model are thus network wide jam density per lane and a saturation flow per link. Saturation flow should be determined carefully. Effects from the node and effects from the link should clearly be distinguished.

For instance, a 2-lane link ending at an intersection with 4 turn lanes should have a saturation flow in the order of magnitude of 2x1800 pcu/h and not 4x1800 pcu/h. Saturation flow is the maximum outflow if the link itself is congested. Turn lanes do not increase this. They are designed to buffer stochastic turn flow fluctuations. Flow in EVAQ is not stochastic and so turn lanes are insignificant for the link model. Generally, the node is not of influence for saturation flow. Saturation flow is thus only different for links if human behaviour on the links is different. These differences may have to do with level of relaxation and the ability to see downstream flow recovery (anticipation).

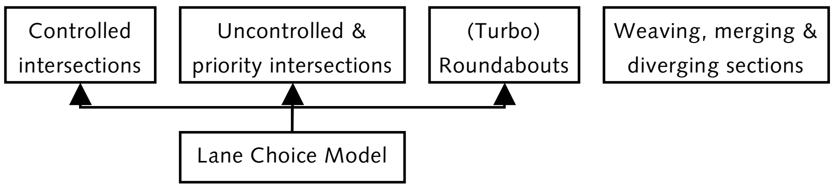
The next two chapters will elaborate on the new node model. The new link and node model will be evaluated in chapter 8.

# 6. Node model improvements

Chapter 4 has presented and selected solutions to improve the DNL model of EVAQ with respect to NPD. Chapter 5 has explained the new link model in detail. This chapter will do the same for the node model. The main task of the node model is to check the potential outflows to constraints by maximum inflow and of the node itself. This results in actual link outflows and link inflows. First, in section 6.1 a new modelling framework is explained for the node model. Next, the Lane Choice Model (LCM) is explained. The LCM is a sub model of other node type specific sub models. Sections 6.3 till 6.5 elaborate on the sub node models for controlled intersections, uncontrolled & priority intersections and (turbo) roundabouts respectively. Another sub node model is for weaving sections. This sub node model is presented in chapter 7. It is dealt with separately, as it is entirely new while the sub node models of this chapter are all based on existing models and formulas. Also the weaving model does not use the LCM. A last node type that will not be discussed is the 'None' node type as this simply means that constraints on the node will not be checked. This can be useful for origin/destination nodes or nodes that connect connectors to the network while there is no actual intersection but rather multiple streets connecting to a single road. At the end of this chapter some conclusions are given.

## 6.1 New node model framework

In section 3.4.2 it is explained that the current node model only deals with the maximum inflow of the connecting links and is ignorant to conflicts on the node itself. Depending on the node type (intersection, roundabout, etc.) the conflicts have different mechanisms and different consequences. The differences are of such an extent that we cannot speak of a single node model. In fact, each node type has its own model. Common among three of the sub-models is the LCM as displayed in Figure 6.1.


Controlled intersections
Uncontrolled & priority intersections
(Turbo) Roundabouts
Weaving, merging & diverging sections
Lane Choice Model
Figure 6.1: Sub-models of the node model

The current node model in EVAQ will stay active, as maximum link inflow still needs to be balanced with potential link outflow. The new node models will limit flow further if constraints are violated. The order of practice will be:

- Determine potential directional flows by applying split fractions from the route choice model on potential outflow.
- Determine secondary potential directional flows using the sub node models.
- Determine actual directional flows as a result of the critical maximum link inflow.

The application of split fractions on potential outflow may produce directional flows in all directions, including a U-turn. In normal circumstances this may not be valid as U-turns inhibit some resistance. For evacuations however it may be expected that drivers are not very sensitive to this. Some movements over nodes may not be possible, such as a U-turn on certain turbo roundabouts and a left-turn from an on-ramp. The latter can be dealt with by applying separated nodes for divided roads. The first can only be dealt with by having a prohibition matrix for the node. A complication is however that split fractions add up to one. Leaving out a certain movement from a link will violate traffic conservation unless the remaining split fractions are recalculated. Therefore, split fractions at nodes with a prohibition matrix will be recalculated for each link. Note that this recalculation is not performed by the route choice model, but is simply an up scaling of the valid split fractions. Currently, the node model does not accept separated nodes for divided roads as EVAQ is based on a bi-directional network. Because of this some definitions will change. Currently, so called joint-nodes are recognized if the number of outgoing links is two, as this would then not be an intersection but more a shape point defining road curvature of a two-way road. Intersections have three or more downstream links. These definitions assume a network with only bi-directional roads. In reality there may be directional roads. Joints are thus actually nodes where there is one upstream and one downstream link. All other nodes form junctions. For route choice to be performed, the number of downstream links should at least be two. New node definitions are applied:

- *Junctions* are all nodes except origins and destinations. This may include dead-end nodes, which should not exist in the first place. Junctions are modelled as having inherent capacity reduction by the sub node models, unless their type is 'None'. For all junctions the maximum inflow is checked.
- *Route nodes* are nodes that have at least two downstream links, including origin nodes but excluding destination nodes. The route choice model is applied for these nodes. Note that the route set generation also uses these nodes.

Currently EVAQ accepts only one downstream link from an origin node. All departures are put on this link. EVAQ will be changed to allow multiple connector links for which the route choice model is applied. Strangely, the route model is already applied for origin nodes.

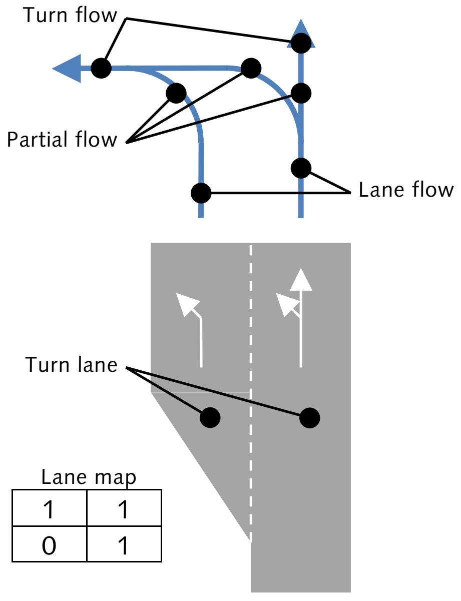
The second step in the node model is the application of the sub models. This is done before the maximum inflow checks for two reasons:

- Flows may interact on the node in a skewed way (priority). If a downstream link should limit flow, all flow should be limited by the skewed ratios. Therefore, the skewed flows should be calculated first.
- Applying the maximum inflow constraints first would change the absolute size of flows that may therefore not accurately represent interaction on the node.

Constraints of the third step, and also some constraints of the second step, will reduce all flow over the node with the same factor. Durlin & Henn (2007) use a generalized merge model based on the merge model by Daganzo (1993) that uses outflow division fractions based on number of entering lanes, priority etc. This is neglected here as it is assumed that potential link outflow will be representative for the number of lanes of a link while the sub node models will deal with such things as priority.

## 6.2 Lane Choice Model

The LCM is the first step in three sub node models. The LCM determines how drivers select turn lanes at the end of links. This behaviour is important as the use of multiple and/or shared lanes towards a link is of great influence in the mechanisms of the sub node models. This will become evident in the three following sections describing the sub node models that use the LCM. Before the LCM is explained, a few definitions of common terms will be given.



| | |
|---|---|
| 1 | 1 |
| 0 | 1 |

Figure 6.2: LCM definitions

A *turn flow* is all flow from one link to another. A *turn lane* is a lane at the end of a link that can be used to turn certain ways over the node. *Lane flow* is all flow from a turn lane. *Partial flow* is part of a lane flow belonging to a specific turn flow.

Which turn lanes can be used to reach the downstream links is given in a *lane map*. The lane map is an *nxm* matrix where *n* is the number of downstream links and *m* is the number of turn lanes. It is mostly filled with zeros and has a '1' for each link reachable from each turn lane. For instance the first element is '1' indicating that the first (left) turn lane can be used to turn to the first (U-turn) link. Each next '1' is in the same row and next column, in the next row and the same column or the next row and the next column. This prohibits turn conflicts that are not allowed, following intersection design practice. A *block* is a lane map combined with turn flows that will divide over the turn lanes. Finally, the *flow-matrix* is a matrix similar to the lane map but filled with actual partial flows. The rows sum up to turn flows and the columns sum up to lane flows. The purpose of the LCM is to translate a block into a flow-matrix.

As the name suggests, the LCM is a choice model describing how drivers choose their lane. This choice is a stochastic process, but it is simplified to a deterministic problem. This consideration is reasonable as only capacity conditions are considered and crucial. These circumstances form a strong incentive to road users to divide equally over the turn lanes. As long as the level of intersection knowledge is large enough among the road users, turn lane use will be balanced. This principle is very similar to the Wardrop principle [Wardrop (1952)] concerning route choice. In fact, here an adapted version of the Wardrop principle is used. The Wardrop principle is based on origins and destinations with routes between them. People choose their route in a way that all used routes have equal and minimal travel time. For the LCM the link is the (only) ‘origin’, the downstream links are the ‘destinations’ and the turn lanes are the ‘routes’. Turn lane selection is not performed by route travel time as in the Wardrop principle, but by flow. This assumes that drivers experience a lower volume on an adjacent lane as an incentive to change to that lane. The larger the volume difference, the larger the incentive. The net result is equal lane use. In reality drivers cannot observe flow. They can however observe density. Assuming the lane speeds to be equal creates a direct relationship with flow. The Wardrop principle is adapted as followes:

1. All used turn lanes towards a downstream link have equal total flow, including flow towards other downstream links on the shared turn lanes.
2. All possible but unused turn lanes towards a downstream link have more total flow than the used turn lanes.

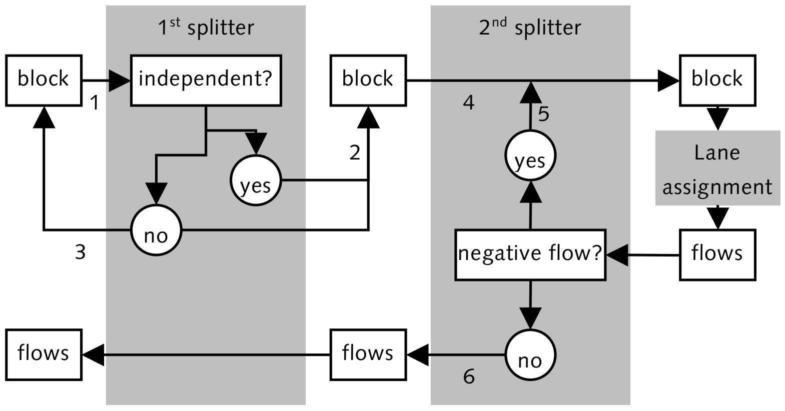
The adapted Wardrop principle assumes that intersections downstream have no influence on lane choice, nor do reductions in lanes just downstream of the intersection. In reality this is often not true if the intersection or lane reduction is nearby. Another assumption is that turn lanes have no length. The buffering effect is not included and the link itself determines total potential outflow. Turn lanes here merely serve as a flow separation. If these assumptions are reasonable or not, is subject for further research.

Solving a route choice problem analytically is very complex, even on small scale. Therefore an algorithm will be used. This algorithm is different than for regular route choice problems. Instead of solving the entire problem iteratively, the problem is split into independent sub-problems that can be solved analytically. There are 2 splitters to derive independent sub-blocks that form sub-problems. Independence means that there is no overlap between blocks. This can be defined as:

- No turn lane within an independent block is used by a turn flow that is outside of the block.
- No turn flow within an independent block uses a turn lane that is outside of the block.

The algorithm is displayed in Figure 6.3 and starts at (1) with a block of the entire intersection. The first splitter separates independent blocks based on the lane map. A new block starts if the next turn flow has no shared lane with the last turn flow. If there are no shared turn lanes and only dedicated turn lanes, each turn flow is a separate sub-block. If an independent block is found, it is forwarded to the second splitter (2). The remainder of the original block, if any, is fed back into the first splitter (3). The second splitter finds additional independence, as partial flows may be zero. This complies with the definition of block independence. The zero-partial flows are found by trying to assign flow to the turn lanes (4). The lane assignment may return a flow-matrix with negative partial flow(s). It is clear that negative flows are impossible and so the assignment has failed. This can be solved by assuming the most negative partial flow to be zero[1]. The block is split into 2 sub-blocks that are both fed back into the lane assignment (5). If all flows are positive, flows are consistent and will be returned (6). The splitters split blocks but merge resulting flow-matrices, resulting in a single flow-matrix for the entire link.

Figure 6.3: LCM framework



The first splitter splits blocks independent of flow, and it is therefore not needed to apply the first splitter every time step. Instead, split locations will be stored during the model initialisation.

The core of the algorithm is the lane assignment. Here, blocks are translated into flow-matrices. Based on the 1st adapted Wardrop principle, all lanes are considered to have equal lane flow, which is average lane flow. The assignment runs through the lane map starting at the first element and works a way to the lower-right corner. For every partial flow, one of the following steps is performed:

[1] Assuming any or all negative partial flows to be zero can produce inconsistency with the adapted Wardrop principle. The most negative partial flow can be assumed to be zero as it is most dominated by other partial flows. This has been made plausible by running many randomly generated blocks through the algorithm. All were verified to comply with the adapted Wardrop principle without having negative partial flows.

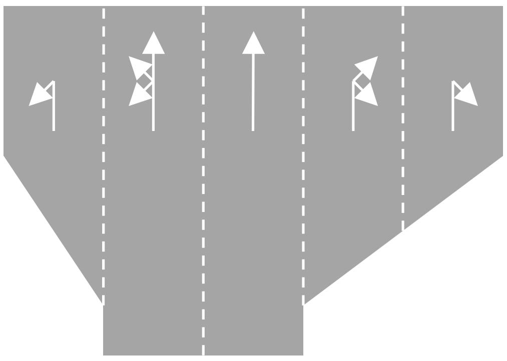
- If the last (only) turn flow of a turn lane is reached, the remainder of average lane flow is assigned.
- If the last (only) turn lane of a turn flow is reached, the remainder of current turn flow is assigned.

These steps will become more apparent in the LCM example.

### 6.2.1. Lane Choice Model example

An example block will be run through the LCM. The U-turn is omitted for clarity. The turn lanes are as in Figure 6.4.

Figure 6.4: Example turn lanes



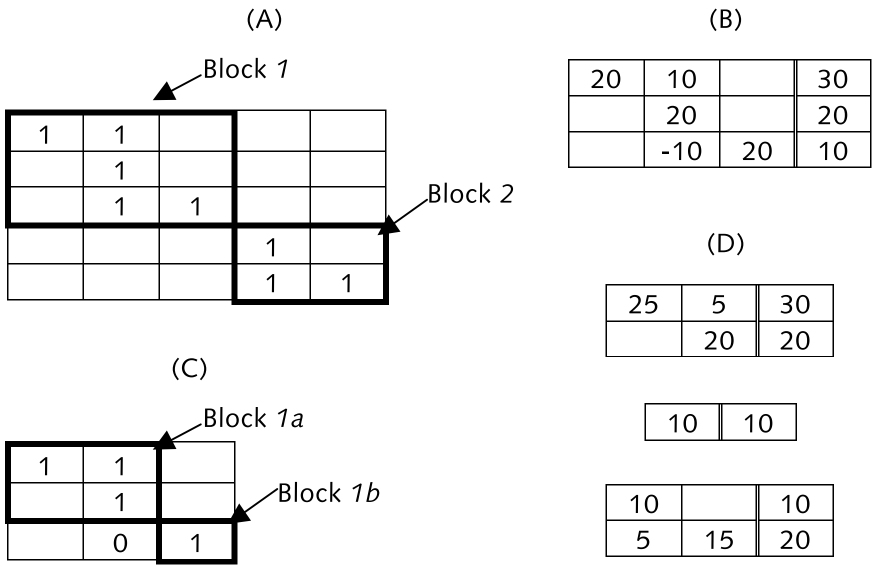
Together with turn flows into the 5 possible turn directions this results in the following block.

Table 6.1: Example block

| | | | | | |
|---|---|---|---|---|---|
| 1 | 1 | | | | 30 |
| | 1 | | | | 20 |
| | 1 | 1 | | | 10 |
| | | | 1 | | 10 |
| | | | 1 | 1 | 20 |

The 1st splitter finds independent sub-block *1* since the next partial flow has no shared lane with the last partial flow, see Figure 6.5 (A). It is forwarded to the 2nd splitter. The remainder (block *2*) is re-analysed by the 1st splitter and found to be completely dependent. Block *2* will thus also be forwarded as such to the 2nd splitter.

Figure 6.5: Blocks as split by the splitters



(A)

| | | | | |
|---|---|---|---|---|
| 1 | 1 | | | |
| | 1 | | | |
| | 1 | 1 | | |
| | | | 1 | |
| | | | 1 | 1 |

(B)

| | | | |
|---|---|---|---|
| 20 | 10 | | 30 |
| | 20 | | 20 |
| | -10 | 20 | 10 |

(C)

| | | |
|---|---|---|
| 1 | 1 | |
| | 1 | |
| | 0 | 1 |

(D)

| | | |
|---|---|---|
| 25 | 5 | 30 |
| | 20 | 20 |

| | |
|---|---|
| 10 | 10 |

| | | |
|---|---|---|
| 10 | | 10 |
| 5 | 15 | 20 |

Block *1* will be run through the lane assignment. Average lane flow is (30+20+10)/3 = 20. Since the upper-left partial flow is the last (only) turn flow of the first turn lane, this must be the average lane flow of 20, see Figure 6.5 (B). The partial flow to the right is the last turn lane of the first turn flow and must be 30 – 20 = 10. The next partial flow is the last (only) turn lane of the second turn flow and is thus equal to the second turn flow, which is 20. The next step will generate infeasible results. The second lane already has a lane flow of 30 while average lane flow equals 20. The next partial flow will be -10 to compensate. The final partial flow can be derived in two ways, both resulting in 20. All row-sums of block *1* are equal to the turn flows (30, 20, 10) and all column-sums are equal to the average lane flow of 20. However, partial flows cannot be negative. Assuming that the (most) negative flow should actually be zero, block *1* can be split into sub-blocks *1a* and *1b* is in Figure 6.5 (C). Note the explicit '0' splitting the sub-blocks. Blocks *1a*, *1b* and *2* are run through the lane assignment resulting in Figure 6.5 (D). The partial flows together form this flow-matrix:

Table 6.2: Example flow-matrix

| | | | | | |
|---|---|---|---|---|---|
| 25 | 5 | | | | 30 |
| | 20 | | | | 20 |
| | 0 | 10 | | | 10 |
| | | | 10 | | 10 |
| | | | 5 | 15 | 20 |
| 25 | 25 | 10 | 15 | 15 | |

From this it can be seen that all lane flows within the sub-blocks are equal and that unused turn lanes have higher lane flows (25 over 10). This complies with the adapted Wardrop principle. All row-sums are equal to the turn flows. The partial flows will be used in the sub node models described in the remainder of this chapter.

## 6.3 Controlled intersection model

Controlled intersections have traffic lights separating conflicting turn flows in time. A conflict point between crossing turn flows should thus be seen as a turn-by-turn use of infrastructure. The separation in time goes further than two crossing turn flows. Groups of multiple turn flows can be identified in which each turn flow crosses all other turn flows within the group. All turn flows in a group thus have to take turns using the infrastructure. The infrastructure has some capacity that forms a limit for the turn flows together.

Turn flows that can use multiple turn lanes have multiple conflict points with conflicting turn flows. Luckily, for every pair of turn flows, only one conflict point is crucial. This is the conflict point where the highest partial flows cross. Remember that partial flows are a part of a turn flow that is specific to a certain turn lane. Also for conflict groups, only these crucial conflict point need to be taken into account. Figure 6.6 shows that if the conflicts indicated by the dots are not constraining, than all other conflicts are also not constraining as they have less flow.

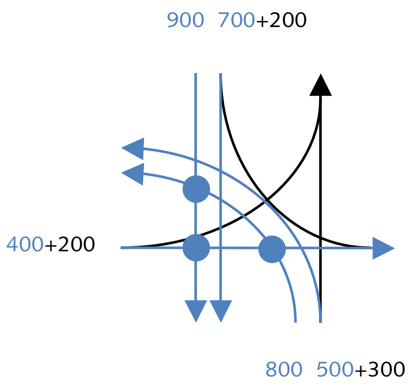


Figure 6.6: Conflict group

For each conflict group with maximum partial flows $p = 1...P$ the following general constraint holds:

Equation 6.1

$$\sum_{p=1}^{P} q_{p,\max} \leq C_{conflict}$$

The sum of all maximum partial flows ($q_{max}$) in a group may not exceed conflict capacity ($C_{conflict}$). As there is hardly any interaction between the turn flows on controlled intersections, the conflict capacity can be considered to be equal to effective saturation flow at traffic lights. The base value is 1800 pcu/h. As mentioned earlier insection 3.4.2, saturation flow can be reduced by many factors. Although most reductions are small, their aggregated effect should not be ignored. A representative reduction is made up from a peak hour reduction (~0,9) and a turn reduction where half of the flows are turning (~0,9). These reductions are taken from the program VRIGEN, which is a Dutch traffic control design program. Additionally there is also time loss at controlled intersections. This follows from yellow time and clearance time for safety. The effective time fraction for a standard cycle length of two minutes is about 0,9 (Syllabus CT4822). The effective saturation flow with all 3 reductions thus becomes $1800*0.9^3$ ~ 1300 pcu/h. Setting this as conflict capacity assumes that at any time one of the partial flows in the conflict group has a green light. Depending on the number of green phases, the number of flows in the conflict group and right turns, this may not be true. Figure 6.7 shows a few situations that describe the ratio between conflict size and effective green phases. Part (A) shows an intersection with four roads. Such an intersection has conflicts with three or four phases. As all 3-phase conflicts hold a right turn, each green phase facilitates the conflict. Note that this is only possible if the right turn has only dedicated turn lanes. Two possible phasing schemes of four phases are shown, both facilitate the conflict equally. Part (B) shows the same intersection where westbound traffic is not possible. This creates a 2-phase conflict without a right turn that is not a subset of any other conflict. Again, no matter what specific phasing scheme is chosen, they perform equally. As there is no right turn in the conflict, only two out of four phases facilitate the conflict. Finally part (C) shows the same intersection as in part (B) but with a 3-phase group. Just as in part (A) all 3-phase groups have a right turn and are thus facilitated by all green phases if the right turns have only dedicated turn lanes.

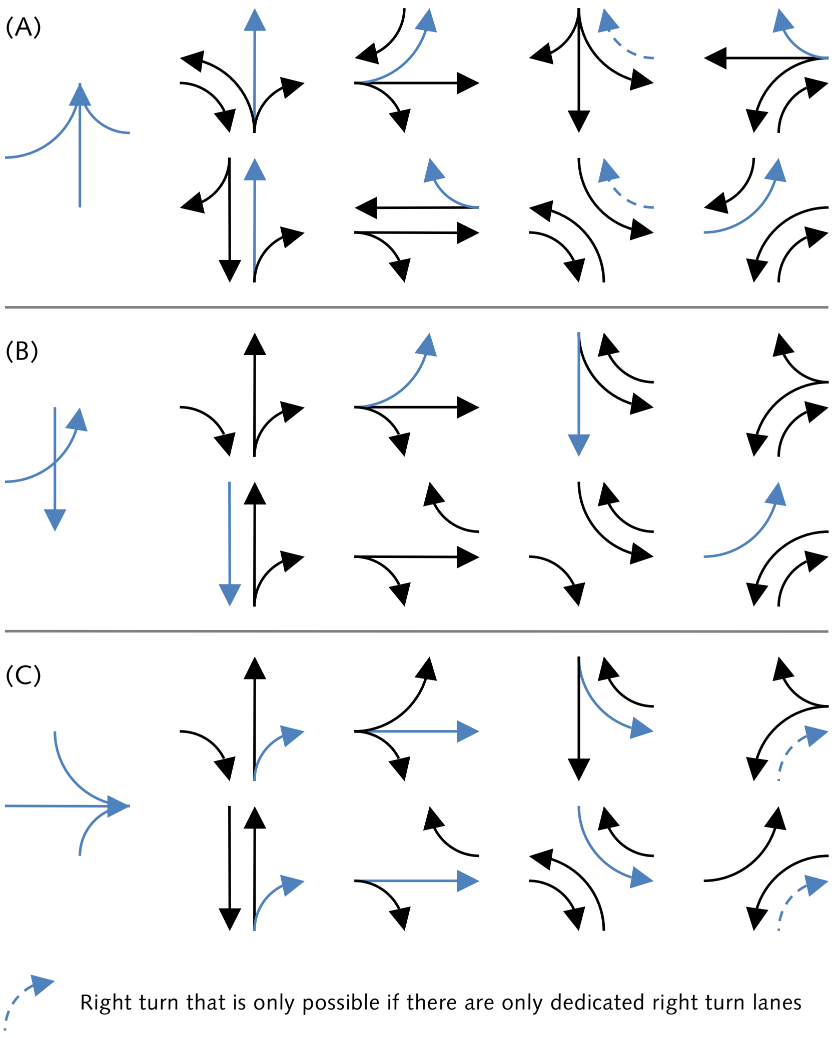

(A)
(B)
(C)
Right turn that is only possible if there are only dedicated right turn lanes

Figure 6.7: Conflict size versus green phases
(A) Valid 3-phase conflict with right turn, each green phase facilitates the conflict.
(B) Valid 2-phase conflict without right turn, 2 out of 4 green phases facilitate the conflict.
(C) Valid 3-phase conflict with right turn, each green phase facilitates the conflict.
Valid: not a subset of any other conflict.

The principle of Figure 6.7 can be generalized to the following:

- The number of green phases is equal to the largest conflict size.
- Conflicts with a size equal to the number of green phases are facilitated by all green phases.
- Smaller conflicts *with* a dedicated right turn are facilitated by all green phases.
- Smaller conflicts *without* a dedicated right turn can use only the number of green phases equal to their own size.

For the last set of conflicts, the hourly capacity is not 1300 pcu as the conflict has a green light for less than an hour. Depending on the green times of the phases the effective time could be anywhere between 0 and 60 minutes per hour. However, should the conflict be critical it may be expected that the green times are balanced towards the specific green phases of the conflict given that optimised control is assumed. The worse case would thus be an equal demand (green time) per green phase if multiple conflicts are more or less critical, giving an effective time of $s_c/n_{gf}$ where $s_c$ is the size of the conflict and $n_{gf}$ is the number of green phases. To derive the phase demand, the control scheme needs to be known. This sort of information is not easily obtained for all controlled intersections in a reasonably sized network. It is better to apply an average reduction factor, which is half way between optimal and worse conditions giving an effective capacity as in Equation 6.2.

Equation 6.2

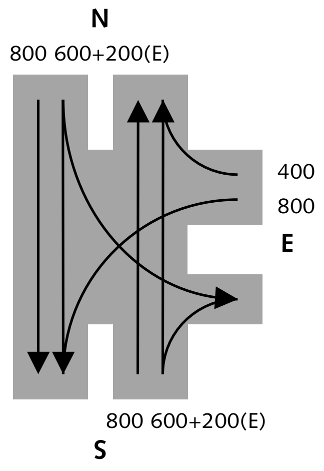
$$C_{conflict} = \begin{cases} 1300 \cdot \dfrac{s_c + n_{gf}}{2 \cdot n_{gf}} & s_c < n_{gf} \quad and \quad no\ dedicated\ right\ turn \\ 1300 & otherwise \end{cases}$$

To check if any conflict group violates the general constraint, the LCM is used to derive partial flows. Next, for each link pair (*i*,*j*) the maximum partial flow ($q^{ij}_{max}$) is listed in the maximum partial flow matrix. Another matrix of the same size holds the total turn flows ($q^{ij}$) per link pair. Constraints are defined as a summation of certain elements of the maximum partial flow matrix. The conflict group that has the highest saturation will be reduced if necessary. Along with it, all turn flows and partial flow that come from the same links are reduced. Other links are not influenced as vehicles wait on the links and not on the node if they are faced with a red light. Still, also on controlled intersections vehicles are often seen standing on the intersection, but this is because of a reduced maximum link inflow (spillback). This should not be confused with conflict points that limit green time and therewith link outflow. After this initial reduction of flows, again the conflict group that has highest saturation will be reduced, but again only if necessary. This continues until all conflict groups are at or below conflict capacity.

Given that each individual conflict is not violating the general constraint does not mean that the actual capacities are known. Conflicts are namely dependant on each other, as different conflicts should assume the same share of green time for a specific green phase that is part of both conflicts. If we assume there are four phases and one critical conflict needs a green-time distribution of 10%-15%-40%-35% while another needs 25%-10%-20%-45% it is obvious that together they need 25%-15%-40%-45%, which is 125% in total. A reduction of 0,8 should thus be applied to the green times. This directly translates to a reduction of 0,8 of the turn flows. Depending on the green phases and the demand distribution, such a reduction can have a large range. A single representative value for all situations does not exist. Earlier it was chosen to not include the control scheme as this is much input that is difficult to obtain. Instead it is assumed that if we use a green phase for all flows per link, a representative reduction will be found. Each critical conflict will list its green time distribution over the relevant links. The maximum green time fractions of all links are added. If the total is above one, all turn flows will be reduced accordingly.

### 6.3.1. Example of the controlled intersection model

The model will be shown for an example intersection with six links (three roads) as shown in Figure 6.8. At the entering links the hourly partial flows are shown as calculated by the LCM. Conflict groups and the conflict capacities are given.



| Conflicts groups | cap. |
|---|---|
| 1. NS ES | 1138 |
| 2. NE SE | 1138 |
| 3. SN EN | 1300 |
| 4. NE ES SN | 1300 |

Figure 6.8: Example of a controlled intersection

For an intersection with 3 roads where all movements are possible, there are in total 4 conflict groups. In the example, 2 of the conflict groups have demand higher than capacity.

$$q_{NS,\max} + q_{ES,\max} = 800 + 800 = 1600 > 1138$$
$$q_{NE,\max} + q_{ES,\max} + q_{SN,\max} = 200 + 800 + 800 = 1800 > 1300$$

As the first conflict group has highest saturation, all partial flows from the accompanying links *N* and *E* will be reduced by a factor of 1138/1600 = 0,71. The green time distribution is 50% for *N*, 50% for *E* and 0% for *S*. Two partial flows of the second over saturated conflict group are also reduced, resulting in a different conflict group demand.

$$q_{NE,\max} + q_{ES,\max} + q_{SN,\max} = 142 + 568 + 800 = 1511 > 1300$$

As the reduced conflict group demand of the second group is still higher than conflict capacity, partial flows from *N*, *E* and *S* are now reduced by a factor of 1300/1511 = 0,86. Partial flows from *N* and *E* are reduced in total by a factor of 0,71·0,86 = 0,61. The green time distribution of this conflict has lower fractions than the previous conflict except for link *S* where 53% is needed. In total the two conflicts need 50+50+53 = 153%. A reduction of 1/1,53 = 0,65 follows for all flows. All resulting partial flows and reductions are shown in Figure 6.9. These partial flows are still consistent with the assumptions in the LCM as there is one net reduction factor for all partial flows from a link. These changes do not influence the relative ratios between partial flows from a link. Figure 6.9 also shows that links can be affected by conflict capacity into different extends.

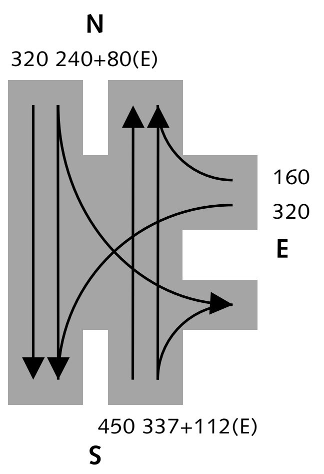


Figure 6.9: Example of a controlled intersection with reduced partial flows

| Reduction factors: | |
|---|---|
| N: | 0,40 |
| E: | 0,40 |
| S: | 0,46 |

### 6.3.2. Permitted conflicts

So far, only fully controlled intersections without a U-turn have been covered. U-turns have been omitted for the simple reason that traffic lights do not account for them. In this context the U-turns could be viewed as permitted conflicts. Besides U-turns there may be explicit permitted conflicts. These are often encountered as a right turn bypass or permitted left turns with small flows. As U-turns only conflict with turn flows towards the same link, it is expected that the capacity of the link will properly deal with these flows. For explicit permitted conflicts, several solutions may be applied:

- For right hand bypasses an additional link could be created. The right turn at the intersection itself should then be omitted.
- Permitted conflicts can be modelled as a non-permitted conflict. The consequence of this would be that the number of green phases might increase which in turn will decrease capacity of conflict groups that are not facilitated by each of the green phases.
- Permitted conflicts could be omitted all together. This is useful if the expected flows performing the permitted turns are very low. The influence of modelling such a conflict as non-permitted might be worse than omitting the turn. Note however that the influence of modelling a permitted conflict as non-permitted is non-existent for regular balanced intersections with dedicated right turn lanes as all conflict groups have a capacity of 1300 pcu/h.

## 6.4 Uncontrolled and priority intersection model

Uncontrolled intersections have no traffic lights and drivers have to regulate infrastructure use by themselves. This is achieved by right-of-way rules. In the Netherlands, traffic from the right has right-of-way. Priority intersections also have no traffic lights. What is different from uncontrolled intersections is that one road always has right of way, indicated by road signs and road markings. For both intersection types it holds that each turn flow has a (possibly empty) set of other turn flows that need to get right-of-way. This is also the framework for the determination of turn flow capacity for both intersection types.

In the Syllabus CT4822, a model is described that determines the capacity of a minor flow (needs to give right-of-way) intersecting with a major flow (gets right-of-way). The model is based on headway distribution in the major flow and gap acceptance of the minor flow. The major flow headways are assumed to be exponentially distributed. The model is computed with Equation 6.3.

Equation 6.3

$$C_{minor} = \frac{\exp(-q_{main} \cdot t_{critical})}{h}$$

where,
$t_{critical}$ = minimum gap acceptance
$h$ = average headway between following vehicles

A property of the exponential distribution is the ability to add major flows into a single value. This is proven in Equation 6.4 where *f* is a reduction factor based on maximum minor capacity.

Equation 6.4

$$\begin{aligned}
&C_{minor} = \\
&C_{\max} \cdot (f_1 \cdot \ldots \cdot f_m) = \\
&\frac{\exp(0)}{h} \cdot \left( \frac{\exp(-q^1_{main} \cdot t_{critical})/h}{\exp(0)/h} \cdot \ldots \cdot \frac{\exp(-q^m_{main} \cdot t_{critical})/h}{\exp(0)/h} \right) = \\
&\frac{1}{h} \cdot \left(\exp(-q^1_{main} \cdot t_{critical}) \cdot \ldots \cdot \exp(-q^m_{main} \cdot t_{critical})\right) = \\
&\frac{\exp\left(-\sum_{n=1}^{m} \{q^n_{main}\} \cdot t_{critical}\right)}{h}
\end{aligned}$$

The capacity of each turn flow is thus defined by the sum of the accompanying major flows. As turn flows can be minor related to some turn flows and major related to others, the capacities cannot be determined directly if flows from the same time step need to be used. This will require an iterative solution method. Note however that the model from the Syllabus CT4822 was derived for hourly averages. If an hour is concerned, delay time between major vehicles entering the intersection and minor vehicles being influenced are insignificant. If however a small time step is concerned, it is evident that not all vehicles will actually interact within the time step. Therefore it is considered to be equally accurate to use turn flows from the previous time step. Minor turn flow capacity can thus be obtained directly.

Using the previous time step as such could potentially introduce unstable oscillations between time steps. For instance assume we have an empty intersection. If at one moment much traffic would reach the intersection from several directions, these flows would be unrestricted, as the previous time step had no flow. This unrestricted flow may be well above capacity, resulting in no capacity for the time step after that. A 2-step cycle will thus exist where much flow allows no flow, and no flow allows much flow. To overcome this unstable behaviour, major flows are determined by the average of the previous two time steps. Still, the first step with traffic would exceed capacity, but at least the oscillation is filtered in only a few time steps. This phenomenon only occurs with large changes in demand for an intersection from several directions at the same time step.

For normal circumstances the minimum gap acceptance is about four seconds and the average headway for following vehicles is about two seconds. During evacuations it can be expected that the minimum gap acceptance is lower. Very low values can also represent the unwillingness of minor turn flows to actually give way. A minimum gap of zero seconds more or less creates a 50%-50% ratio between the major and minor flow.

Up to now the phrase minor/major flow has meant a (sum of) turn flow. This is not accurate as turn flows can share lanes with other turn flows and/or use multiple lanes. A minor flow should thus be a combination of partial flows that use the same turn lane and are faced with the same set of major flows. The major flow can still be taken from turn flows. If a turn flow is conflicting, all of its partial flows are conflicting. Each constraint is thus a group of minor partial flows $p$ = 1...$P$ and major turn flows $f$ = 1...$F$ as input in Equation 6.5.

Equation 6.5

$$\sum_{p=1}^{P} q_p \leq \frac{\exp\left(-\sum_{f=1}^{F}\{q_f\}\cdot t_{critical}\right)}{h}$$

The following steps need to be performed to acquire the groups:

1. For each turn lane and for each possible combination of partial flows from that turn lane, find the common major turn flows of the partial flows.
2. Discard minor and major sets that are a subset of another minor and major set.

These steps are performed once prior to the model run. For a simple priority intersection with a 1-lane minor road, Figure 6.10 shows all resulting groups of step one. The bottom row can be merged as they all have the same major turn flow (only one actually). The last group is equal to the merged group. Figure 6.11 shows merged groups if the minor road would have two lanes.

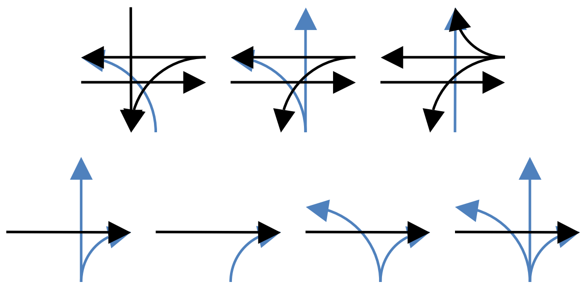


Figure 6.10: Major/minor groups on a priority intersection for a 1-lane minor road.

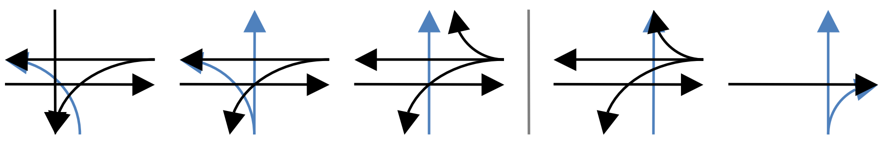


Figure 6.11: Merged major/minor groups on a priority intersection for a 2-lane minor road.

The LCM is used to derive the partial flows. For all merged groups the general constraint is checked. If the minor flows exceed capacity, all flows (also outside of the group) from the link will be reduced. The reduction is based on the most saturated group per link.

## 6.5 Roundabout model

### 6.5.1. Regular roundabouts

Roundabouts are characterized by a one-way circular road to which all roads connect. In the Netherlands, traffic on the roundabout has right of way. Roundabouts can have one or two lanes. A relatively new phenomenon is turbo-roundabouts, but these will be covered in the next section. Roundabouts could be placed in the framework of uncontrolled and priority intersections, as a group of minor partial flows has to give right of way to major turn flows. This would however generate poor results. First of all, the LCM is incapable to divide intersecting traffic at multi-lane links. Intersecting traffic is traffic on the right lane going left and traffic on the left lane going right (or through). Also, the so-called pseudo conflict is of significant influence, which the uncontrolled and priority intersection model cannot deal with. A pseudo conflict is caused by traffic that is perceived as staying on the roundabout, but actually exits the roundabout. Drivers reaching a roundabout are not always able to distinguish what drivers on the roundabout will do. For the Dutch situation a model developed by Cetur (1986) and adapted by Bovy (1991) [Yperman & Immers (2003)] is able to determine entrance capacity for various layouts. A division of traffic over turn lanes is not necessary. The model explicitly deals with the pseudo conflict and the number of lanes on the roundabout and on the link. The model is based on circulating flow (conflicting) and exit flow (pseudo conflict).

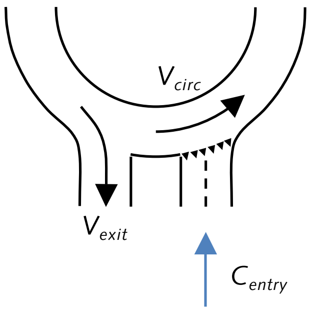


Figure 6.12: Roundabout entrance capacity framework

The framework is given in Figure 6.12. The used formula can be seen in Equation 6.6.

Equation 6.6

$$C_{entry} = \frac{1500 - \frac{8}{9}\left(\beta V_{circ} + \alpha V_{exit}\right)}{\gamma}$$

where,

$\alpha$ = Pseudo conflict influence (0 – 0.8)
$\beta$ = Influence of number of lanes on the roundabout (0.6 – 1.0)
$\gamma$ = Influence of number of lanes on the entrance link (0.6 – 1.0)

The numerical ranges for the parameters were recommended by Bovy. The value for $\alpha$ is related to the distance between C and C′ as shown in Figure 6.13. The graph shows a range depending on the exiting flow rate and the circulating speed. The flow rate is of influence because drivers tend to use their direction indicator more with higher flow rates. The speed is of influence as the distance C-C′ is covered in less time. The value of $\alpha$ could be made dependent of exit flow and roundabout diameter (speed). There is however no validated method to do this. Representative values will be used (thick line in Figure 6.13).

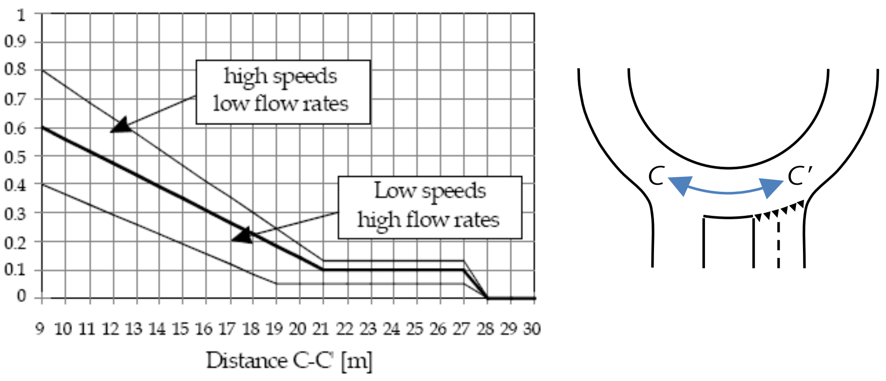


Figure 6.13: Relation between distance C-C′ and $\alpha$.

Similarly to the model from the Syllabus CT4822 for minor flow capacity, the model from Bovy is for hourly averages determined with an iterative method. The time step is however very small and it cannot be stated that equilibrium holds for traffic within a single time step. It is considered equally accurate to use the flows from the previous 2 time steps as with uncontrolled and priority intersection, enabling a direct determination of entrance capacity for all links. All that is needed per link is a set of turn flows that make up $V_{circ}$ and $V_{exit}$ and the parameters. The sets of turn flows and parameters can be determined prior to the model run. Often the values for $\beta$ and $\gamma$, and sometimes also for $\alpha$, are the same for all connecting links and can be coupled to the node. Links that have no individual parameters can use the parameters of the node.

### 6.5.2. Turbo roundabouts

At the time the model from Bovy was developed, turbo-roundabouts did not exist. Today they are increasingly common in the Netherlands. In terms of capacity and conflicts, the following differences can be identified:

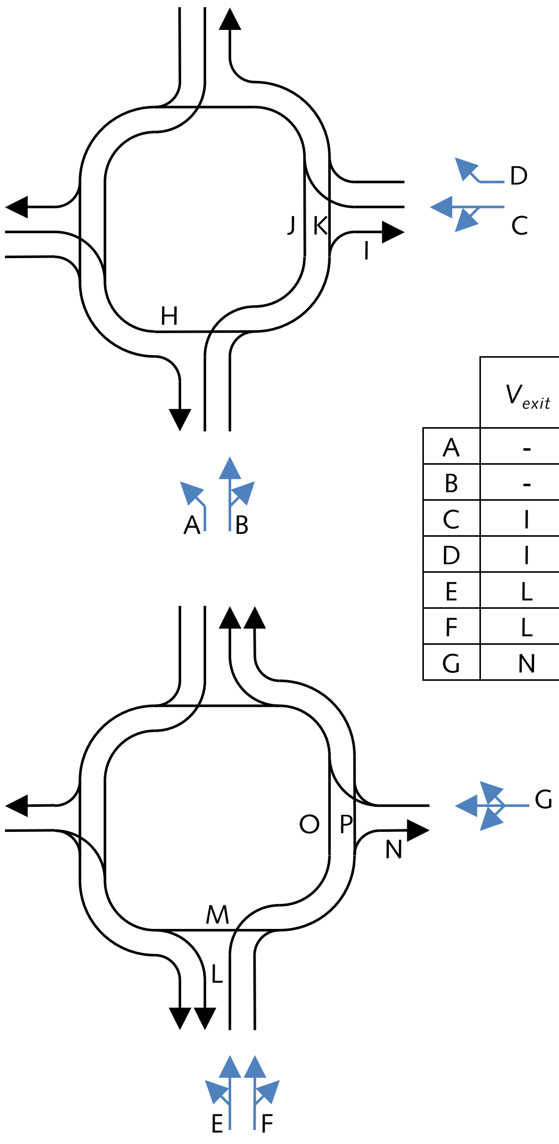
- Drivers have to select a turn lane before entering the roundabout.
- When entering the roundabout, drivers can be faced with zero, one or two roundabout lanes, depending from which link and which lane they enter the roundabout.
- Related to the conflicting roundabout lanes, drivers entering the roundabout have different sets of partial flows that form $V_{circ}$.

Looking at the differences it can be observed that the model is unusable on a link level. Indeed Yperman & Immers (2003) state that new formulas should be derived from existing empirical formulas, but coefficients could not be calibrated, as there are not many turbo-roundabouts with the same layout. For this reason Yperman & Immers (2003) used micro simulation. Here it is identified that the basics of a roundabout are not different at a turbo-roundabout, only the layout is. On lane level the same method could be applied. Figure 6.14 shows how input can be defined for each lane, accounting for the various types of turbo-roundabout approaches. Lanes *A* & *B* but also *E* & *F* have equal properties. Still the lanes need to be modelled individually as turbo-roundabouts force the use of turn lanes, that is, the lanes have individual demand. The LCM is used to derive this demand. Also, the partial flows from the LCM form the flows at *H* till *P*. For example, flow *L* is the partial flow from the north to the south and flow *H* is all partial flows from the north and west to the east and north. An interesting fact to note is that few entrance lanes have no $V_{exit}$, possibly contributing to the capacity difference often witnessed between 2-lane and turbo roundabouts. All lanes have, evidently, one lane 'at the link' as the model is performed at lane level for turbo-roundabouts. The value for $\gamma$ is therefore always one.

Just as with uncontrolled intersections, as soon as a link or lane capacity is exceeded, all flows from the link are reduced by a single factor.

De Leeuw (1997) developed an extension on the model from Bovy that included the influence on capacity of slow traffic (cyclists), which is very common in the Netherlands. Slow traffic is not a part of EVAQ and is therefore excluded.

Figure 6.14: Lane specific properties at turbo-roundabouts



| | $V_{exit}$ | $V_{circ}$ | Lanes at roundabout |
|---|---|---|---|
| A | - | H | 1 |
| B | - | H | 1 |
| C | I | J+K | 2 |
| D | I | K | 1 |
| E | L | M | 1 |
| F | L | M | 1 |
| G | N | O+P | 2 |

## 6.6 Conclusions

Based on existing models and formulas, models have been developed and explained for controlled intersections, uncontrolled & priority intersections and (turbo) roundabouts. The LCM is a common sub model without parameters that has been developed to model the lane choice behaviour at intersections and turbo roundabouts. Generally the existing models are used in frameworks that relate all flow over an intersection. The parameters of the existing models are already calibrated for normal circumstances. Calibration for evacuation circumstances is difficult and beyond the scope of this research. In chapter 8 it will be evaluated if output of the new models resembles data of the microscopic model VISSIM. The next chapter will first elaborate on the newly developed weaving model. This model will need to be calibrated. The calibration is also performed in the next chapter.

# 7. Weaving model

This chapter relates strongly to the previous chapter. In the previous chapter the new node model and node type specific sub models have been explained. This chapter additionally explains the newly developed weaving model. The reason for this development is the lack of a generally accepted weaving model. First, two existing models from the 1985 HCM and by Rakha & Zhang (2006) are discussed. The theoretical background and drawbacks are mentioned. As these models are empirical and layout specific, a flexible theory-based model is desired. The new weaving model is explained in section 7.2. In section 7.3 the new weaving model is calibrated. At the end of this chapter some conclusions are made.

## 7.1 Existing weaving models

Weaving sections are found at highways where an on-ramp is closely followed by an off-ramp (<1500m). The on-ramp and/or off-ramp can also be replaced with a 2nd highway. Vehicles may have to change lanes to arrive at the desired downstream link. Merge sections are equal to weaving sections but without the off-ramp. Diverge sections are equal to weaving sections but without the on-ramp. Merge and diverge sections can be seen as special cases of a weaving section. They can be modelled by a weaving section model if demand for or from the missing ramp is zero. No generally accepted model to determine weaving section capacity exists. Models do exists such as the 1985 HCM approach described in the Syllabus CT4821 and a model by Rakha & Zhang (2006). The latter appears to be able to determine capacity with reasonable accuracy. These models do however have drawbacks. The models are layout specific and are determined empirically. If a layout is found that is not dealt with in these models, there is no way to interpolate or extrapolate the underlying mechanism of capacity reduction to the new layout. Both models assume that capacity reduction is caused by 'turbulence' by the lane changing behaviour. The net capacity is measured either from reality or a microscopic model, and the result is mathematically fitted to functions that are thought to describe the following factors influencing the amount of turbulence:

- Demand pattern
- Speed differences between merging roads
- Percentage of trucks
- Weaving section length

The demand pattern is often specified as having a volume ratio (weaving flow over total flow) and weaving ratio (smallest or off-ramp weaving flow over all weaving flow). Speed differences influence deceleration behaviour and therewith the amount of turbulence. However, at capacity conditions, it can be assumed that drivers adapt their speed to more or less match the speed on the other road. The effect of speed differences thus disappears for capacity estimations. Rakha & Zhang (2004) show that speed differences are indeed not a significant factor. They also show that the percentage of trucks can be translated into pcu by the 2000 HCM method. For now this is not necessary as EVAQ has only one mode, being passenger cars. Weaving section length is thought to force lower speed to compensate the reduction in time to weave. These lower speeds eventually lead to lower flows.

Both models describe capacity as the flow just prior to traffic breakdown. The assumptions made about the state of the influencing factors seem valid for capacity conditions. However, that these factors are significantly related to turbulence prior to traffic breakdown is unlikely. Prior to breakdown traffic is in free flow state, a traffic state that is not characterized by turbulence. In other words, the turbulence is insignificant prior to traffic breakdown and can thus not be a determining factor of capacity. Turbulence would also mean that vehicles need more space resulting in a lower capacity per lane. In reality it is often seen that at weaving sections there are lanes that actually have more vehicles than on regular sections. For a short while at least, drivers may accept very small gaps.

## 7.2 New weaving model

Looking at traffic on weaving sections both in reality and microscopic simulations, a more logical cause, also related to the previously mentioned factors, can be found. This cause is the choice of lane that drivers make before the weaving section. It is often witnessed that the right lane of the highway is used at capacity, while other lanes are not. This is because traffic that wants to leave the highway prefers this lane. Also traffic that stays on the highway might not experience a very strong incentive to not use the lane, especially heavy vehicles. If a weaving section is short, the preference to use the right lane increases. With similar mechanisms the demand for all lanes at the weaving section can be explained. Depending on the layout and demand pattern, it may be another lane that is critical. Generally we can state that weaving section capacity is reached whenever a lane at or just before the weaving section is used at capacity.

Here, a new model is developed that looks at lane demand. It is thought that drivers will experience some utility for the lane they select. In fact, drivers have a preferred movement over the weaving section and will select the lane that enables them to make this movement. Any movement is defined as a combination of an entrance and an exit lane, using the least lane changes needed in between. As drivers may deviate from their preferred movement on the weaving section, the utility only describes the initial preference and not the actual movement. Still, the lane demand at the start of the weaving section may be determined with these utilities. As soon as drivers have finished their desired movement, additional lane changes are often performed to evenly distribute lane demand at the downstream side of the weaving section. That is, per downstream link.

### 7.2.1. Weaving section capacity factors

As mentioned earlier, speed differences and the percentage of trucks can be excluded. The demand pattern and weaving section length are also excluded as explained below. As no factors remain, capacity will be a fixed value that may not be exceeded by demand.

*Demand pattern*

Movement utility is expected to be insensitive to demand, as only saturated conditions are important. The model therefore does not describe under-saturated conditions and behaviour. Utility being insensitive to demand may seem counter-intuitive, but the demand pattern still defines capacity. Demand is namely divided by the utilities. The more weaving traffic, the more weaving traffic assigned to the best weaving movement, the higher the lane demand, the lower the weaving section capacity.

*Weaving section length*

Weaving section length is thought to be an influence on the utility itself. However, for weaving sections that are designed following the Dutch standards (see NOA), the lengths are such that the influence is often insignificant [Vermijs (1998)]. Note that the negative influence of the weaving section length is large for very short weaving sections, but quickly reduces as weaving section length increases. Therefore the Dutch standards avoid short weaving sections in the first place. Still, the length may be significant in some cases, but this will be considered later on. For now the weaving section length is ignored.

### 7.2.2. Weaving movement utility factors

Factors contributing to the utility for the movements have not yet been recognized. An obvious factor is the number of lane changes that needs to be made. Each lane change has some disutility ($u_{lc}$). Complicating design features are tapers at both the merge and diverge locations. For example at the merge taper, many vehicles may perform a lane change. Vehicles on the taper lane either merge with the main road or select the right lane. Vehicles on the right lane of the highway may also change a lane to the left. Lane capacity thus needs to be considered after the merge taper. The utility should describe the net utility of all choices made prior to this point, including the taper. If there is no merge taper, this is equal to the utility just before the weaving section. If there is a merge taper, the downstream lane that it merges with will have disutility ($u_{ta}$). This disutility will deviate traffic from the taper lane, but also from the right lane of the highway. Note that this disutility does not describe the preferred movement prior to the weaving section, but prior to the critical section as in Figure 7.1. The utility per movement from lane $i$ to lane $j$ can be calculated as in Equation 7.1.

Equation 7.1

$$u_{ij} = \left| i + \nabla_{pre}(i) - j + \nabla_{post}(j) \right| \cdot u_{lc} + d_0 \cdot u_{ta}$$

with,

$$d_0 = \begin{cases} 1 & i \in \{r_t - 1, r_t\} \\ 0 & otherwise \end{cases}$$

where,

$\nabla_{pre}$ Adjustment array for the lane number translating the lane number before the critical section to the lane number inside the critical section. This translation is needed to account for a taper.

$\nabla_{post}$ Adjustment array for the lane number translating the lane number after the critical section to the lane number inside the critical section. This translation is needed to account for a taper.

$d_0$ Dummy variable, one if lane *i* is the merging taper lane or merging with it, zero otherwise.

$r_t$ Number of the merging taper lane, zero if there is no merging taper lane.

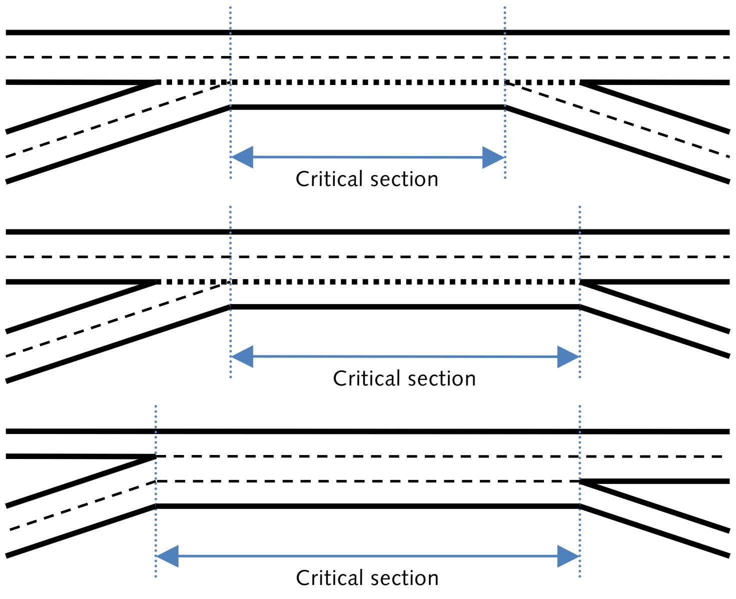


Figure 7.1: Critical section related to tapers

### 7.2.3. Flow distribution and peak demand

The utilities can be represented in an *m*-by-*n* matrix where *m* is the number of lanes entering the weaving section (including a possible taper lane) and *n* is the number of lanes leaving the weaving section (also including a possible taper lane). For each from- and to-link combination, the actual flows performing certain movements can be found using the logit model in Equation 7.2, assuming the utilities to be independent and identically Gumbel distributed.

Equation 7.2

$$q_{ij}^{AB} = q^{AB} \cdot \frac{d_1 \cdot \exp(\lambda \cdot u_{ij})}{\sum_i \sum_j d_1 \cdot \exp(\lambda \cdot u_{ij})}$$

with,

$$d_1 = \begin{cases} 1 & pre(i) = A,\ post(j) = B \\ 0 & otherwise \end{cases}$$

where,

$q_{ij}^{AB}$ Flow from link *A* to link *B* making the movement from lane *i* to lane *j*.

$q^{AB}$ All flow from link *A* to link *B*.

| | |
|---|---|
| $d_1$ | Dummy variable, one if the movement is from link *A* to link *B*, zero if not. |
| $u_{ij}$ | Utility to perform the movement from lane *i* to lane *j*. |
| $\lambda$ | Logit scale factor, here equal to one. |
| pre | Array with road numbers (one or two) of lanes prior to the weaving section. |
| post | Array with road numbers (one or two) of lanes after the weaving section. |

Assigning flow to all utilities, we have an *m*-by-*n* matrix with flows. If there is a merging taper, two rows can be merged. Note that this addition is valid as only the net choices determining the situation at the start of the critical section are considered. Each row now describes the demand of a lane at the start of the critical section. Often, this is not the exact location where maximum lane demand is found. Especially for type B weaving sections, where one weaving flow has to perform at least two lane changes, the influence of vehicles that change over a lane is high and found somewhere downstream of the start of the critical section. The exact location is not important, the additional lane demand however is. In this context one would also think of other flows that move from or to a lane, but as stated earlier, the utilities do not describe actual movements, but rather preferred movements. Weaving traffic that changes over a lane is a certain demand for this lane while all other flows will tend to distribute around the peak demand (in space). An additional variable ($f_{co}$) is introduced, describing a fraction of weaving flow over a lane at the peak demand (at a non-fixed point in space) additional to the demand at the start of the weaving section, as it is suspected that the actual peak demand will be located close to the start of the critical section. It can be expected that $f_{co}$ will also capture some phenomena that are not explicitly modelled, such as traffic performing lane changes from the critical lane at the very start of the critical section. What is important is that it gives representative peak lane demand although precise behaviour at the critical section is uncertain. Weaving traffic performing at least two lane changes here thus creates an additional lane demand as in Equation 7.3.

Equation 7.3

$$D(r) = f_{co} \cdot \sum_i \sum_j \left( d_2 \cdot d_3 \cdot q_{ij}^{AB} \right)$$

with,

$$d_2 = \begin{cases} 1 & pre(i) \neq post(j) \\ 0 & pre(i) = post(j) \end{cases}$$

$$d_3 = \begin{cases} 1 & \begin{matrix} i + \nabla_{pre}(i) < r,\ j + \nabla_{post}(j) > r \quad or \\ i + \nabla_{pre}(i) > r,\ j + \nabla_{post}(j) < r \end{matrix} \\ 0 & otherwise \end{cases}$$

where,

| | |
|---|---|
| *D(r)* | Additional demand on critical section lane *r* by weaving flows over lane *r*. |
| $d_2$ | Dummy variable describing that a flow must be weaving. One if the movement from *i* to *j* is weaving, zero if not. |

| | |
|---|---|
| $d_3$ | Dummy variable describing that a flow must change over lane *r*. One if the movement changes over lane *r*, zero if not. |

If any lane's capacity is exceeded, all flows will be reduced by a single factor, assuming equal flow disruption for all lanes. The constraints are defined as in Equation 7.4.

Equation 7.4

$$\sum_{i \in \nabla'} \left( \sum_{j} \left( q_{ij}^{AB} \right) \right) + D(r) \leq C_{peak}$$

where,

| | |
|---|---|
| $\nabla'$ | Set of lanes where $i + \nabla_{pre}(i) = r$. This is usually one lane. |
| to | Only if there is a merge taper will this set be two lanes in order merge the demand of the right lane on the highway and the taper lane. |
| $C_{peak}$ | Lane capacity for the peak (rather than average) demand where the location (cross section) is not fixed but related to the variable location of the peak demand. |

Flow over a weaving section is often lower after traffic break down than just before traffic break down (capacity). Such a drop in flow should not be seen as an attribute of the weaving section itself, but rather as a capacity drop of the congested links. In other words, CBQ should deal with this. A representative value for saturation flow (maximum congested outflow) is thus of importance for links upstream of a weaving section.

### 7.2.4. Example of the weaving model

Parameters resulting from the calibration in section 7.3 are:

| | |
|---|---|
| $u_{lc}$ | = -0,95 |
| $u_{ta}$ | = -0,17 |
| $f_{co}$ | = 0,79 |
| $C_{peak}$ | = 3791 pcu/h |

An example will now be presented using the weaving section and demand to the left from Figure 7.2. None of the links' capacity is exceeded. The demand to the right is the reduced demand for which the calculations will be given.

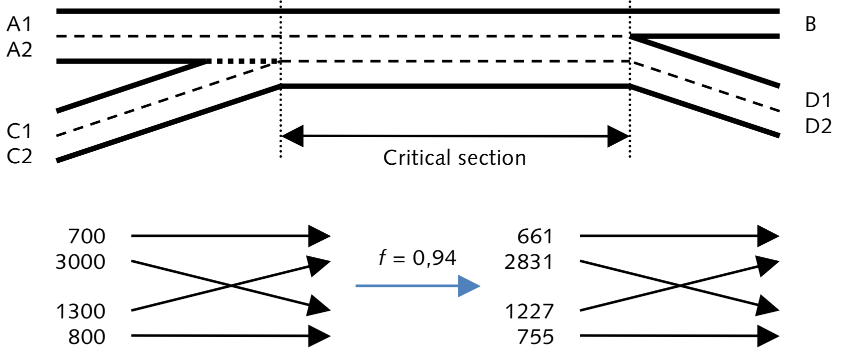


Figure 7.2: Example weaving section

The lane change utilities are given in Figure 7.3 (A). Note that the taper lane (C1) has equal lane change utility as A2. Figure 7.3 (B) shows the diverging effect as flows A2 and C1 come together. Figure 7.3 (C) is the total utility. Applying the logit model per link combination (boxes in thick lines), the flows as in Figure 7.3 (D) result. The three lanes at the critical section have a demand of 1471, 3487 and 843 pcu/h respectively. These are the row sums of table (D) where A2 and C1 are merged. The middle lane at the critical section has additional demand from weaving flows A1-D2 and C2-B as these change lanes over lane A2-D1. The additional demand equals 0,79x(263 + 409) = 531 pcu/h. Total lane demand now reaches 3487 + 531 = 4017 pcu/h. As this exceeds peak capacity, a reduction factor of 3791/4017 = 0,94 follows.

Figure 7.3: Utility and flow matrices

(A) Lane change utility

| | B | D1 | D2 |
|---|---|---|---|
| A1 | 0 | -0,95 | -1,90 |
| A2 | -0,95 | 0 | -0,95 |
| C1 | -0,95 | 0 | -0,95 |
| C2 | -1,90 | -0,95 | 0 |

(B) Merge taper utility

| | B | D1 | D2 |
|---|---|---|---|
| A1 | 0 | 0 | 0 |
| A2 | -0,17 | -0,17 | -0,17 |
| C1 | -0,17 | -0,17 | -0,17 |
| C2 | 0 | 0 | 0 |

(C) Total utility

| | B | D1 | D2 |
|---|---|---|---|
| A1 | 0 | -0,95 | -1,90 |
| A2 | -1,12 | -0,17 | -1,12 |
| C1 | -1,12 | -0,17 | -1,12 |
| C2 | -1,90 | -0,95 | 0 |

(D) Movement flows

| | B | D1 | D2 |
|---|---|---|---|
| A1 | 528 | 680 | 263 |
| A2 | 172 | 1483 | 574 |
| C1 | 891 | 264 | 102 |
| C2 | 409 | 121 | 313 |

## 7.3 Calibration of the weaving model

The weaving model was calibrated to data generated by Fosim (www.fosim.nl), which is a microscopic model validated for Dutch highways. For the calibration it is important to cover many combinations of input in order to let the model represent a wide range of possibilities. Input consists of the weaving section layout and the demand pattern.

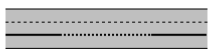
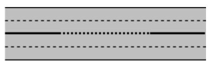
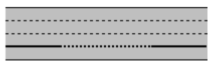
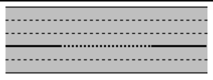
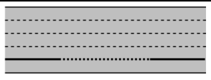
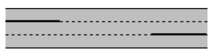
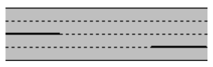
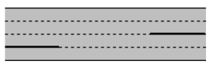
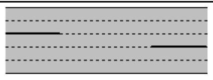
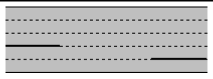
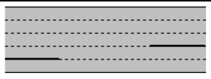
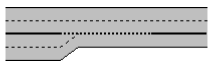
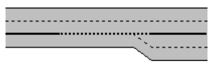
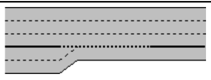
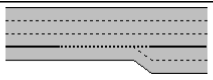
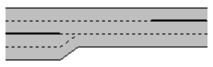
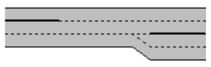
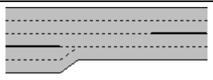
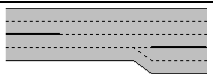
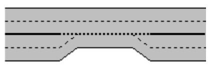
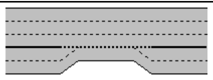
### 7.3.1. Weaving layouts

The selected layouts are all possible layouts following the rules below. These represent Dutch weaving sections (see NOA).

- The number of entering lanes is three, four or five.
- The number of exiting lanes is three, four or five, but never more than one lane different from the number of entering lanes.
- Links with a taper have two lanes.
- Links next to a link with a taper have at least two lanes.
- All directions have a minimum of two or less lane changes.

The resulting layouts are listed in Table 7.1. Five groups are distinguished based on layout similarity (not to be confused with capacity reduction similarity). Group *A* exists of standard weaving sections where the entering and exiting lanes are in balance and all weaving movements require at least one lane change. Group *B* layouts are similar to group *A* but the movement weaving to the right needs at least two lane changes while the other weaving movement requires none. Group *C* is similar to group *A* but there is one taper. Group *D* is similar to group *B* but also here there is one taper. Group *E* is similar to group *A* but with two tapers.

Table 7.1: Calibration layouts
*) length determined by similar layout

| | | |
|---|---|---|
|  A21 (250m) |  A22 (500m) |  A31 (417m) |
|  A32 (500m) |  A41 (500m) |  B12 (833m) |
|  B22 (625m) |  B22′ (833m) |  B23 (833m*) |
|  B32 (833m) |  B32′ (833m*) |  C22 (625m) |
|  C22′ (625m*) |  C32 (708m) |  C32′ (708m*) |
|  D22 (833m) |  D22′ (833m*) |  D32 (833m*) |
|  D32′ (625m*) |  E22 (625m*) |  E32 (708m*) |

All layouts also have a 2-digit number where the first digit indicates the number of entering lanes at the left link and the second digit indicates the number of entering lanes at the right link. A taper lane, if present, is also included within the second digit. Horizontally flipped (axial) counterparts are indicated by an apostrophe. For these the digits indicate the outgoing lanes. Vertically flipped (tangential) layouts are omitted as tangential movement to the right or left is considered equal. For example there is no B21 (nor a B12′ which is equal).

Weaving section lengths given in Table 7.2 were set at the default length for each configuration as in NOA. Some configuration were not listed and got a length equal to a similar layout.

### 7.3.2. Demand patterns

For each layout, various demand patterns need to be included in the calibration process. The left road goes from *A* to *C* and the right road goes from *B* to *D*. EVAQ uses split fractions at the nodes so the same fraction from *A* and *B* will go to C. The demand pattern can thus be defined by an *A*/*B* ratio and a *C*/*D* ratio. These ratios are rewritten into $\psi_A = q_A/(q_A+q_B)$ and $\psi_C = q_C/(q_C+q_D)$. The two remaining fraction are easily calculated as $\psi_B = 1 - \psi_A$ and $\psi_D = 1 - \psi_C$. The ratios follow some rules to prevent situations that either will not reach capacity, or will reach capacity of a downstream link rather than the weaving section:

- There is at least as much traffic as the minimum capacity of link C or D.
- Capacity of links A and B is not exceeded.
- Capacity of links C and D is not exceeded as this is captured by maximum link inflow.

Mathematically the rules form Equation 7.5.

Equation 7.5

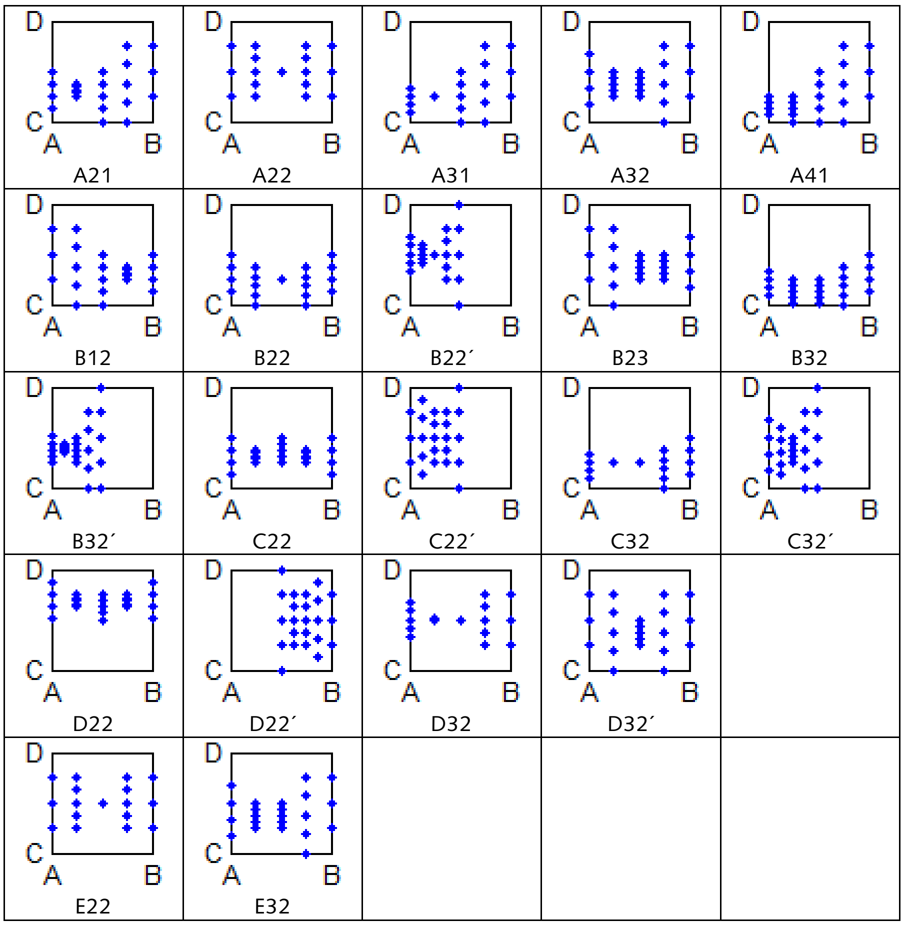
$$
\begin{gathered}
q_A + q_B \geq \min(C_C, C_D) \\
\max\left(1 - \frac{C_B}{q_A + q_B}, 0\right) \leq \psi_A \leq \min\left(\frac{C_A}{q_A + q_B}, 1\right) \\
\max\left(1 - \frac{C_D}{q_A + q_B}, 0\right) \leq \psi_C \leq \min\left(\frac{C_C}{q_A + q_B}, 1\right)
\end{gathered}
$$

Per layout, up to 25 demand patterns are investigated. First, $\psi_A$ is determined as five equidistant values spanning the allowable range. Next, either $q_A$ or $q_B$ is equal to the corresponding link capacity, whichever is critical. The other flow can be determined by the ratio and is less than or equal to its corresponding link capacity. Knowing the maximum flow at links A and B, the range for $\psi_C$ can be determined. It will also span the range with five equidistant values. In total this gives 5x5 = 25 demand patterns. Some are however excluded. The corners of the $\psi_A/\psi_C$ plane neither comply with the definition of a weaving section, nor with a merge, nor with a diverge section. All traffic comes from one link and goes to one link. The edges of the $\psi_A/\psi_C$ plane are included as these form merging and diverging situations since one link has zero flow. For some layouts and at certain values for $\psi_A$, the range for $\psi_C$ consists of a single value. These layouts will have four demand patterns less. All resulting ratios are given per layout in Table 7.2.

Information needed to determine the allowable demand pattern is the link capacity in Fosim. These were found by excluding trucks, using a maximum speed of 100 km/h, and having each of the three Fosim driver classes be present for 33,3%. The resulting capacities are 2879, 5796, 8715 and 11580 pcu/h for one, two, three and four lanes respectively.

Table 7.2: Calibration demand patterns for each layout.

In corner A, $\psi_A$ equals 1; in corner C, $\psi_C$ equals 1 etc.



### 7.3.3. Fosim runs and settings

Fosim is a stochastic model and multiple runs are needed to get some certainty about capacity. The number of runs required is 24, assuming a standard deviation of 250 pcu/h in the measured capacity and a certainty of 95% that the actual capacity is within a range of ±100 pcu/h [Dijker & Knoppers (2004)].

Settings in Fosim were set at the default settings for driver behaviour, including the lane change areas. Vehicle and driver composition are equal to the link capacity runs with each of the three user classes at 33,3% and no trucks. The speed is set at 100 km/h as this is common in the Netherlands at weaving sections. In order to retrieve information from Fosim, two detectors are put into place. The detectors return 5-minute averages of density, speed and flow. The first detector is located 500m upstream of the weaving section. Some congestion does not start at the actual weaving section but a bit upstream. This detector can register this congestion. A second detector is located at the start of the critical section. This detector measures flow able to enter the weaving section. Flow is increased from zero to the maximum flow as determined by Equation 7.5 in 90 minutes. After this, the maximum demand is maintained for 30 minutes. If at any of the detectors average speed over the lanes drops below 80 km/h, it is assumed that congestion has started. The maximum 5-minute flow before this time at detector two is taken as capacity. For each layout and for each demand pattern, the capacities returned by the 24 runs are averaged. Average standard deviation of capacity was found to be 266 pcu/h. 24 runs thus seems about enough as a standard deviation of 250 was assumed.

### 7.3.4. Calibration

Prior to the actual calibration, a grid-search was performed over the four input variables to be sure an absolute minimum of errors is found instead of a local minimum. The comparison between Fosim and the weaving model is done by flow reduction factors (*f*), which are one if congestion is not found and between zero and one if capacity was reached. Errors are defined as $|f_{weave}-f_{fosim}|/f_{fosim}$, which is an absolute relative error. Maximum error, average error and standard deviation are multiplied with one another to create a single performance indicator. This multiplication yields that an equal relative gain or loss of each individual performance indicator is equally bad or good. Absolute changes are filtered so that large absolute performance indicators do not get the overhand. The grid is defined as in Equation 7.6. A few coarse grid searches were performed to ensure a wide enough range.

Equation 7.6

$$\begin{aligned}
u_{lc} &\in \{-2, -1.8, -1.6, -1.4, -1.2, -1, -0.8, -0.6, -0.4, -0.2, 0\} \\
u_{ta} &\in \{-1, -0.9, -0.8, -0.7, -0.6, -0.5, -0.4, -0.3, -0.2, -0.1, 0\} \\
f_{co} &\in \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\} \\
C_{peak} &\in \{2500, 2750, 3000, 3250, 3500, 3750, 4000, 4250, 4500\}
\end{aligned}$$

The minimum grid value for the combined performance indicator has:

| | |
|---|---|
| Average error | = 5,40% |
| Maximum error | = 25,74% |
| Standard deviation | = 5,25% |
| $u_{lc}$ | = -1.0 |
| $u_{ta}$ | = -0.2 |
| $f_{co}$ | = 0.7 |
| $C_{peak}$ | = 3750 pcu/h |

Next, the weaving model was calibrated using a general minimization function in Matlab (fminsearch) with the grid minimum as an initial guess. The function feeds a set of variables in a 'black box' and receives a single result. By changing the input, the function 'reads' the black box and minimises the result. The black box here is actually a comparison between the reduction factors from Fosim and the weaving model. The comparison returns the combined performance indicator. The results of the calibration are:

| | |
|---|---|
| Average error | = 5,50% |
| Maximum error | = 24,03% |
| Standard deviation | = 5,36% |
| $u_{lc}$ | = -0,95 |
| $u_{ta}$ | = -0.17 |
| $f_{co}$ | = 0.79 |
| $C_{peak}$ | = 3791 pcu/h |

All parameters have the expected sign but peak capacity of 3791 pcu/h may be observed to be rather high. The value can be made plausible by the following reasons:

- *Small gaps*
  Drivers allow small gaps at weaving sections, at least for a very short period of time during lane changes by themselves or surrounding vehicles.
- *Complex process, simple model*
  The weaving process is a very complex process. The weaving model tries to mimic this process using only four variables. It may be expected that parameters will not have their actual value as some excluded weaving complexity will pull on the parameters. One particular excluded movement is any movement from the critical lane before the location of the peak demand. Would this be included, peak demand would be lower and capacity could also be lower resulting in an equal reduction factor.
- *Peak capacity vs. average capacity*
  Regular lane capacity values in the range of 2000-2200 pcu/h are derived with detectors at a fixed location. A small period of time, for instance 5 minutes, is aggregated to derive the average traffic state. From average values one cannot make conclusions about the maximum allowable demand for a few seconds. The latter is however exactly what the lane demand of the weaving model is. The location of the peak demand will in reality be variable as it relates to the specific locations where vehicles are weaving at a particular time. If a detector would continuously change its location to the peak demand, higher capacity values would be found. These capacities would relate to the accepted gaps as mentioned at the first bullet.

Performance is reasonable with a small average error and an acceptable maximum error for a macroscopic model. The distribution of absolute errors is skewed towards small errors and so the errors close to the maximum error are rare, as the standard deviation also indicates. From Table 7.3 can be learned that there are only nine errors larger than 20%. Six of these belong to layout B23 and B32. From the mean errors per layout we can conclude that the more lanes a layout has, the less accurate the model performs on average. This is logical as layouts with more lanes also have more movements that are not explicitly accounted for at the peak demand.

Table 7.3: Calibration errors (non absolute) for each layout and demand pattern.
Blanks are either not a weaving section (all traffic from one and to one link) or have the same demand pattern as a previous one as the $\psi_c$ range is a single value.

| Layout | Relative error [%] | | | | | | | | | | | | | | | | | | | | | | | | | Mean |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A21 | | 0,00 | 0,00 | 0,00 | | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | -1,63 | -1,88 | -2,48 | -7,14 | -4,60 | -4,16 | -4,17 | -4,83 | -5,13 | -7,53 | -5,15 | -7,31 | -5,25 | 0,00 | | -2,78 |
| A22 | | -8,04 | -3,93 | -13,38 | | -5,54 | -1,06 | 2,63 | -0,18 | -9,21 | -5,55 | | | | | -10,15 | -3,80 | 3,12 | -0,50 | -5,57 | | -16,29 | -3,90 | -5,72 | | -5,12 |
| A31 | | 0,00 | 0,00 | 0,00 | | -0,75 | 0,00 | 0,00 | 0,00 | 0,00 | -2,59 | 0,00 | 0,00 | 0,00 | 0,00 | 1,02 | 0,85 | 0,69 | 2,15 | 1,99 | -2,29 | -5,29 | -9,41 | -3,75 | | -0,79 |
| A32 | | -8,04 | -3,92 | -13,41 | | -5,51 | 1,20 | -0,25 | -12,15 | -24,03 | -1,63 | 1,67 | 1,78 | -1,87 | -6,50 | -2,68 | 4,49 | 7,05 | 9,15 | 7,54 | -15,21 | -9,94 | -0,80 | -9,27 | | -3,74 |
| A41 | | 0,00 | 0,00 | 0,00 | | -0,85 | 0,00 | 0,00 | 0,00 | 0,00 | -3,58 | 0,00 | 0,00 | 0,00 | 0,00 | -3,95 | -3,20 | -10,87 | -6,65 | -12,80 | -1,50 | -5,22 | -8,40 | -3,05 | | -2,73 |
| B12 | -2,31 | 0,00 | 0,55 | 9,21 | | -0,89 | 1,25 | 5,39 | 5,89 | 5,50 | -3,94 | 0,00 | 0,00 | 0,00 | -0,18 | -1,92 | 0,00 | 0,00 | 0,00 | 0,00 | | 0,00 | 0,00 | 0,00 | | 0,84 |
| B22 | -3,08 | 0,00 | 2,27 | 10,88 | | 1,07 | 10,02 | 12,88 | 15,57 | 17,06 | 11,45 | 11,45 | 11,45 | 11,45 | 11,45 | -0,46 | -3,73 | 0,00 | 0,00 | 0,00 | -1,81 | -3,76 | -6,01 | 0,00 | | 4,7 |
| B22' | -0,25 | 0,00 | 0,00 | 0,00 | -0,48 | -2,34 | -0,16 | 0,00 | -0,31 | -2,55 | 5,11 | | | | | 7,07 | 5,41 | 2,40 | 0,04 | -3,17 | 3,18 | -1,08 | -8,42 | -8,40 | -7,79 | -0,56 |
| B23 | -12,48 | -7,88 | 10,97 | 24,03 | | 13,92 | 17,39 | 20,55 | 22,44 | 23,41 | 13,10 | 14,91 | 15,15 | 14,04 | 12,14 | -5,32 | 4,00 | 0,41 | 0,00 | 0,00 | | -8,67 | -0,14 | -4,94 | | 7,59 |
| B32 | -2,65 | 0,00 | 2,84 | 11,42 | | 2,79 | 10,48 | 13,37 | 16,07 | 18,62 | 17,11 | 23,36 | 21,85 | 19,54 | 13,84 | 6,31 | 3,98 | -0,31 | 0,00 | -1,80 | 6,76 | 1,48 | -5,26 | -2,87 | | 7,69 |
| B32' | -0,24 | 0,00 | 0,00 | 0,00 | 0,00 | -2,88 | 0,00 | 0,00 | 0,00 | 0,05 | 6,23 | 6,93 | 8,69 | 4,57 | -3,64 | 7,95 | 8,22 | 7,49 | 4,94 | 2,78 | -3,65 | -6,23 | -11,05 | -9,84 | -8,73 | 0,46 |
| C22 | -9,05 | -8,51 | -8,14 | -2,50 | | 0,34 | 1,55 | 2,34 | 5,73 | 7,76 | 5,05 | 3,54 | 0,12 | -6,15 | -17,59 | 5,36 | 6,85 | 6,52 | 5,84 | 5,07 | -8,11 | -9,18 | -5,61 | 0,00 | | -0,82 |
| C22' | 0,21 | 0,00 | 0,00 | -0,21 | -1,05 | -13,86 | -7,96 | -3,70 | -4,93 | -6,70 | -2,89 | -3,57 | -4,36 | -8,46 | -12,03 | 1,76 | 1,99 | -2,09 | -1,13 | -6,05 | | -9,81 | -4,28 | 0,00 | | -3,87 |
| C32 | -9,45 | -8,48 | -8,08 | 0,09 | | -3,43 | 5,26 | 9,98 | 14,22 | 17,32 | 6,73 | 6,73 | 6,73 | 6,73 | 6,73 | 12,58 | 12,62 | 12,65 | 12,69 | 12,72 | -6,14 | -11,67 | -8,80 | -3,10 | | 3,68 |
| C32' | 0,41 | 0,00 | 0,00 | 0,00 | 0,00 | -5,53 | 1,08 | 0,46 | -7,74 | 0,62 | -5,97 | -6,44 | -5,63 | -7,13 | -7,71 | 4,20 | 5,08 | 3,21 | -4,83 | -8,42 | -2,96 | -0,11 | -3,05 | -2,26 | | -2,2 |
| D22 | | -8,11 | -8,07 | -7,77 | -1,89 | 2,57 | 3,61 | 5,75 | 6,45 | 7,59 | -12,67 | -1,40 | 5,22 | 6,94 | 7,48 | 7,12 | 7,14 | 6,32 | 2,95 | 2,60 | | -17,83 | -10,22 | -5,63 | -5,63 | -0,33 |
| D22' | | 0,00 | 0,00 | 8,18 | | -5,50 | 0,00 | 0,47 | 7,16 | 11,20 | -8,60 | -4,54 | 0,62 | 4,25 | 4,06 | -5,83 | -2,49 | -0,89 | -1,60 | -6,50 | 0,23 | 0,00 | 0,00 | 0,00 | -0,76 | -0,02 |
| D32 | | -8,19 | -1,42 | 13,46 | | 0,83 | 7,83 | 13,04 | 17,73 | 21,76 | 16,93 | | | | | 8,87 | 8,90 | 8,93 | 8,97 | 9,00 | -17,27 | -18,11 | -17,25 | -12,80 | -11,10 | 2,64 |
| D32' | | -4,40 | -6,69 | 0,58 | | -7,03 | -4,09 | 1,25 | 4,78 | 8,69 | -8,40 | -5,67 | -2,04 | -0,77 | -1,29 | 12,57 | 8,47 | 2,01 | 0,00 | 0,00 | | 11,82 | 4,98 | 0,00 | | 0,7 |
| E22 | | -8,06 | -8,08 | -3,68 | | -14,90 | -6,20 | -3,75 | 2,21 | 10,81 | -6,13 | | | | | -6,40 | -2,94 | -3,33 | -2,80 | -2,15 | | -23,37 | -10,90 | -3,55 | | -5,48 |
| E32 | | -8,27 | -7,97 | -1,65 | | -14,22 | -6,90 | 3,12 | 9,44 | 16,60 | -7,21 | -4,36 | -2,45 | 0,43 | -0,93 | -4,87 | -2,67 | -5,07 | 0,88 | 0,18 | -24,03 | -15,79 | -12,82 | -7,04 | | -4,35 |

## 7.4 Conclusions

In this chapter a new weaving model has been presented based on a new theory that focuses on lane demand and leaves the theory of turbulence. A utility based demand distribution results in lane demands at the start of the critical section. A peak demand is calculated by also including a fraction of certain weaving movements. This results in a very local and location variable peak demand that should not exceed peak capacity. The resulting performance is reasonable.

The proposed model has some additional features. Merge and diverge sections can be modelled by excluding a link. To deal with short weaving section lengths, the logit scale factor ($\lambda$) can be increased. This makes drivers more sensitive to the disutility elements. A calibration for this has not been performed. The model can also cope with lane-specific elements. For instance dedicated lanes for heavy vehicles can be taken into account by introducing classes and defining different utilities per class. Road marks can be accounted for by setting the disutility of some movements to minus infinity. In short, the model is very flexible and is easily adapted.

This chapter and the previous chapter have explained the new node model while chapter 5 has explained the new link model. The next chapter will evaluate the entire new model.

# 8. Evaluation

To evaluate all proposed changes from the previous three chapters, the various sub models will be evaluated separately in terms of accuracy and realism. Separate evaluations make it easier to distinguish phenomena that are related to different models. The link model (CBQ) will be evaluated in two ways. First, in section 8.1 a hypothetical example will be given with a random pattern of link in- and outflow. It is checked whether cumulative flows relate according to shockwave theory. In the following, the various node models will be evaluated, first qualitatively and than quantitatively by a comparison with VISSIM. Networks with a single node and a few connecting links will be evaluated by traffic state patterns. Section 8.3 will evaluate the significance of the changes to EVAQ. The old and the new version of EVAQ will be related to VISSIM results. In section 8.4, the entire model performance in terms of CPU and memory will be evaluated. Section 8.5 will elaborate on the applicability of the new model after which conclusions are presented.

## 8.1 Link model evaluation

To evaluate the link model and in particular CBQ, a link with the following properties will be tested.

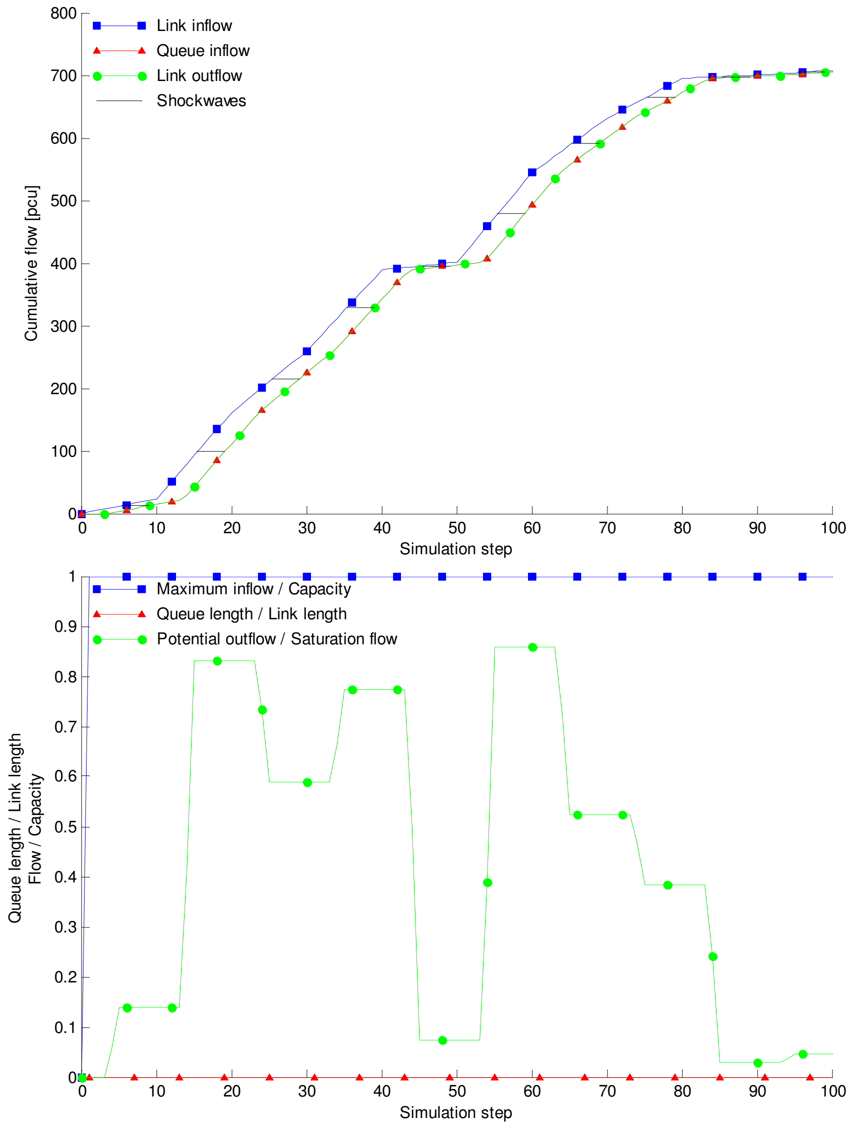
| | |
|---|---|
| • Length: | 2 km |
| • Capacity: | 4000 pcu/h |
| • Lanes: | 2 |
| • Maximum speed: | 100 km/h |
| • Saturation flow: | 3000 pcu/h |

With a time step of 20 seconds and a jam density of 150 pcu/km/lane, it follows that there are 24 cells, the last of which is 40% of the usual length. Congested shockwaves thus take 23.4 time steps to transverse the link. Free flow shockwaves take 3.6 time steps, which is the free flow travel time over the link. Three scenarios will be evaluated. The first will cover free flow, the second will cover congestion and the last scenario will cover a combination.

### 8.1.1. Scenario 1: Free flow

This scenario will show that free flow shock waves are modelled correctly. To keep the link fully free flow, outflow will be equal to potential outflow. Inflow will be a random fraction in the range [0 ... 0.7] of maximum inflow. The fraction is fixed for ten time steps. This allows recognition of the waves. The maximum of 0,7 of the range is useful for scenario 3 where the same randomly generated pattern will be used. It balances capacity for inflow and saturation flow for outflow, as capacity is higher than saturation flow. In Figure 8.1 it can be seen that queue inflow coincides with link outflow. This is consistent with the fact that there is no queue. The shockwaves are plotted from

Figure 8.1: CBQ scenario 1



several queue inflow values towards link inflow (and outflow, but these have no length). The time that the shockwaves span is calculated from free flow shockwave speed and free flow distance, it is therefore not surprising that indeed the shockwaves take 3,6 periods if there is no queue. A validation can however be found in the slope of the shockwaves. The value for link inflow is namely taken at time *t*–3,6 where the value should match cumulative queue inflow at time *t*. As the free flow shockwaves are indeed horizontal, this is modelled correctly. The second graph in Figure 8.1 shows that maximum inflow is always equal to capacity. Potential outflow shows the random inflow pattern delayed by 3.6 periods. Queue length is always zero.

### 8.1.2. Scenario 2: Congestion

This scenario will show that congested shockwaves are modelled correctly. To create a congested link, inflow will be equal to maximum inflow and outflow is a random fraction of potential outflow. Also here each fraction is maintained for 10 steps to visualize the shockwaves. In Figure 8.2 it can be seen that for a fully congested link, link inflow and queue inflow do not perfectly coincide. This is because queue inflow

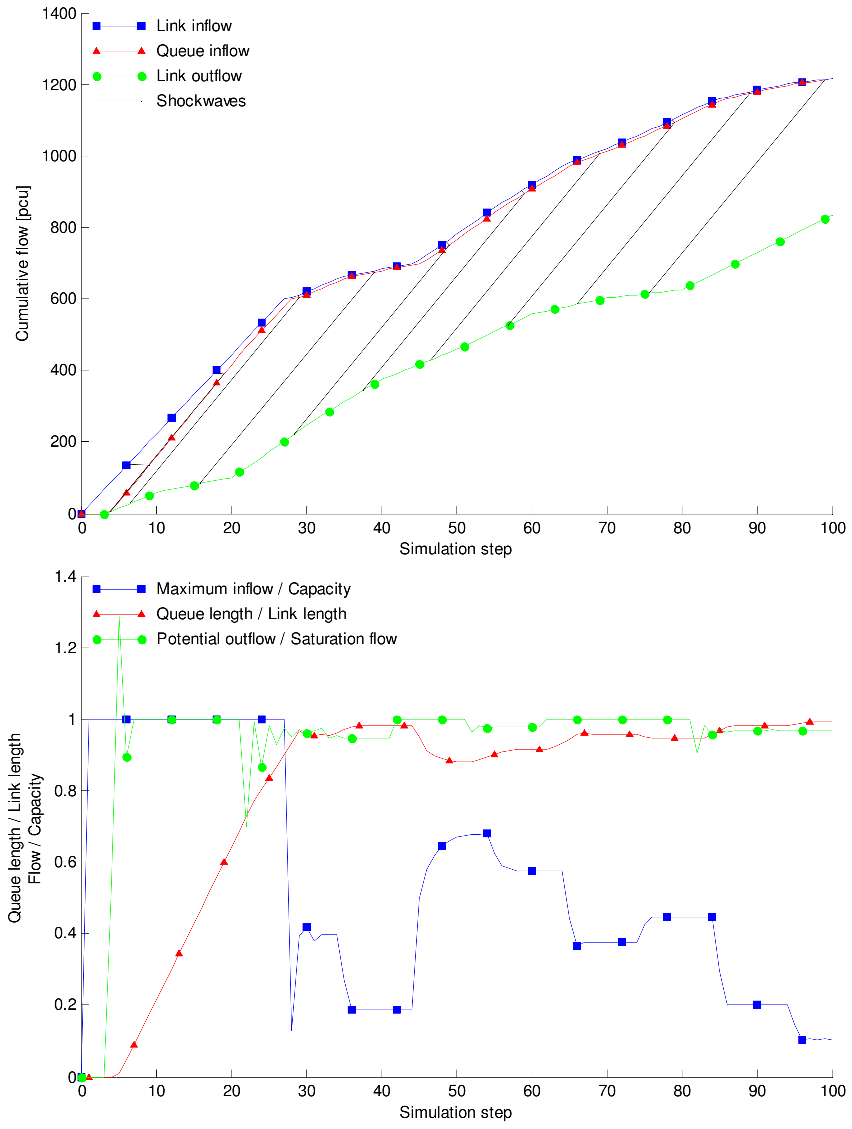


Figure 8.2: CBQ scenario 2

can only be determined from vehicles on the link. For a fully congested link, vehicles that flow into the queue in a time step actually come from other links. This creates small inconsistencies. Just as for free flow shockwaves, the time that congested shockwaves span is calculated from the queue length and the congested shockwave speed. Congested shockwaves on fully congested links thus indeed span about 23,4 periods. Again the slope of the shockwaves functions as a validation tool. All slopes are similar. Moreover, at this slope the pattern of link outflow is repeated at link inflow. In the second graph of Figure 8.2 it can be seen that potential outflow can be larger than saturation flow. This is however only true when congestion has just started (queue is short). Free flow vehicles than still play a role. It can also be seen that even with long queues, potential outflow may be smaller than saturation flow. The cause of this is queuing dynamics and resulting densities together with speeds. Whenever the outflow pattern changes value, some oscillation might be visible for the potential outflow. This is because a new equilibrium flow needs to be found where the fraction of potential outflow generates the same potential outflow for the next time step. The effect quickly dampens out and is barely visible in the cumulative flow. Maximum inflow shows the outflow pattern with a delay, representing the congested shockwave accurately. Also inflow will find an equilibrium value that is both dependant on itself and outflow (with a delay). The queue quickly grows to almost the link length. As queue inflow is derived from free flow vehicles, it cannot reach full link length.

### 8.1.3. Scenario 3: Combined

This scenario is a combination of the previous two scenarios. Both inflow and outflow are based on a pattern. The two patters are different from one another, but equal as in the previous scenarios. From Figure 8.3 it can be seen that inflow is the same as in scenario 1. This is logical, as the queue never spans the entire link. Queue inflow is shifted to the left and slightly morphed. This is consistent with the short and variable queue length from the second graph in Figure 8.3. Link outflow is shaped like a combination of outflow from the previous scenarios. For short queue lengths the shape is more similar to scenario 1. For longer queue lengths the shape is more similar to scenario 2. This is perfectly logical, as scenario 1 has no queue while scenario 2 does.

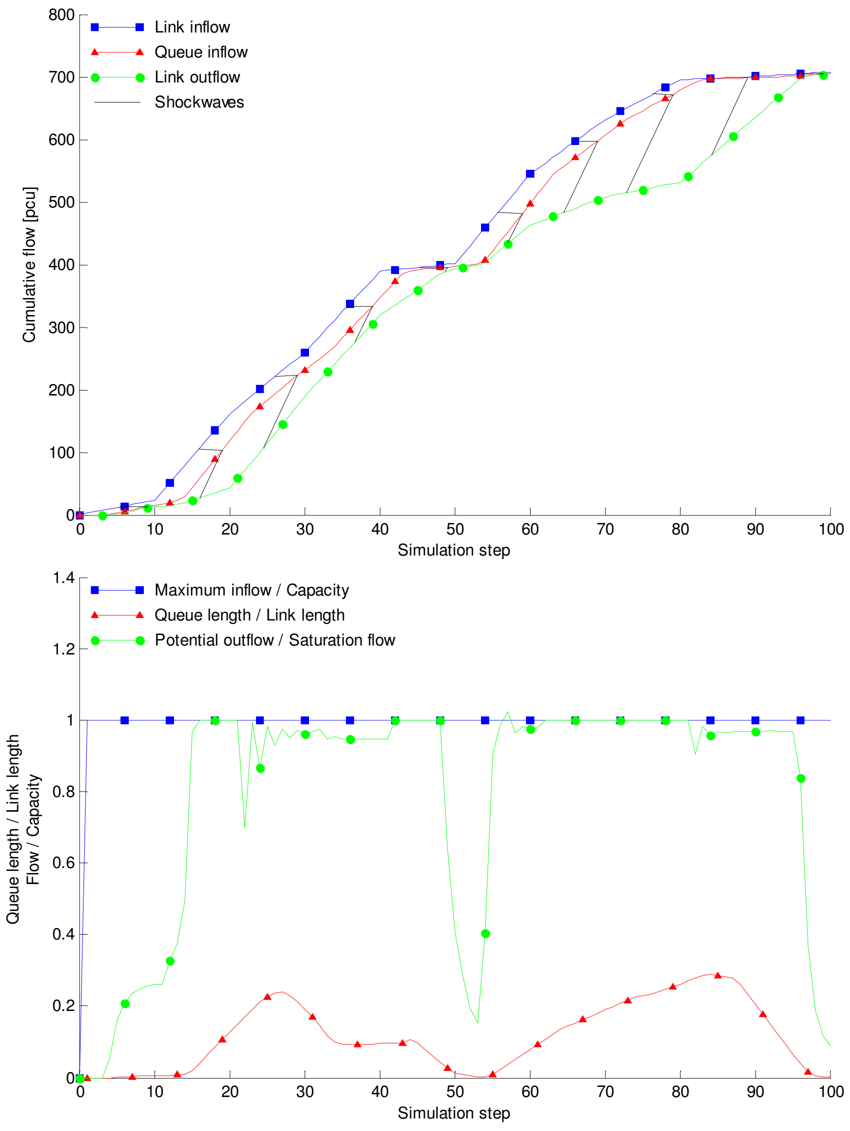


Figure 8.3: CBQ scenario 3

### 8.1.4. Sensitivity analysis

A sensitivity analysis was performed to analyse outcome sensitivity for each input parameter. Capacity, saturation flow, maximum speed and jam density are either increased or lowered. All but saturation flow determine the shape of the fundamental diagram. The 3rd scenario is used for a comparison. The eight resulting plots are shown in Figure 8.4. Outcome appears to be relatively insensitive to maximum speed and jam density. As maximum speed defines density at capacity, it can be concluded that outcome is rather insensitive to the density dimension of the fundamental diagram. The model is sensitive to capacity and saturation flow. Total flow appears linear with capacity as a capacity of 5000 pcu generates about 5/3rd of flow with a capacity of 3000 pcu. Saturation flow is a factor determining outflow. A linear relation can however not be found as outflow also depends on inflow. Still it can be seen that higher saturation flow results in a significant increase of outflow for congested circumstances.

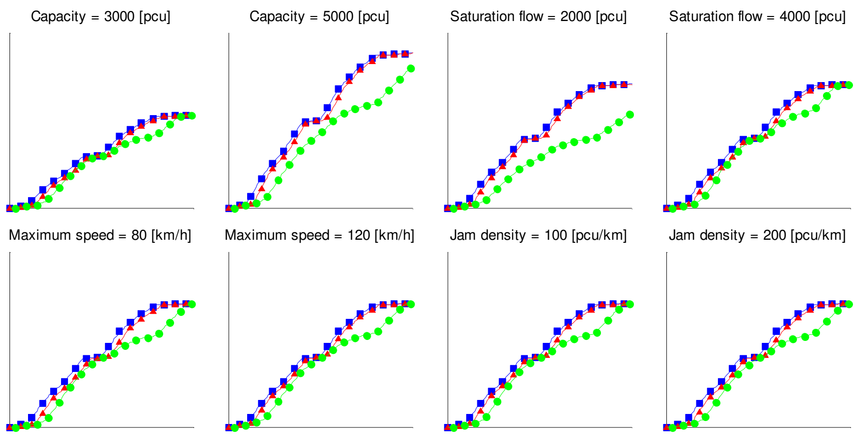


Figure 8.4: CBQ sensitivity analysis

### 8.1.5. Conclusions

CBQ is able to model both congested and free flow shockwaves in a consistent manner. The effect of using the cell states for potential outflows is valid as potential outflow is often smaller than saturation flow during congestion. Saturation flow thus functions as an upper limit of congested outflow. Outcome is sensitive to capacity and saturation flow.

## 8.2 Node model evaluation

### 8.2.1. Controlled intersections

To evaluate the mechanism for controlled intersections a regular intersection with four roads will be used as in Figure 8.5. The roads are one kilometre long and have a maximum speed of 50 km/h. The north and south road have one lane while the east and west road have two lanes. Capacity equals 2000 pcu/h/lane and saturation flow equals 1800 pcu/h/lane. From the north and the south 1000 pcu will leave evenly spread throughout an hour. For the east and west this is 2000 pcu.

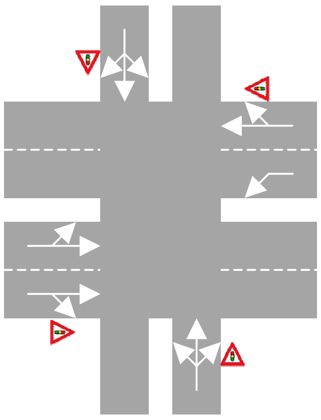


Figure 8.5: Controlled intersection

Outflow in the four directions is unlimited, except for the west link where there is a random limit of about 1/3rd of average flow. At some point this will create spillback for the central intersection. Split fractions at the intersection are random for each time step and evenly distributed (on average) over all links. The turn lane layout can be seen in Figure 8.5. Outcome of the model over an hour was exported into a movie from which qualitative observations can be made. The movie displays the traffic states of the cells, not the traffic itself. The following observations were made:

Shockwaves

1. Free flow traffic states move downstream with relatively high and constant speed.
2. Congested traffic states move upstream with a relatively slow and constant speed.

Traffic state patterns

3. The east and west link have very similar traffic states moving upstream. This is logical as these roads have most traffic and are thus usually both part of the critical conflict group at the intersection. Their reduction factors are thus equal.
4. The north and south link have more or less similar traffic states moving upstream. Usually all four links are part of the critical conflict group, but sometimes either the north or the south link is not. Reduction factors are than unequal.
5. The fact that the links show similar patterns despite random split fractions shows that the intersection control is optimal and distributes disturbance evenly over the critical conflict group. The total flow over an hour is practically equal for the north link and south link. The same holds for the east link and west link.

Congestion and spillback

6. The intersection limits flow from all links as congestion start at all links as soon as traffic reaches the intersection. After this congestion continues to increase.

7. Congestion builds up from the west link towards the central intersection. Congested traffic states move upstream.
8. As soon as the congestion on the west link reaches the central intersection, congested traffic states on all links are very similar. This is due to the fact that the limit of spillback is dominant. Shockwaves from the west link travel over the intersection and on to all upstream links.

Controlled intersections behave as expected. Both free flow and congested traffic waves move according to shockwave theory. The amount of similarity between links in congested traffic states coincides with the change of being part of the critical conflict group. Traffic states move over the intersection as soon as spillback starts.

### 8.2.2. Uncontrolled and priority intersections

An equal approach as for controlled intersections was performed for uncontrolled intersections. Two movies were generated, one where the east and west roads have priority and one where there is no special priority rule, see Figure 8.6.

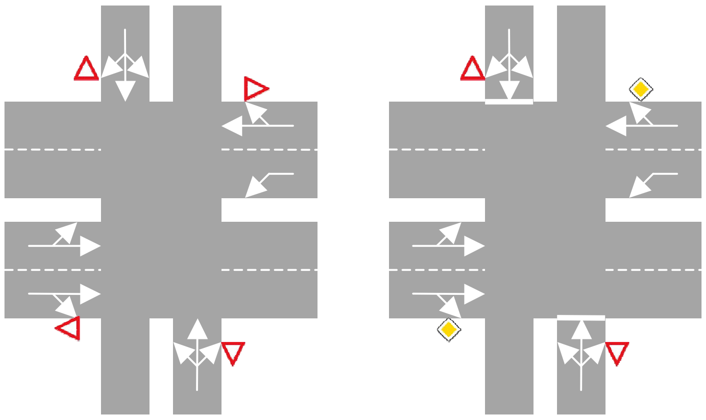


Figure 8.6: Uncontrolled and priority intersections

The following observations were made:

1. The shockwaves and congestion and spillback observations from controlled intersections also hold for uncontrolled and priority intersections.
2. All links show dissimilar traffic states. Limits on capacity are thus link specific.

Without priority for the east and west roads:

3. The north and south link have lower density, higher speeds and shorter queues. This is strongly related to the size of the flows and the turn lane layout and is thus not a general property of uncontrolled intersections.
4. As soon as spillback occurs, the south link has more capacity than the north link and the west link has more capacity than the east link. As flow towards the west reduces, so does its impact on total flow. The other links, for which the south and west link have more priority than their counterparts, thus

increase in their influence. In other words, the north and east link are unable to fully utilize their priority as their priority is largely towards the congested west link.

With priority for the east and west roads:

5. A similar phenomenon to observation 4 exists for the priority intersection. Since priorities are different, so is the size of the effect for the north/south and east/west combinations.
6. The east and west links have low densities and shorter queues while the north and south link have very high densities and longer queues. This is completely in line with the priority rule and the resulting capacities from the links.

Uncontrolled and priority intersection behave as expected. The amount of congestion coincides with the amount of priority. The impact of spillback and priority towards the link that produces spillback is logical. Links that rely on this priority for their outflow capacity show more degeneration.

### 8.2.3. Roundabouts

The same framework is again used for the evaluation of roundabouts. A 2-lane roundabout and a turbo roundabout were evaluated. The turbo roundabout was designed for more flow from the east and the west. This is displayed in Figure 8.7.

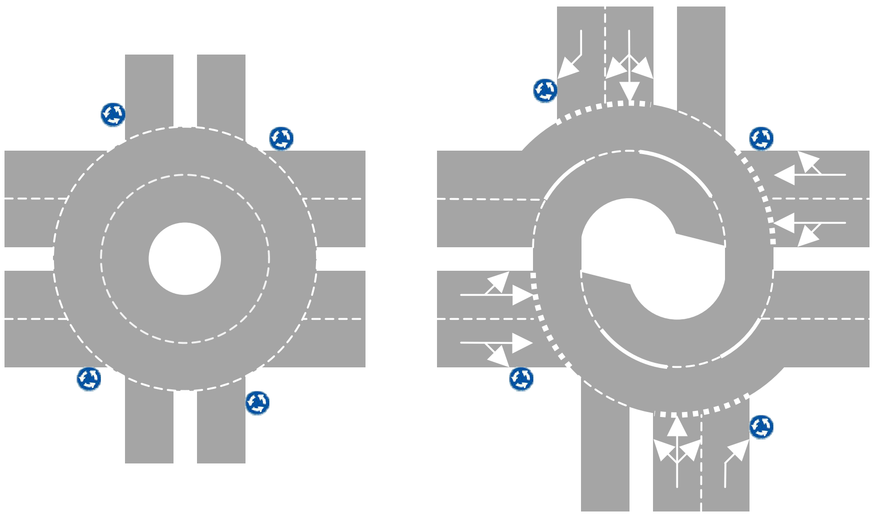


Figure 8.7: 2-lane and turbo roundabout

1. The shockwaves and congestion and spillback observations from controlled intersections also hold for roundabouts.
2. All links show dissimilar traffic states. Limits on capacity are thus link specific, this is similar as for uncontrolled and priority intersections.

2-lane roundabout:

3. Traffic states on all links, although different, are all within a small range. This is dependant on the demand pattern which in this case more or less coincides with the capacity pattern.

Turbo roundabout:

4. Densities on the east and west link are lower even though the demand is twice as high. Capacity is thus more than twice as high. This is in line with the design of the turbo roundabout.

Roundabouts show expected traffic states. The effect of a turbo roundabout with respect to a 2-lane roundabout follows the design as traffic from the west and east has less degeneration.

### 8.2.4. Weaving sections

A calibration for weaving sections has already been discussed in section 7.3. The interaction with CBQ however has not been covered yet. For weaving sections a new framework is used. Figure 8.8 shows the used layouts. Demand is shown and split fractions are again random but follow the displayed outflow demand on average. Roads are again one kilometre long and have a maximum speed of 100 km/h. The capacity is 2000 pcu/h/lane and saturation flow is 1500 pcu/h/lane.

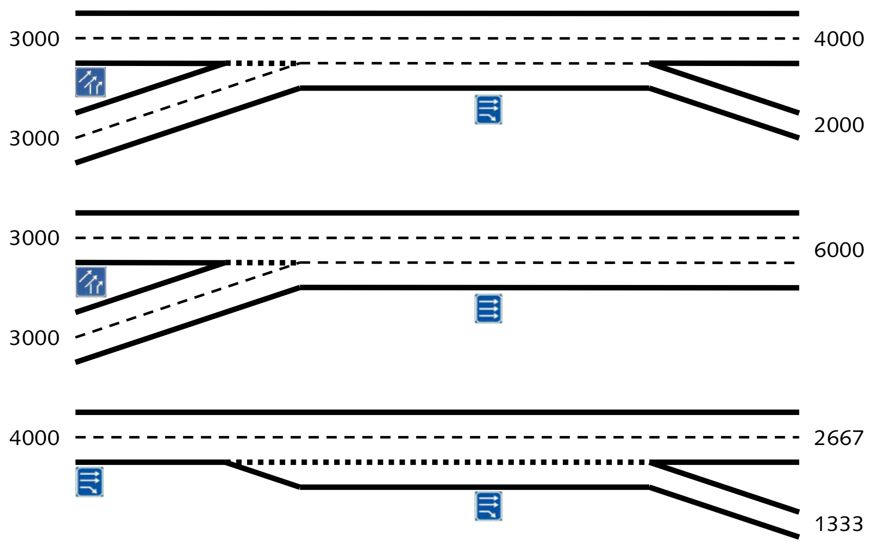


Figure 8.8: Weaving sections

1. The shockwaves and congestion and spillback observations from controlled intersections also hold for weaving sections. The link with spillback is however either the off-ramp or the highway itself for the merge section. Despite the under saturated condition of the links, all layouts have congestion at the weaving section before there is spillback. Even the off-ramp, which is an additional lane, creates congestion. The amount of congestion is however less.
2. For both the on-ramp and the weaving section, both entering links are affected equally. This is consistent as the links have equal demand and capacity.

All layouts show congestion as predicted by the weaving model based on lane choice. Impact on the links is equal as assumed in the weaving model.

### 8.2.5. A comparison with VISSIM

Each of the nodes as in the previous sections is modelled using VISSIM. VISSIM is a microscopic model that takes many details into account. It forms a good benchmark for model outcome. External spillback is excluded as the node capacity and the resulting queue length are of interest. Default settings and intersection definitions in terms of links, connectors and priorities were defined as indicated by the user manual (VISSIM 5.10 User Manual). The simulation period is one hour. Each node type is modelled in 30 runs to even out stochastic dispersion. Table 8.1 shows the average resulting link outflow capacities of both EVAQ and VISSIM. Also the errors are given.

Table 8.1: Link outflow capacities from EVAQ and VISSIM [pcu/h (including link travel time)]

| Link | Con-trolled | Uncon-trolled | Priority | 2-lane round. | Turbo round. | Weave | On-ramp | Off-ramp |
|---|---|---|---|---|---|---|---|---|
| **EVAQ** | | | | | | | | |
| 1 | 714 | 971 | 376 | 460 | 483 | 2192 | 1981 | 2924 |
| 2 | 1016 | 1370 | 1862 | 935 | 1225 | 2192 | 1981 | |
| 3 | 673 | 975 | 377 | 458 | 482 | | | |
| 4 | 1054 | 1517 | 1835 | 931 | 1228 | | | |
| All | 3456 | 4834 | 4450 | 2784 | 3418 | 4384 | 3962 | 2924 |
| **VISSIM** | | | | | | | | |
| 1 | 496 | 851 | 480 | 522 | 394 | 2619 | 2926 | 3856 |
| 2 | 999 | 1091 | 1967 | 1094 | 1094 | 2208 | 1316 | |
| 3 | 507 | 850 | 512 | 505 | 387 | | | |
| 4 | 954 | 1496 | 1969 | 994 | 1280 | | | |
| All | 2957 | 4287 | 4929 | 3115 | 3155 | 4826 | 4242 | 3856 |
| **Errors** | | | | | | | | |
| 1 | 44% | 14% | -22% | -12% | 23% | -16% | -32% | -24% |
| 2 | 2% | 26% | -5% | -14% | 12% | -1% | 51% | |
| 3 | 33% | 15% | -26% | -9% | 24% | | | |
| 4 | 10% | 1% | -7% | -6% | -4% | | | |
| All | 17% | 13% | -10% | -11% | 8% | -9% | -7% | -24% |

Generally the total node capacity is modelled with reasonable precision. The off-ramp is modelled worst with an error of 24%, which is in line with the error range of the weaving model. Larger errors can be found for the specific link outflows. Large errors are found for the on-ramp. This largely follows from the assumption that any reduction in the weaving model applies equally on both entering links. In VISSIM the main highway is affected less than the on-ramp. This follows from priority-like behaviour at the merge taper. It should be noted that such merging behaviour is difficult for microscopic models such as VISSIM and may not be a very good benchmark. Large errors are also found for the controlled intersection at links 1 and 3. These links have 1 lane that may cause larger reductions of saturation flow than the used representative value of 1300 pcu/h. This can be assumed from the fact that shared turn lanes with two directions have larger reductions than for only a single left or right turn. Turn lanes with three directions probably have larger reductions. Statements about smaller errors are difficult to make as VISSIM is also (just) a model. In other words, different models will always have different results and more certainty can only be acquired using actual data from reality.

Appendix C holds plots that display the queue length through time for all links of the various nodes. Generally the queue lengths are correct. Slight differences between EVAQ and VISSIM are visible at the slope of the plots (queue length growth). These differences can be explained by the errors from Table 8.1. An over estimation of capacity entails a more flat slope (smaller queue) while an under estimation entails a more steep slope (larger queue). Besides the differences in slope, a few plots appear to have rather different patterns. For these plots can be seen that the existence of a queue in itself is uncertain. Either EVAQ or VISSIM hardly shows any queue while the other model will show a queue that grows slowly on average. The growth of these queues is in the order of 100-200 pcu/h. Such a difference can easily follow from slight capacity differences. A last observation is that especially for the off-ramp scenario, queue length never fully reaches link length. This is correct as the link model calculates queue inflow from the current free flow vehicles on the link. In other words, for any equilibrium of flow through a queue, a certain free flow section is required that can provide this flow each time step. The length of this free flow section is dependent on the free flow speed, the equilibrium flow and the time step size. As flow through the queue (per lane) and maximum speed are relatively high for the off-ramp scenario, the free flow section required to represent a fully congested link is somewhat long. Spillback will however be modelled correctly as this free flow section represents the maximum queue inflow. It can thus be concluded that given correct constraints by the node model, CBQ is very well able to model the queue correctly.

## 8.3 New versus old EVAQ outcome

In the previous sections it was shown that the new link and node model show correct behaviour but at the cost of additional calculation time. In order to accept additional calculation time, the behaviour should deviate significantly from the previous version of EVAQ. To analyse the changes all networks from section 8.2 are run through various model versions:

- Old EVAQ
- Old EVAQ with new link model
- Old EVAQ with new node model
- New EVAQ (performed in section 8.2)
- VISSIM (performed in section 8.2)

### 8.3.1. Queue lengths comparison

Changes are analysed by the queue length, as it is the result of both the link and the node model. In this paragraph a visual comparison is given for one link and a quantitative comparison is given for all links. The visual comparison can be seen in Figure 8.9. It shows the queue length through time of 30 runs from the 4th link of the uncontrolled intersection. For this link the capacity error of the node model is 1% allowing a good comparison also of the link model with the VISSIM result. The black line is the average queue length of the 30 runs. The smooth flattening of the average queue length follows from more and more queue lengths that have reached link length. It is not an expected shape of a single queue growth.

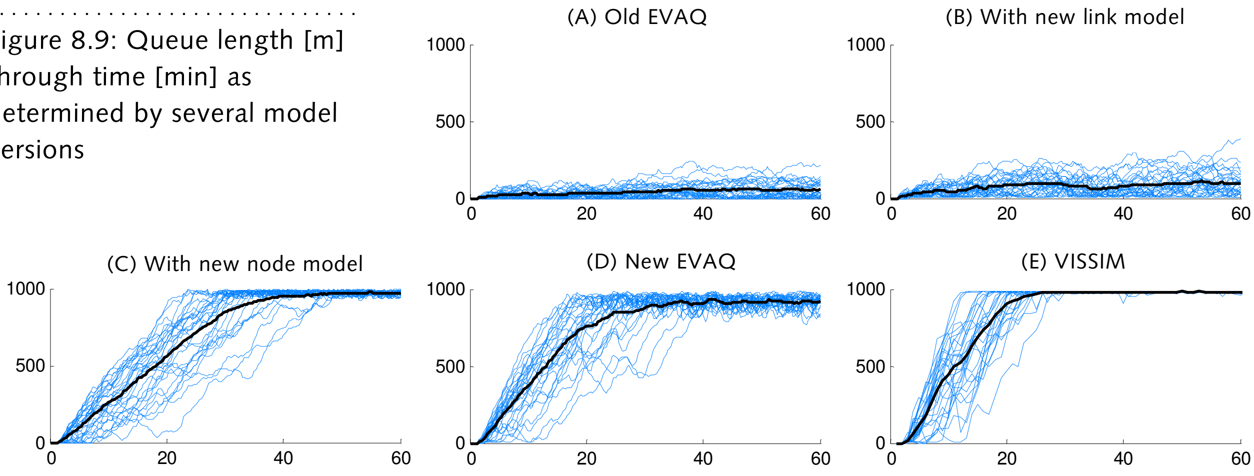


Figure 8.9: Queue length [m] through time [min] as determined by several model versions

From Figure 8.9 (A) and (B) it follows that CBQ produces slightly longer queues which follows from lower densities that are related to congested flow via a fundamental diagram. These densities are always lower or equal to jam density, which the old model uses. Longer queues are also found between (C) and (D). From Figure 8.9 (A) and (C) it follows that the new node model significantly reduces outflow, as queues are much longer with equal density. Finally, from Figure 8.9 (B), (C), (D) and (E) we can see that both the new node model and the new link model are needed to produce results similar to VISSIM. It can thus be concluded that both sub models significantly contribute to the results.

To compare the queue length quantitatively, only the average queue length of the 30 runs is used. The queue length grows up to the length of the link for many links and for various model versions. Assessing the queue length after the modelled hour is thus useless. Instead, for each link and for each model version the maximum five-minute queue growth is taken. The five minutes may be between any two time steps and not only at 0, 5, 10, 15, etc. minutes. Table 8.2 lists the queue growth for all models. Also the absolute relative and absolute errors are given.

| Model/version: | | Old EVAQ | | | | Old + link model | | | | Old + node model | | | | New EVAQ | | | | VISSIM | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Link number: | | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| **Maximum 5 minute queue length growth [m]** | | | | | | | | | | | | | | | | | | | | | |
| Node type | controlled | 24 | 24 | 24 | 24 | 39 | 39 | 39 | 39 | 204 | 335 | 209 | 338 | 270 | 393 | 286 | 395 | 445 | 533 | 446 | 480 |
| | uncontrolled | 25 | 25 | 25 | 25 | 42 | 42 | 42 | 42 | 32 | 192 | 19 | 167 | 60 | 281 | 49 | 236 | 220 | 430 | 291 | 347 |
| | priority | 23 | 23 | 23 | 23 | 40 | 40 | 40 | 40 | 400 | 65 | 393 | 73 | 464 | 131 | 452 | 134 | 464 | 5 | 419 | 5 |
| | roundabout | 24 | 24 | 24 | 24 | 40 | 40 | 40 | 40 | 349 | 348 | 351 | 338 | 436 | 433 | 438 | 433 | 465 | 453 | 476 | 491 |
| | turbo round. | 23 | 23 | 23 | 23 | 45 | 45 | 45 | 45 | 335 | 262 | 348 | 249 | 426 | 328 | 421 | 344 | 531 | 483 | 526 | 462 |
| | weave | 925 | 925 | | | 878 | 878 | | | 240 | 240 | | | 524 | 524 | | | 337 | 778 | | |
| | on ramp | 925 | 925 | | | 878 | 878 | | | 309 | 309 | | | 570 | 570 | | | 80 | 418 | | |
| | off ramp | 1005 | | | | 1005 | | | | 100 | | | | 505 | | | | 692 | | | |
| Average/VISSIM: | | 207 / 50% | | | | 214 / 52% | | | | 248 / 60% | | | | 364 / 89% | | | | 411 | | | |
| **Absolute error with VISSIM [m]** | | | | | | | | | | | | | | | | | | | | | |
| Node type | controlled | 421 | 509 | 422 | 456 | 405 | 494 | 407 | 441 | 240 | 198 | 237 | 142 | 175 | 140 | 160 | 85 | | | | |
| | uncontrolled | 195 | 405 | 266 | 322 | 178 | 389 | 249 | 305 | 187 | 238 | 272 | 180 | 160 | 149 | 242 | 111 | | | | |
| | priority | 441 | 18 | 396 | 19 | 424 | 35 | 379 | 36 | 65 | 59 | 26 | 69 | 0 | 126 | 33 | 130 | | | | |
| | roundabout | 441 | 429 | 452 | 467 | 425 | 414 | 436 | 451 | 116 | 105 | 125 | 153 | 29 | 21 | 38 | 59 | Absolute error: $\lvert EVAQ - VISSIM \rvert$ | | | |
| | turbo round. | 509 | 460 | 504 | 439 | 486 | 437 | 481 | 416 | 196 | 220 | 178 | 213 | 106 | 154 | 106 | 118 | | | | |
| | weave | 588 | 147 | | | 541 | 100 | | | 97 | 538 | | | 187 | 254 | | | | | | |
| | on ramp | 845 | 507 | | | 798 | 460 | | | 228 | 109 | | | 489 | 151 | | | | | | |
| | off ramp | 313 | | | | 313 | | | | 591 | | | | 187 | | | | | | | |
| Average: | | 399 | | | | 380 | | | | 191 | | | | 136 | | | | | | | |
| **Absolute relative error with VISSIM [-]** | | | | | | | | | | | | | | | | | | | | | |
| Node type | controlled | 0,9 | 1,0 | 0,9 | 1,0 | 0,9 | 0,9 | 0,9 | 0,9 | 0,5 | 0,4 | 0,5 | 0,3 | 0,4 | 0,3 | 0,4 | 0,2 | | | | |
| | uncontrolled | 0,9 | 0,9 | 0,9 | 0,9 | 0,8 | 0,9 | 0,9 | 0,9 | 0,9 | 0,6 | 0,9 | 0,5 | 0,7 | 0,3 | 0,8 | 0,3 | | | | |
| | priority | 0,9 | 3,3 | 0,9 | 4,0 | 0,9 | 6,4 | 0,9 | 7,7 | 0,1 | 11,0 | 0,1 | 14,9 | 0,0 | 23,3 | 0,1 | 28,0 | Absolute relative error: $\frac{\lvert EVAQ - VISSIM \rvert}{VISSIM}$ | | | |
| | roundabout | 0,9 | 0,9 | 0,9 | 1,0 | 0,9 | 0,9 | 0,9 | 0,9 | 0,2 | 0,2 | 0,3 | 0,3 | 0,1 | 0,0 | 0,1 | 0,1 | | | | |
| | turbo round. | 1,0 | 1,0 | 1,0 | 1,0 | 0,9 | 0,9 | 0,9 | 0,9 | 0,4 | 0,5 | 0,3 | 0,5 | 0,2 | 0,3 | 0,2 | 0,3 | | | | |
| | weave | 1,7 | 0,2 | | | 1,6 | 0,1 | | | 0,3 | 0,7 | | | 0,6 | 0,3 | | | | | | |
| | on ramp | 10,5 | 1,2 | | | 9,9 | 1,1 | | | 2,8 | 0,3 | | | 6,1 | 0,4 | | | | | | |
| | off ramp | 0,5 | | | | 0,5 | | | | 0,9 | | | | 0,3 | | | | | | | |
| Average: | | 1,54 | | | | 1,74 | | | | 1,53 | | | | 2,55 | | | | | | | |

Table 8.2: Queue length growth of the old and new EVAQ and VISSIM

A general tendency is visible throughout the table. The link model alone forms only a slight improvement towards the VISSIM results. The node model alone performs much better but still does not resemble the VISSIM results. The new EVAQ model is a significant improvement relative to the other EVAQ versions. It should be noted that the average absolute relative error is worse, but dominated by only two links of the priority node that do not actually show large absolute errors. In short it can be concluded that the old EVAQ model has queue growth that is on average about 50% of the VISSIM queue growth. The new EVAQ model is a significant improvement with queue growth that is on average about 89% of the VISSIM queue growth.

### 8.3.2. Macroscopic Fundamental Diagram comparison

On link level the new model performs significantly different from the old model. Interesting also is to look at the network wide MFD again as in section 3.3. The same MFD's, but generated by the new model, are given in Figure 8.10 below the MFD's of the old model.

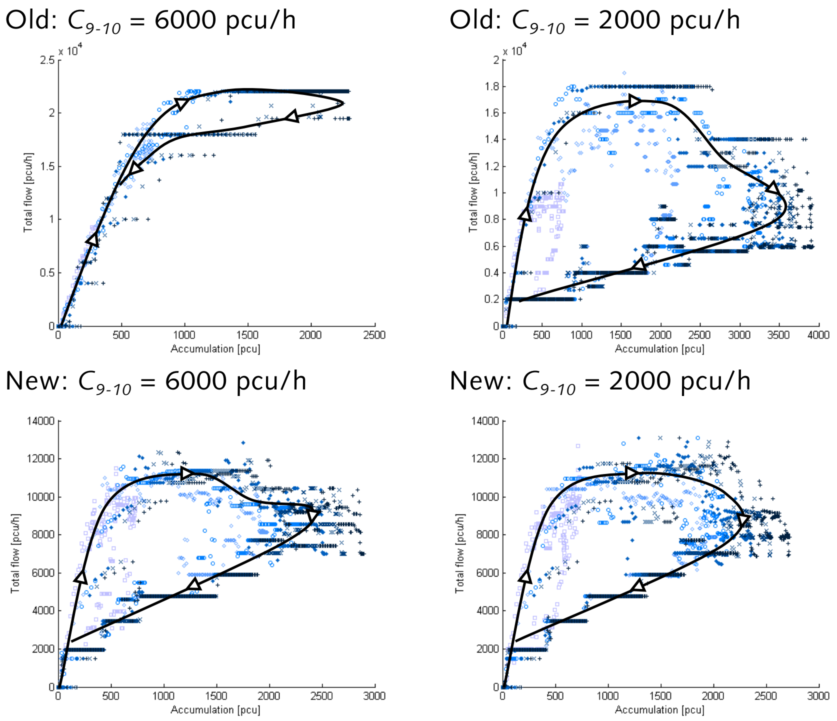


Figure 8.10: Old versus new macroscopic fundamental diagrams

In the new situation there is a much lower peak flow throughout the network. This follows from node constraints that are more limiting than link inflow. Also the situation where the exit link has a capacity of 6000 pcu/h shows significant NPD while this is not true for the old situation. Despite different exit link capacities, both new MFD's look very alike. Again this follows from node constraints that are more limiting but also the same. The new model still displays ranges where the accumulation changes but the flow remains perfectly equal. In the network filling phase this is however much less. The remaining horizontal parts in the filling phase can be explained as being an equilibrium state of outflow and route choice. The network depleting phase still shows large horizontal ranges. Just as with the old model this follows from links that go from being queued to being empty. The new MFD's still follow a cycle that is dissimilar from the MFD derived by Daganzo & Geroliminis (2008). Qian (2009) however explains that this coincides with the used modelling framework where density and flow are taken from different locations.

## 8.4 New versus old EVAQ performance

Both the new link model and the new node model have additional steps in relation with the old model. The cell based representation of queuing and the constraints at the node are the main additional steps. Logically it follows that additional calculation time is needed. During the implementation of the new model into the old model, evaluations were done to optimise the speed of the code. Some code from modules that were not functionally changed appeared highly inefficient. The most significant involved the repetition of a large matrix in order to have equal size in the expanded dimension to another matrix. Instead of actually expanding the matrix, smart indexing is much more efficient and has been implemented. This code is part of the split fraction generation. Both old and new code has been optimised in order to prevent such needles inefficiencies. Because of this the new model might not take as long as one would expect. Table 8.3 shows calculation time for several types of evacuations for 1000 time steps on a Dual Pentium 1,79 Ghz with 1,99 Gb of memory running on Windows XP. The network has 145 links and 61 nodes, 14 of which are modelled to have conflicts in the new node model. Other nodes are origins, destinations or merely a manner to connect an origin to a road with a single connector while there are many small intersections in reality. The voluntary evacuation has one class while recommended and mandatory evacuations have 23 classes. It can be seen that for the old model the split fractions are most significant in terms of calculation time, but the matrix expansion is only needed for recommended evacuations. In the new model this takes much less time but the new node and link models also have significant calculation time. The increase in calculation time is 123%, 36% and 742% for voluntary, recommended and mandatory evacuations respectively. Recommended evacuations take the longest. For these the increase in calculation time is very reasonable. Memory use has also increased mainly due to additional time steps needed in memory for CBQ.

Table 8.3: Performance for 1000 steps

| | | | Module time [s] | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Model** | **Evacuation** | **Memory [Mb]** | **EVAQ** | **Route set** | **Split fractions** | **Node** | **LCM** | **CBQ** |
| | Voluntary | 26 | 24 | 4 | 2 | | | |
| Old | Recommended | 174 | 87 | 4 | 70 | | | |
| | Mandatory | 173 | 13 | | | | | |
| | Voluntary | 28 | 55 | 4 | 3 | 25 | 7 | 21 |
| New | Recommended | 243 | 118 | 5 | 33 | 30 | 7 | 46 |
| | Mandatory | 242 | 108 | | | 30 | 8 | 75 |

## 8.5 Applicability of the new EVAQ model

### 8.5.1. Dynamic Network Loading model

The DNL model was adapted and tested within EVAQ. Attention was paid to the special circumstances that evacuations create. The resulting DNL model is however a module of EVAQ that could easily be used in any other DTA model. A common assumption is that of capacity conditions. This assumption is equally valid for normal circumstances. The capacity condition itself may however need a different definition in terms if link capacity, saturation flow, conflict group capacity, average headway between following vehicles, minimum gap acceptance etc. The *mechanism* of human behaviour will not be different between evacuations and normal circumstances.

### 8.5.2. Reversed engineering

A merit concerning evacuations is that the new DNL focuses on actual phenomena such as lane choices and vehicle interaction. The model could thus function as a reversed engineering tool to evaluate where most friction is present and where measures would be effective to decrease evacuation time. Turn movements could be prohibited, possibly taking away all friction on an intersection. Of course this comes at the cost of having less route possibilities. Also for regular conditions there are reversed engineering opportunities. The weaving model for example clearly shows that the main cause of a lower capacity is inefficient lane choice behaviour. If drivers could somehow be stimulated to avoid the critical lane well before the weaving section and before congestion actually starts, capacity could be significantly increased. Without it there is also a stimulus to change lanes as the critical lane will be over saturated but this is, by the definition of the stimulus, too late. Besides functioning as a model component, the weaving model is also able to estimate capacity for design purposes. The turbo roundabout model shows the cause of increased capacity with respect to a 2-lane roundabout with more detail than just ‘less conflict points’. For specific lanes it is made explicit where the capacity increase comes from.

### 8.5.3. Additional input

The newly introduced precision of EVAQ requires more detailed input. The network must resemble the actual network where nodes are actual intersections. Often for macroscopic models, a single node may represent a cluster of intersections. For example intersections that are simply rather close to one another or a complete highway junction. Such nodes cannot be modelled correctly, as the assumed interaction is unrealistic. It is however possible to define such nodes as type ‘none’. Interaction will then not be taken into account. This at least provides access to CBQ for such networks. Clearly it would be better to define the network in line with reality. Besides the obvious consequences of additional computation time due to the additional links and nodes, there is also a problem that these situations often have short links. As explained in section 5.1.3, short links are now possible. The node model requires data about the nodes to analyse the conflicts. The input required is difficult to generate by hand as there may be many conflicts that need to be defined in one or several small matrices. It is easy to confuse which row and column or even additional dimensions represent a given conflict. To generate the node input a utility was developed that is described in appendix B. It allows graphical and semi-automatic generation of the input. Additional input for the entire model is listed in Table 8.4.

Table 8.4: Additional input

| **Element/Component** | **Input/Parameter** |
|---|---|
| **Network-wide** | |
| Links | Jam density per lane ($k_{jam}$) |
| Controlled intersections | Conflict capacity ($C_{conflict}$) |
| Uncontrolled & priority intersections | Minimum gap acceptance ($t_{critical}$) |
| | Average headway ($h$) |
| Roundabout | Alpha curve |
| | Beta for 1 and 2 lanes |
| | Gamma for 1 and 2 lanes |
| Weave model | Lane change utility ($u_{lc}$) |
| | Taper utility ($u_{ta}$) |
| | Weaving fraction at peak ($f_{co}$) |
| | Peak capacity ($C_{peak}$) |
| **Link model** | |
| All links | Saturation flow ($q^{sat}$) |
| **Node model** | |
| None | Turn matrix [optional] |
| Controlled intersections | Lane map per link |
| Uncontrolled & priority intersections | Lane map per link |
| | Priority (yes/no) per link |
| Roundabouts | Type (1-lane/2-lane/Turbo) |
| 1-lane & 2-lane roundabouts | Pseudo conflict distance per link |
| | Number of lanes (at the node) per link |
| Turbo | Pseudo conflict distance per lane |
| | Layout drawing |
| Weaving sections | Merge taper (yes/no) |
| | Diverge taper (yes/no) |

### 8.5.4. Permitted conflicts

The node model was designed for several types of intersections. In reality more complex intersections may be found. A rather common intersection type is controlled intersections with permitted conflicts. Usually such conflicts are only found for small flows. They can be modelled either as controlled (not permitted) or not existent. The first has consequences for the assumed number of green phases while the latter takes away a turn possibility.

## 8.6 Conclusions

This chapter has evaluated several aspects of the new EVAQ model. Section 8.1 has shown that the shockwave theory is correctly implemented in CBQ. Behaviour is as expected. Section 8.2 showed similar results for the various node sub model where link outflows had expected proportional or skewed patterns. Outflow capacities were within a reasonable margin of error. The new model is a significant improvement compared to the old model. Capacities and queue lengths resemble VISSIM data much closer. The two derived MFD's of the new model show much more resemblance with each other, suggesting a single MFD for the given network. Daganzo & Geroliminis (2008) theorized this as being a property of networks. Their shape of the MFD is however different.

The improvements of EVAQ come at the cost of additional CPU time and memory use. For the most critical scenario, voluntary evacuations, the gain is reasonable with 36% additional calculation time. Other scenarios show large relative gains in calculation time, but the gain is reasonable in absolute terms.

The DNL of the new model can be used in other DTA models. It also provides a good basis for reversed engineering as the DNL is theory based and relies on realistic network. The added precision does however require additional input. The node input, error prone if manually given, can be given with a semi automated graphical utility.

# 9. Conclusions & recommendations

## 9.1 Conclusions

The research questions from the introduction can be answered using the findings of this research. The research questions of phase one have been answered in chapter 3. For completeness the answers are given again. The research questions of phase two will also be answered. The research objective is revisited. From the answers to the research questions it may be concluded that the objective has been reached.

### 9.1.1. Phase one

*What processes influence network performance degeneration?*
These processes are discussed in section 3.1 and are flow decrease with density increase (fundamental link diagram), capacity drop for congested traffic, spillback and gridlock. All these processes originate from congestion. From section 3.4.2 it may also be concluded that capacity constraints of both the links and the nodes can trigger congestion.

*What processes are explicitly modelled in EVAQ?*
From chapter 2 it follows that the node model has two causes for congestion that both relate to the maximum link inflow. Either the link capacity is exceeded or the link is fully congested. Congestion is thus explicitly triggered and spillback is explicitly modelled.

*What processes are not explicitly modelled, but are an effective part of EVAQ?*
Related to spillback, gridlock is also an effective part of EVAQ. It has been shown in section 3.3 that the single reduction factor enables blocking cycles.

*What processes need to be included in order to achieve better accuracy?*
Remaining processes that are not covered by EVAQ are flow decrease with density increase, capacity drop and capacity constraints of intersections.

### 9.1.2. Phase two

*What solutions can be created to include additional processes?*
Chapter 4 has mentioned several ideas to include the remaining processes. Solutions were:

- Using an average congested state
- Directly implement a fundamental diagram
- Cell Based Queuing
- Congested outflow limits
- Additional constraints in the node model

The last three have been selected on the basis of realism and with a preference for theory based models.

*What assumptions need to be made for these solutions?*
*Are these assumptions more realistic considering network performance degeneration than the assumptions they avoid?*
A comparative overview of assumptions of the old and new model is given in Table 9.1. The new assumptions are more realistic.

| | **Old EVAQ** | **New EVAQ** |
|---|---|---|
| Maximum link inflow limit | Capacity & remaining storage. | Id. Remaining storage is however dependent on queuing behaviour. |
| Potential link outflow limit | Capacity. | Capacity for free flow. For congestion it is dependant on speeds through the queue. |
| Queuing behaviour | At a fixed jam density. | Kinematic shockwaves initiated by outflow. Traffic states are deduced via a triangular fundamental link diagram. Speed in the first cell is determined by saturation flow. |
| Spillback | Single reduction factor. | Id. Reduction factor is however dependant on maximum link inflow. |
| Lane choice behaviour | Not significant. | Equal lane flows. Shared lanes with higher flow are not used. |
| Controlled intersections | Not significant. | Shared use of conflict space with an effective capacity of 1300 pcu/h. |
| Uncontrolled & priority intersections | Not significant. | Minor capacity dependant on the sum of major flows. |
| Roundabouts | Not significant. | Entrance capacity dependant on conflict and pseudo conflict. The same model can be used for turbo roundabouts at lane level. |
| Weaving sections | Not significant. | Peak demand determined by movement utilities assuming saturated conditions. |

Table 9.1: Old and new assumptions

It should be mentioned that most sub models are based on existing models and formulas for regular conditions while EVAQ is about evacuation conditions. It is difficult to determine into what extend parameters will change. A good example is for instance gap acceptance. It may be expected that drivers care less about forcing vehicles with priority to decelerate.

*Are the processes indeed significant for network performance degeneration?*
Section 8.3 has shown that both the new link and node model contribute significantly to results that are closer to VISSIM results. The capacities and queues show much more NPD in the new MFD's. The included processes are thus indeed significant for NPD.

*What are the consequences of the solutions on calculation time and memory use?*
Both increase as more detail is taken into account. The critical scenario, voluntary evacuations, has an increase of 36% in calculation time. Memory use has increased similarly. EVAQ is still reasonably fast and

given the increase in realism, the additional calculation time can be justified.

### 9.1.3. Research objective

Below the research objective from the introduction is given.

Research objective

> To develop modelling solutions that correctly include processes that contribute to network performance degeneration in order to improve the accuracy of EVAQ and other DTA models.

The research objective has been reached. The new EVAQ model is more detailed and has a higher level of accuracy. The new DNL is theory based enabling a better assessment of evacuation plans. A new weaving model and theory is also introduced. As the new DNL relies on realistic networks with nodes representing actual intersections it is possible to apply reversed engineering to optimise evacuation measures. The DNL is also applicable in other DTA models enabling the same benefits for regular circumstances. The higher level of detail requires additional input that can be graphically given for the nodes. Calculation time and memory use have increased with about 1/3rd. This is justified by the increased accuracy.

## 9.2 Recommendations

The recommendations are divided into two sections. The first section focuses on the theory side of traffic modelling whereas the second section focuses on the implementation. The traffic modelling recommendations will focus on further research, further development of the new DNL model and further development of other EVAQ components. The recommendations regarding the implementation focus on the code structure of EVAQ and further development of the Node Input Generator.

### 9.2.1. Modelling recommendations

*Further research regarding assumptions*

From section 8.2 it follows that the DNL model produces good results. Some link specific capacities however show large errors. Additionally it should be mentioned that not all details of the model were extensively researched. For the following assumptions it is recommended that further research will be performed:

- *Fixed saturation flow*
  Saturation flow is given per link and fixed. Whether saturation flow is fixed has not been proven. It may well be that at a highway the congestion discharge rate is different if the queue drives at 10 or 60 km/h.
- *No influence of waiting areas at intersections*
  Uncontrolled and priority intersections often have a waiting area in the centre. These are assumed to not influence capacity, as vehicles might also have to wait for this area to clear. They are constructed to allow easier intersection crossing but according to the exponential minor capacity formula, major

flows can simply be added and any area in between is not of influence. Note that this may only hold if the flow is critical as the waiting area itself is than often occupied.

- *No spillback bias due to turn lane layout*
  Turn lanes are assumed to have no length. In reality they obviously have length that is used as a buffer during the red phases. Explicit red and green phases are ignored and averaged; turn lanes do not have to function as a buffer. Turn lanes can however differentiate spillback for different turn flows from the same link. It is not expected that this is significant as turn lanes are quickly filled if spillback has any significance. This has however not been shown.
- *First order turn lane choice*
  In urban networks the assumption that drivers divide equally over available turn lanes will often not hold. Second order elements such as downstream intersections and lane reductions will influence the preference of drivers. In under saturated conditions such behaviour is very real. What exactly the aggregated result is at saturated conditions is unknown. It might be that a bias exists but that the bias is small enough to allow enough drivers to use the less attractive turn lane. FIFO will than not hold but conflicts at the intersection are modelled with correct partial flows. It should be investigated how significant the second order influences are.

*Further development of the DNL model*
Besides investigating the validity of assumptions, it is also important to look at opportunities to further improve the DNL model.

- *Further development of the weaving model using real data*
  The weaving model has been calibrated to data from the microscopic model FOSIM. It is recommended to enrich the model using real data. This can especially be useful to recognize lane choice and lane change behaviour. This would require number plate data or similar data with other detection technologies. Detector data alone will not be as useful. It is also recommended to research the influence of weaving section length and if the logit scale factor can be used to capture this effect.
- *Reduction bias at weaving sections*
  Some link specific capacities from the node model are a fairly large under or over estimation. These are found for weaving sections and controlled intersection. The errors at weaving sections mainly follow from a single reduction on both merging roads. In reality there is often a bias in the influence as non-critical lanes can be utilized into different extents. Links to weaving sections can easily be the bottleneck for an entire region. A capacity error of 40% is than very undesirable. It is recommended that the weaving model is extended with an effect bias module. The bias may for instance be determined differently if the critical lane is directly downstream of the road or not. Also a mechanism for lane interaction could be determined. Finally it is recognized that merging tapers or clear

on-ramps (rather than highway merges) result in different driver behaviour.

- *Turn lane specific reduction factors for controlled intersections*
  For controlled intersections large errors are made for some but not all links. In the Syllabus CT4822 many turn lane specific reduction factors are given that may significantly reduce saturation flow. It is recommended that the controlled intersection model will be extended with the detail of turn lane specific reductions. These reductions should not actually reduce capacity, but virtually increase the flow as multiple flows are subject to a single conflict capacity. The end effect is obviously the same. Most detailed reductions can be determined once prior to the model run. For shared lanes the reduction is dependent on demand and should thus be calculated for each time step. The total additional calculation time will be very minor.
- *Permitted conflicts*
  Controlled intersections often have permitted conflicts, including the U-turns. U-turns are currently only covered by maximum link inflow. Other permitted conflicts are either modelled as a non-permitted conflict or they are excluded all together. It is recommended that an investigation is performed into the influence of permitted conflicts and that any significant mechanisms are included in the controlled intersection model.

*Further development of EVAQ*
Changes to EVAQ that have been implemented in this research focus on the DNL module. The improved DNL both enables and relies on further development of other EVAQ modules. The results of EVAQ will only be as good as the least accurate module of EVAQ. Therefore the quality of evacuation plans may rely more on other modules. Other modules should thus be further developed, or at least be investigated for their accuracy. Researchers can rely on a more accurate DNL while further developing EVAQ.

*Implications for findings using the old model*
Estimations for evacuation time and the number of casualties of the old model will generally be too positive. This research has resulted in lower capacities at nodes. This logically results in longer queues. Additionally, queues are even longer as densities are lower. More spillback and NPD results. Measures to improve evacuation time might be needed where formally thought unnecessary. In case of a disaster this could potentially cost human life. However, as mentioned, other modules of EVAQ might form the accuracy bottleneck.

### 9.2.2. Implementation recommendations

Two aspects of the new EVAQ will benefit from a better implementation. The first is the coding structure of EVAQ and the second is the general development of the Node Input Generator.

*Code structure*
EVAQ is programmed in Matlab, which is understandable as it works intuitive and is easy to learn. Developers of traffic models can thus focus on traffic theory instead of the programming language. Matlab gives the user a lot of freedom, which often leads to indistinct, less maintainable and possibly inefficient code. EVAQ is no exception as the split fraction generation shows in section 8.4. The code is indistinct because very similar blocks of code exist in multiple places. For voluntary and for recommended/mandatory evacuations the main model loop, the route set generation and the split fraction generation have separate sections while the differences are often very minor. The main difference is the existence of a class dimension for recommended/mandatory evacuations. Such code organisation is indistinct and leads to bugs because changes may not be (equally) applied at all instances of the code. It is better to define an algorithm at one location. This also makes the code much easier to maintain. For EVAQ this can easily be achieved by allowing the class dimension to be singleton for voluntary evacuations. A good framework to use is object-oriented programming. As of version 2008a, Matlab allows objects that are easily defined and applied along with regular script-like code. Currently, the TUDelft uses version 2007b while Rijkswaterstaat uses 2008b. At this moment it is thus not feasible to recode EVAQ in an object-oriented framework in MATLAB. Other programming languages will create large hurdles for further development. It is therefore recommended to restructure EVAQ such that all modules are defined within a single script or function. Care should be taken to minimise memory use when forwarding data from one function to another, as MATLAB will in some cases copy variables.

*Node Input Generator*
The Node Input Generator was developed within a small time span and covers a minimum of features. Input errors can be made, as there are only a few rough checks within the program. Furthermore, once node input is accepted, there is no way to graphically check the resulting input. There has also been no feedback by end-users if the program works as expected and is generally user friendly. It is recommended that the Node Input Generator will be further developed.

# 10. Bibliography

Bliemer, Michiel C.J.  (2007) – Dynamic Queuing and Spillback in an Analytical Multiclass Dynamic Network Loading Model; *Transportation research record, 2007, no. 2029, pp. 14-21*

Bliemer, Michiel C.J.; Taale, Henk (2006) – Route Generation and Dynamic Traffic Assignment for Large Networks; *Proceedings of the first international symposium on DTA, pp. 90-99*

Cova, Thomas J.; Johnson, Justin P.  (2002) – A Network Flow model for Lane-based Evacuation Routing; *Transportation Research Part A: Policy and Practice, vol. 37, pp. 579-604, 2003*

Daganzo, Carlos F.; Geroliminis , Nikolas (2008) – An Analytical Approximation for the Macropscopic Fundamental Diagram of Urban Traffic; *UC Berkeley Center for Future Urban Transport: A Volvo Center of Excellence*

Daganzo, Carlos F. (1993) – The Cell Transmission Model; A Dynamic Representation of Highway Traffic Consistent with the Hydrodynamic Theory; *Transportation Research Part B: Methodological, vol. 28, pp. 269-287, 1994*

Daganzo, Carlos F. (1993) – The Cell Transmission Model; Part II; Network Traffic; *Transportation Research Part B: Methodological, vol. 29, pp. 79-93, 1995*

Durlin, Thomas; Henn, Vincent (2007) – Dynamic Network Loading Model with Explicit Traffic Wave Tracking; *Transportation research record, 2008, no. 2085, pp. 1-11*

Dijker, T.; Knoppers, P. (2004) – FOSIM 5.0 Gebruikershandleiding, FOSIM 5.0 User manual; *www.fosim.nl*

Lenz , H.; Sollacher, R.; Lang, M. (2001) – Standing Waves and the Influence of Speed Limits; *IEEE Transactions on Intelligent Transportation Systems, 2005, vol. 6, pp. 102-112*

Pel, Adam J.; Bliemer, Michiel C.J.; Hoogendoorn, Serge P. (2008) – EVAQ: A New Analytical Model for Voluntary and Mandatory Evacuation Strategies on Time-varying Networks; *11th International IEEE Conference on Intelligent Transportation Systems, 2008. ITSC 2008, pp. 528-533*

Planung Transport Verkehr AG (2008) – VISSIM 5.10 User Manual; *www.ptv.de*

Qian, Xiaoyu (2009) – Application of Macroscopic Fundamental Diagrams to Dynamic Traffic Management; *ITS Edulab, Rijkswaterstaat & Delft University of Technology (www.its-edulab.nl)*

Rakha, Hesham; Zhang, Yihua (2006) – Analytical Procedures for Estimating Capacity of Freeway Weaving, Merge, and Diverge Sections; *Journal of Transportation Engineering, vol. 132, no. 8, 2006, pp. 618-628*

Rakha, Hesham; Zhang, Yihua (2005) – Systematic Analysis of Weaving Section Capacity; *Transportation Research Board 84th Annual Meeting, Washington D.C., CD-ROM [Paper 05-0916]*

Rijkswaterstaat (2007) – Nieuwe Ontwerprichtlijn Autosnelwegen (NOA), New Design Guidelines for Highways; *www.verkeerenwaterstaat.nl*

Southworth, Frank (1991) – Regional Evacuation Modeling: A State-of-the-art Review; *Oak Ridge National Laboratory, Energy Division, ORNL/TM-11740*

Taale, Henk (2008) – Integrated Anticipatory Control of Road Networks; *T2008/15, December 2008, TRAIL Research School, the Netherlands*

Vermijs, Raymond (1998) – New Dutch Capacity Standards for Freeway Weaving Sections Based on Micro Simulation; *Journal of Transportation Engineering, vol. 132, issue 8, pp. 618-628, August 2006*

Wardrop, J. G. (1952) – Some theoretical aspects of road traffic research; *Proceedings of the Institution of Civil Engineers, Part II, vol.1, pp. 352-362*

Wu, Ning (2001) – A new approach for modeling of Fundamental Diagrams; *Transportation Research Part A: Policy and Practice, vol. 36, pp. 867-884, 2002*

Yperman, Isaak; Immers, Ben (2003) - Capacity of a Turbo-Roundabout determined by Micro-Simulation; *Proceedings of the 10th World Congress on ITS, Madrid, Spain, November 2003*

Yperman, Logghe, Tampere, Immers (2005) – The Multi-Commodity Link Transmission Model for Dynamic Network Loading; *Proceedings of the 85th Annual Meeting of the Transportation Research Medel for Dynamic Network Loading, Washington DC, Jan 2006*

Zhang, H. M. (2000) – A note on highway capacity; *Transportation Research Part B: Methodological, vol. 35, pp. 929-937, 2001*

Syllabus CT4801 (2006) – P.H.L. Bovy, MC.J. Bliemer & R. van Nes – Transport modeling; *Delft University of Technology*

Syllabus CT4821 (2006) – Serge P. Hoogendoorn – Traffic Flow Theory and Simulation; *Delft University of Technology*

Syllabus CT4822 (2008) – Dynamic Traffic Management; *Delft University of Technology*

# Appendix A: EVAQ Algorithm Overview

In this appendix the modules reprisented in Figure A.1 will be elaborated. Products that are forwarded from one module to another will be indicated by boxed equations. First, the demand model will be described. The route choice model and the network loading (link and node model) follow after that. Finally, travel time estimation and route set generation will be discussed briefly.

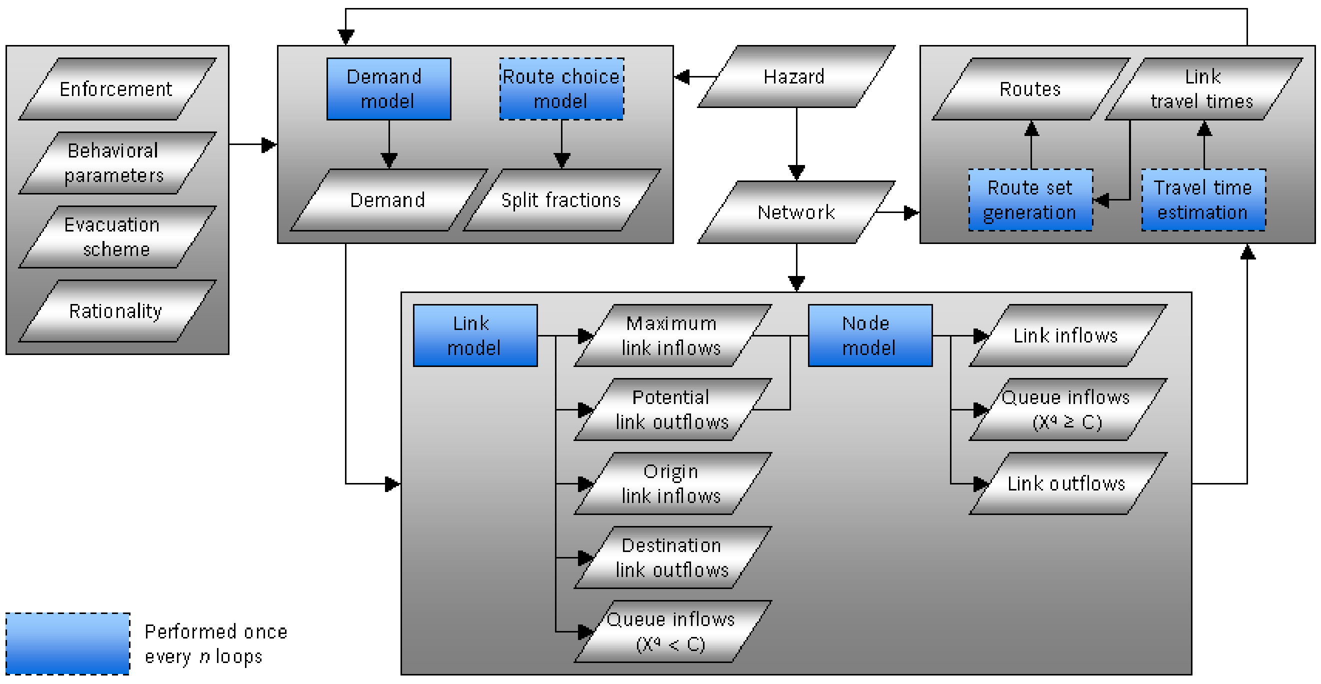


Figure A.1: EVAQ Framework

## Multiclass dynamic travel demand model

Depending on the status of the hazard, location and instructions, people will or will not leave before a certain time. This is a binary choice that is modelled with a binary logit model. Such a model requires, in this instance, a utility to leave, and a utility to stay. What actually determines the fraction of people that will have left at a certain time is given by the difference between the two utilities:

Equation A.1

$$V_m^{r,stay}(k)-V_m^{r,evac}(k)=\alpha_0-\alpha_1 \upsilon^r(k)-\alpha_2 \rho_m^{time}(k)\xi_m^{time}(k)$$

where,

| | |
|---|---|
| $V_m^{r,stay}(k)-V_m^{r,evac}(k)$ | Net utility to stay at origin *r* for class *m* at departure time *k* |
| $\alpha_0,\alpha_1,\alpha_2$ | Behavioural parameters |

$\upsilon^r(k)$ Thread by hazard(s) at origin *r* and time *k*

$\rho_m^{time}(k) = \omega_m^{time}(k)/(1-\omega_m^{time}(k))$

Level of enforcement for class *m* at time *k*

$\omega_m^{time}(k)$ Departure enforcement parameter [0…1] for class *m* at departure time *k*

$\xi_m^{time}(k) = \min_{k' \in K_m}(k-k')$

Time overlap with instruction for class *m* at time *k*

Also of importance is the level of rationality the road users inhibit during an evacuation. The response is modelled by an aggregated parameter.

Equation A.2

$$\mu^{time}(k) = \phi^{time}(k)/(1-\phi^{time}(k))$$

where,

$\phi^{time}(k)$ Departure response parameter [0…1]

The utility is multiplied by this parameter to represent the rationality of the aggregated users. The fraction of people that will have left origin *r* for class *m* at time *k* can now be calculated with the binary logit model.

Equation A.3: Demand as fraction of people

$$\boxed{\chi_m^r(k) = \frac{1}{1+\exp\left(\mu^{time}(k)\left(V_m^{r,stay}(k) - V_m^{r,evac}(k)\right)\right)}}$$

This proportion is multiplied with the population to determine the cumulative number of people that has left. The inflow of link *a* with origin *r* as tail node can thus be given by (see also Figure A.2, the link model):

Equation A.4

$$U_a(k) = P_r \cdot \sum_m \chi_m^r(k)$$

## Multiclass dynamic route choice model

Route choice is modelled with a path-size logit model. Route overlap is taken into consideration. Again a utility is needed, in this case to choose a certain route.

Equation A.5

$$V_{mp}^{ns}(t) = -\beta_0 \tau_p^{ns}(t) + \beta_1 \rho_m^{route} \xi_m^{s,destination} + \beta_2 \rho_m^{route} \xi_{mp}^{route}$$

where,

$V_{mp}^{ns}(t)$ Utility to take route *p* to destination *s* from node/origin *n* for class *m* at time *t*

$\beta_0, \beta_1, \beta_2$ Behavioural parameters

$\tau_p^{ns}(t)$ Route travel time from n to s on route p at time t

$\rho_m^{route}(k) = \omega_m^{route}(k)/(1-\omega_m^{route}(k))$

Level of route enforcement for class *m*

$\omega_m^{route}(k)$ Route enforcement parameter [0…1] for class *m*

$\xi_m^{s,destination}$ Destination *s* overlap factor (binary) for class *m*

$$\xi_{mp}^{route} = \frac{1}{l_p} \sum_{a \in A_{mp}} l_a$$

Factor for route overlap with advised or enforced route

$l_p$, $l_a$ Length of route *p*, length of link *a* (part of route *p*)

For considering the route overlap, a path size formula is used.

Equation A.6

$\lambda$ Path size parameter

$$z_{mp}^{ns}(t) = \sum_{a \in p} \left( \frac{l_a}{l_p} \frac{1}{N_{am}^{ns}(t)} \right)$$

Path size factor

$$N_{am}^{ns}(t) \equiv \left| \{ p' \in P_m^{ns}(t) \mid p' \supset a \} \right|$$

Number of routes in route set P using link *a*

For route choice, rationality is introduced similarly to departure time choice.

Equation A.7

$$\mu^{route}(k) = \phi^{route}(k) / (1 - \phi^{route}(k))$$

where,

$\phi^{route}(k)$ Route response parameter [0…1]

Route proportions from node/origin *n* to destination *s* for class *m* on route *p* at time *t* can be calculated with the path-size logit model using the above utility and factors.

Equation A.8: Split fractions

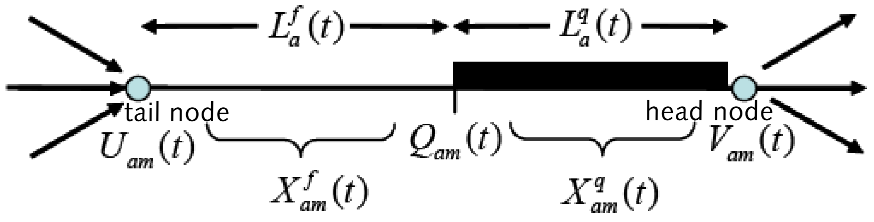
$$\boxed{\psi_{mp}^{ns}(t) = \frac{\exp\left(\mu^{route}(t)\left(V_{mp}^{ns}(t) + \lambda \ln z_{mp}^{ns}(t)\right)\right)}{\sum_{s' \in S} \sum_{p' \in P_m^{ns}(t)} \exp\left(\mu^{route}(t)\left(V_{mp'}^{ns'}(t) + \lambda \ln z_{mp'}^{ns'}(t)\right)\right)}}$$

Route choice is modelled not only at the departure, but also en-route. In the route choice model, every node is treated separately. Route proportions can thus easily be translated to split fractions at the nodes. All traffic from the incoming links is aggregated and divided over the outgoing links with these split fractions.

## Multiclass dynamic network loading model

The DNL model is derived from Bliemer (2007). It contains two sub models; a link and a node model. Links are split into two sections, a free flow section and a congested section as in Figure A.2. The congested part might have a length of zero. The link model determines how traffic propagates over the link and keeps track of the queue and what can potentially leave the link in a time step. The node model determines the amount of traffic that can actually cross the intersection, thus determining outflow and inflow of links ahead.

Figure A.2: Link model



The dynamic loading model is used slightly different from how it is described by Bliemer (2007). The main difference is the discretisation as opposed to a continuous definition. This paragraph describes the loading model as it is used in EVAQ. Important things to keep in mind are the following:

- The status of time step *t* is used to derive a new status for time step *t+1*
- A clear distinction between cumulative and momentary quantities should be made. Cumulative quantities are denoted as $\overline{Q}$ .
- For simplicity, classes are omitted.

**Link model**

First, the link model is applied on every link *a*. Potential link outflows are the main purpose of the link model. These potential flows depend heavily on the queue. The number of vehicles at time step *t* on the link is determined as the difference between the cumulative inflow and cumulative outflow.

Equation A.9

$$X_a = \overline{U}_a(t) - \overline{V}_a(t)$$

The maximum inflow is determined by either using the flow capacity or remaining storage capacity.

Equation A.10: Maximum link inflow

$$\boxed{U_a^{\max} = \min\left(C_a, X_a^{\max} - X_a\right)}$$

The number of vehicles in the queue can be calculated as the difference between the cumulative queue inflow and the cumulative link outflow.

Equation A.11

$$X_a^q = \overline{Q}_a(t) - \overline{V}_a(t)$$

With the number of vehicles in queue and a link queue density, a queue length can be acquired.

Equation A.12

$$L_a^q = \frac{X_a^q}{k_a^q}$$

Next, the potential outflows will be determined. Depending on the length of the queue, this might need the cumulative queue inflow of the next time step. If the number of vehicles in the queue is less than the capacity per time step, free flow traffic might also leave the link within the next time step. For links *b* with $X_a^q < C_a$, the free flow length and speed are used to determine τ, which is the (non integer) number of periods relative to *t* that vehicles now entering the queue, entered the link. The queue inflow is calculated by linear interpolation between $t - \lceil \tau \rceil$ and $t - \lfloor \tau \rfloor$.

Equation A.13: Queue inflows ($X^q < C$)

$$\boxed{\overline{Q}_b(t+1) = \overline{U}_b(t - \lceil \tau \rceil) + \left[(\lceil \tau \rceil - \tau) \cdot \left\{\overline{U}_b(t - \lfloor \tau \rfloor) - \overline{U}_b(t - \lceil \tau \rceil)\right\}\right]}$$

A new number of vehicles in queue is calculated for these links:

Equation A.14

$$X_b^q = \overline{Q}_b(t+1) - \overline{V}_b(t)$$

Note that this is not the actual number of vehicles in queue at time *t*, however it is the number of vehicles that determines the potential outflow from *t* until *t+1*.

All vehicles 'in queue', limited by the capacity, can potentially leave the link.

Equation A.15: Potential link outflows (*b*)

$$\boxed{V_b^{potential} = \min\left(X_b^q, C_b\right)}$$

For links *c* with $X_c^q \geq C_c$ the potential outflow is simply the capacity.

Equation A.16: Potential link outflows (*c*)

$$\boxed{V_c^{potential} = C_c}$$

For links *o* with origin *r* as tail node, the inflow is determined by applying the maximum inflow or the cumulative demand $P_n \cdot \chi^n(t)$ at node (origin) *n* if it is less.

Equation A.17: Origin link inflows

$$\boxed{\overline{U}_o(t+1) = \min\left(\overline{U}_o(t) + U_o^{\max}, P_n \cdot \chi^n(t)\right)}$$

Links *d* connected to destinations cannot have a queue. Therefore the cumulative outflow is equal to the cumulative queue inflow as calculated for links *a*. Links *d* form a subset of links *a* since $X_d^q = 0 < C_d$.

Equation A.18: Destination link outflows

$$\boxed{\overline{V}_d(t+1) = \overline{Q}_d(t+1)}$$

Potential outflows and maximum inflows have now been determined for all links. Also for links connected to origins, the inflow is determined. For links connected to destinations, the outflow is determined. The node model will handle all other nodes and connected link inflows and link outflows.

**Node model**

The node model is applied on every node *n*. The node has incoming links *a* and outgoing links *b*. The potential inflow of links *b* is calculated by applying the split fractions on the potential outflow from the link model of links *a*, or on the maximum cumulative outflow $\overline{V}_a^{\max}$ determined by the hazard (capacity reduction).

Equation A.19

$$U_b^{potential} = \psi^{ns}(t) \cdot \min_a\left(V_a^{potential}, \overline{V}_a^{\max} - \overline{V}_a(t)\right)$$

If the potential flow is higher than maximum flow on any link *b*, a ratio smaller than one is applied on all flows. This assumes that vehicles will wait at the intersection and not before the intersection. Otherwise a ratio of one is used.

Equation A.20

$$ratio = \min_b\left(\frac{U_b^{\max}}{U_b^{potential}}, 1\right)$$

Equation A.21: Link inflows

$$\boxed{\overline{U}_b(t+1) = \overline{U}_b(t) + ratio \cdot U_b^{potential}}$$

Equation A.22: Link outflows

$$\boxed{\overline{V}_a(t+1) = \overline{V}_a(t) + ratio \cdot U_a^{potential}}$$

The only thing that remains is calculating the queue inflows of links *c* from the link model. This is calculated similarly to links *b* from the link model.

Equation A.23: Queue inflows ($X^q \geq C$)

$$\boxed{\overline{Q}_c(t+1) = \overline{U}_c(t-\lceil\tau\rceil) + \left[(\lceil\tau\rceil - \tau) \cdot \left\{\overline{U}_c(t-\lfloor\tau\rfloor) - \overline{U}_c(t-\lceil\tau\rceil)\right\}\right]}$$

All link inflows, queue inflows and link outflows have now been calculated for time step *t+1* based on the status at time step *t*.

## Travel time estimation

Since time is of the essence in evacuations, travel time is an important measure used for route choice (optionally together with an instruction). Travel time is estimated by assuming a fixed speed for congested traffic.

Equation A.24: Link travel times

$$\boxed{\tau_{am}(t) = \frac{L_a - L_a^q}{\vartheta_{am}^{\max}} + \frac{L_a^q}{\vartheta^q} + \varphi \cdot \infty}$$

where,

| | |
|---|---|
| $\tau_{am}(t)$ | Instantaneous travel time of link *a* for class *m* |
| $L_a$ | Length of link *a* |
| $L_a^q$ | Queue length on link *a* |
| $\vartheta_{am}^{\max}$ | Maximum (free flow) speed on link *a* for class *m* |
| $\vartheta^q$ | Assumed queue speed |

|  |  |
|---|---|
| $\varphi$ | Link affected by hazard (0 or 1) |

Links that are hit by the hazard get an infinite travel time. This assures that these links will not be chosen.

## Route set generation

The route set generation algorithm was based on an algorithm by Bliemer & Taale (2006). For every OD pair, where all nodes are an origin, multiple routes will be generated. For every next route, all link costs are determined as the travel time estimation, but travel times are made stochastic. For this a normal distribution is used with increasing standard deviation for every next route starting with a standard deviation of zero. Routes are then generated using a shortest path algorithm. Duplicate routes will be omitted.

# Appendix B: Node Input Generator

The changes for the DNL model involve a lot of input for nodes that can be specific for the nodes, the entering links or even the entering lanes. Much of this input is graphically not complex, but has to be represented in many small two or three dimensional matrices that have a subset of the following dimension: entrance link, exit link, lane and conflict. Creating such input manually will most certainly generate errors as conflicts are easily overlooked and matrix elements are easily filled in a wrong column and/or row. For these reasons a small utility was developed that allows a graphical generation of the node input. The utility works from an existing network and will define node input dependant on the connecting links and the user input. Generating all node input will take some time, it is therefore wise to first define the network links and after that the nodes. Adding or deleting links afterwards will create inconsistencies between the actual links and the links that are assumed to exist for the node model. The Node Input Generator currently only works as a generator and not as a viewer/verifier of existing nodes and conflicts.

## The Graphical User Interface

When starting the node input generator program, a blank screen will appear. In the menu you can select ‘Network > Load network’ that lets you select a Matlab data file. The data file should at least hold the EVAQ data: links, nodes, coordinates & mapscale. A map of the network will be displayed on the screen as in Figure B.1 (left).

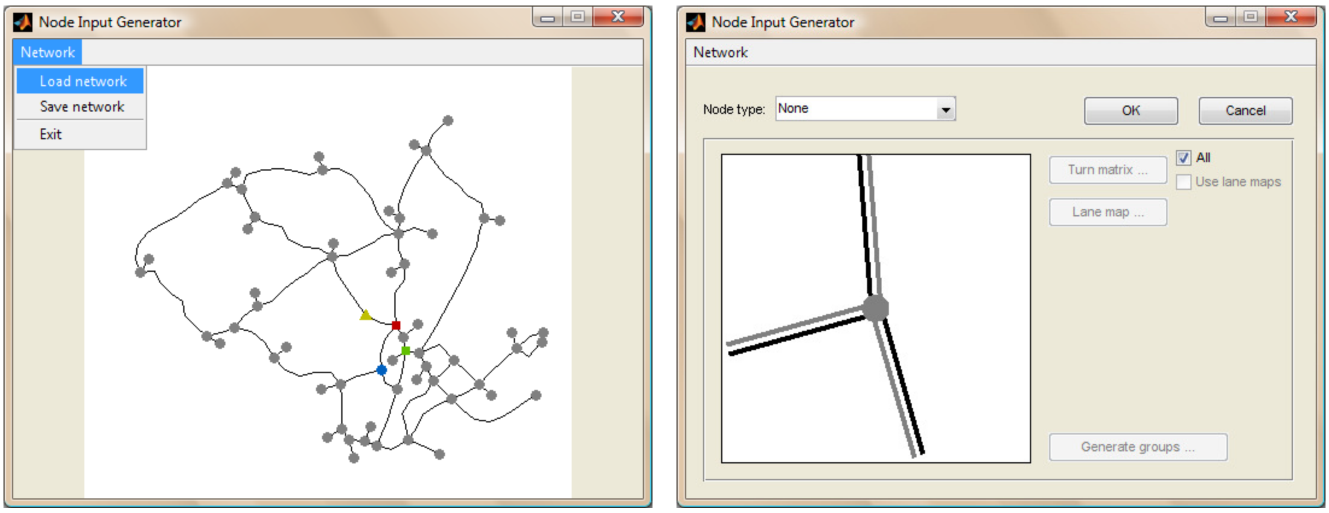
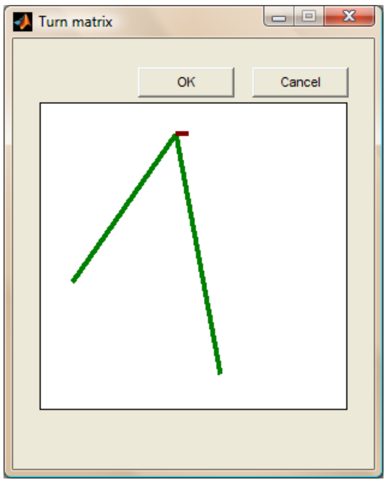


Figure B.1: Network view (left) and type ‘none’ (right)

By right-clicking on a node you can select ‘Define input’ that lets you generate the node model input. The screen will change to Figure B.1 (right). By default the node type is ‘None’ as can be seen in the node type selection box. This type will not have any simulated conflicts on the node. It can be used for origin nodes, destination nodes, nodes that function as a link in-homogeneity (for instance a change in maximum speed) and nodes that merely function as an entry from a connector but are not actually an intersection. Often it is not needed to explicitly define these nodes as 'None' as all nodes start as such a node. It is however sometimes needed to prohibit certain turns over the node. By selecting a link (it will become red) and un-checking the 'All' checkbox, the button 'Turn Matrix ...' enables. It will show a pop-up window as in Figure B.2 (left).



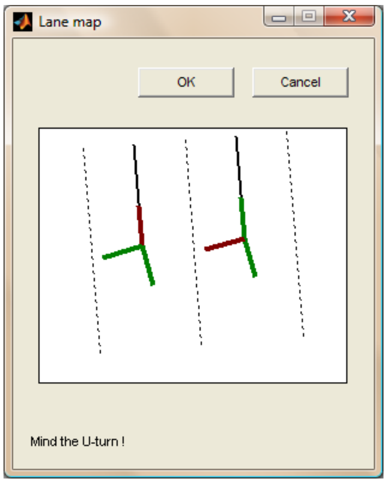


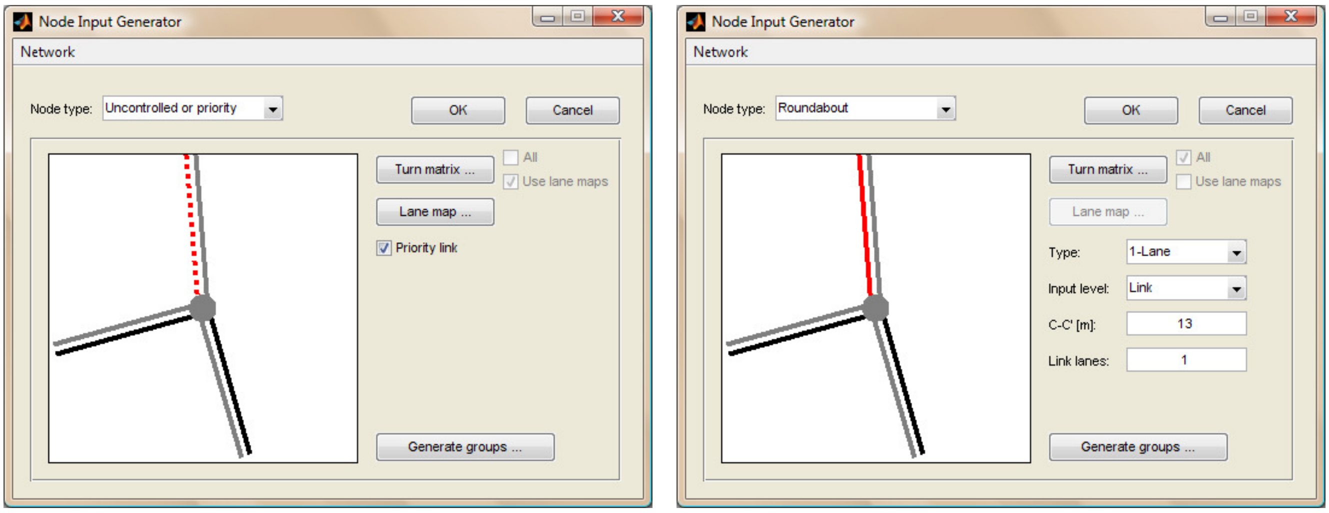
Figure B.2: Turn possibilities (left) and lane map (right)

By clicking on a turn direction in the pop-up window, the turn will be enabled (green) or disabled (red). In Figure B.2 (left) the U-turn from and to the north is disabled. Click 'OK' in the pop-up window to accept any changes. This process needs to be repeated for every link from which turns are impossible. If all turns are possible, select the 'All' option.

**Controlled intersections**

By selecting 'Controlled' in the node type selection box a few additional features become available. The node is now a controlled node for which turn lanes need to be defined. Similar to the turn matrix, turn lanes are defined per link. To define the turn lanes, select a link and click on the 'Lane map ...' button. You will be asked to give a number of lanes. After this a pop-up window will appear as in Figure B.2 (right). The number of lanes can be changed at any time by clicking on the 'Lane map ...' button again and giving another number of lanes. Any existing lane map will be discarded. If you give a number of lanes equal to the current lane map (default answer), the current lane map will be shown. In Figure B.2 (right) we see two turn lanes from the north. For each turn lane, lines in the directions of all downstream links are displayed. Green lines mean that from that turn lane, the given direction is possible. Red means that the direction is impossible. The status can be changed by clicking on the lines. Additional to the arrows on the road itself, a U-turn (if possible) should also be included on the left lane. By clicking 'OK' you accept the changes. The lane map will be checked. All lanes should be used and turn directions should not conflict. As there are lane maps for each link of a controlled intersection, the lane maps are used to define the turn matrix. The 'Use lane maps' checkbox is checked by default and the lane maps will be automatically translated into a turn matrix for the entire intersection. The ‘Generate groups …’ button will generate all valid conflict groups for the intersection based on all possible moves of the turn matrix. Also the effective conflict capacity fraction will be determined for groups that cannot be facilitated by all green phases. The algorithm behind this button will be explained in a following section for all node types the button applies to.

**Uncontrolled and priority intersections**
Uncontrolled and priority intersections are very similar to controlled intersections in the context of this program. The only difference is that for uncontrolled and priority intersections, links may be defined as being a priority link by selecting ‘Priority link’. The selected link will become dashed to indicate the priority as in Figure B.3 (left).


Node type: Uncontrolled or priority

Priority link [✓]

Node type: Roundabout

Type: 1-Lane

Input level: Link

C-C' [m]: 13

Link lanes: 1

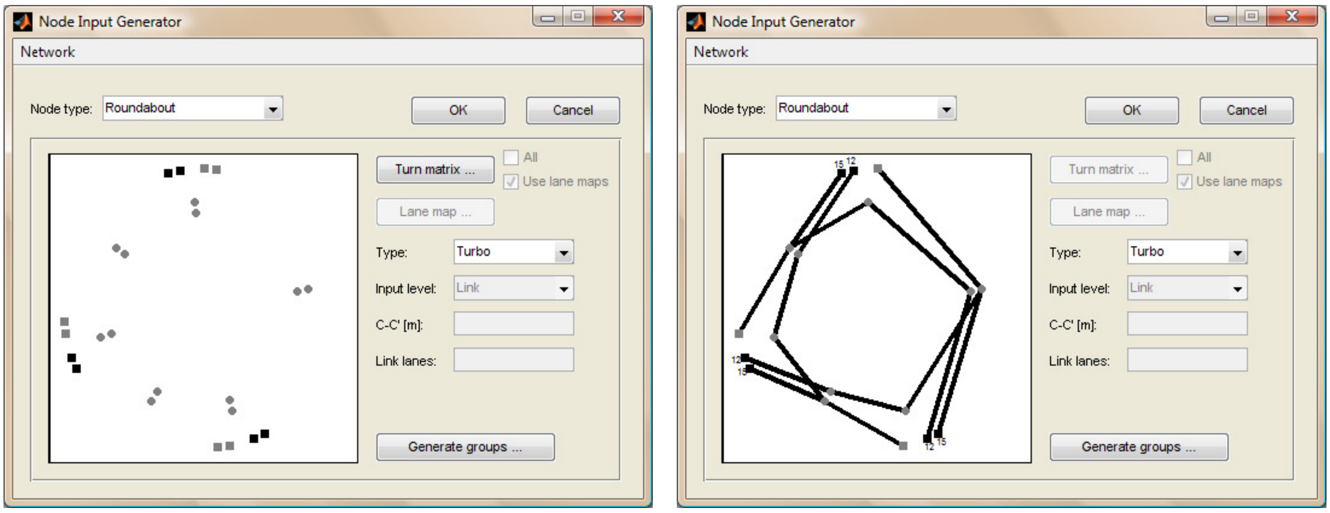
Figure B.3: Uncontrolled (left) and 1-Lane roundabout (right)

Another difference is the group generation. The ‘Generate groups …’ button will generate all valid minor and major flow groups per link and per lane. Minor flows are a set of lane specific partial flows, major flows are conflicting turn flows common among all related minor flows per group.

**Roundabouts**
Roundabouts are a very different story. First of all, a roundabout type needs to be defined. It can be 1-Lane, 2-Lane or Turbo. The first two options work similarly. The 1-Lane and 2-Lane roundabout model needs alpha, beta and gamma values representing the influence of the pseudo conflict, number of lanes on the roundabout and number of lanes on the link respectively. The beta value is thus related to the roundabout type and will be calculated automatically. Input for alpha and gamma can be given per link and/or for the entire node, see Figure B.3 (right). The latter is useful if for example most connecting links have 1 lane. Parameters that are not given for a link will be taken from the node. A mixture of node level and link level input is thus possible. To perform these actions on the link level, a link needs to be selected. The ‘Generate groups …’ button will generate groups of turn flows that make up $V_{exit}$ and $V_{circ}$ per link.

**Turbo roundabouts**

Turbo roundabouts can exist in many different forms. The same model as for 1-Lane and 2-Lane roundabouts is used but on lane level. To analyse which partial flows make up $V_{exit}$ and $V_{circ}$ and what the value for beta should be, the turbo roundabout needs to be drawn as a set of links between various nodes. Figure B.4 (left) shows the default node layout for the selected network node.



Node type: Roundabout

☑ Use lane maps

Type: Turbo

Input level: Link

Figure B.4: Turbo nodes (left) and complete with links (right)

Black squares are the entrance lanes while the grey squares are the exit lanes. The grey circles are lanes at the roundabout. Between any two links a set of roundabout nodes is given. At any set of nodes, a node can be added or removed by right clicking on any node within the set and selecting 'Add node' or 'Remove node'. Links can be created between the nodes by dragging from one node to another. The driving direction will automatically be determined as being against the direction of the clock. Links can be deleted by right-clicking on them and selecting 'Delete link'. It is good practice to set the right number of nodes at all sets before drawing the links. The drawing can be reset by temporarily setting the roundabout type to 1-Lane or 2-Lane. For each entrance lane, an alpha value needs to be given. This can be done by right-clicking on the node, selecting 'Set alpha' and giving the C-C' distance. Figure B.4 (right) shows an example turbo roundabout including the alpha values at the entrance lanes. The 'Generate groups ...' button will analyse the drawing to define a set of partial flows per lane that make up $V_{exit}$ and $V_{circ}$. Also the beta value for each lane will be calculated. Additionally, lane maps for all links and the turn matrix will be created. For this process to work, the turbo roundabout should be drawn correctly. Entrance lanes should only connect to the set of roundabout lanes in the downstream direction. Exit lanes should only connect to the set of roundabout lanes in the upstream direction. Finally, circular lanes should never skip a roundabout set. Incomplete roundabouts however can be defined, should they ever be encountered. Note that dog-bone and oval roundabouts are better modelled by a single roundabout with appropriate parameters for the pseudo conflict.

**Weaving sections**
A last node type is a weaving section, also used for on-ramps and off-ramps. This type is only selectable if the number of entering and exiting links is one or two. Also the number of incoming and exiting lanes may differ up to one. According to the difference in the number of lanes, either a merging taper or diverge taper can be indicated. If the number of lanes is equal it is possible to indicated both a merge and a diverge taper.

In the network view, all defined node types are made visible as in Figure B.1 (left).

| | |
|---|---|
| • Red square: | controlled intersection |
| • Green square: | uncontrolled or priority intersection |
| • Blue circle: | roundabout |
| • Yellow triangle: | weaving section |

These markers will also be visible after loading a network where conflicts are already defined. The specific node input will however not be shown after selecting 'Define input'. The network can be saved through the network menu.

## Conflict generation algorithm

The various algorithms to generate the conflict groups are quite extensive in size. For detailed information the reader is advised to look at the 'genGroups' function in the Matlab code, see appendix D. Comments are provided to explain the code, still knowledge of Matlab is needed. Here a general description of the various algorithms will be given.

**Controlled intersections**
Conflict groups at controlled intersections are groups where all turn flows intersect with all other turn flows in the group. A first step is to find all crossings between turn flows. This is performed by defining straight lines between the entrance and exit links in a circle with a radius of one. If two lines intersect at a location within the circle, the two turn flows cross. U-turns are ignored, as these are not taken into account for the design of traffic light phasing. Step one results in a set of 2-phase conflicts. The second step is to loop as long as larger conflicts are found. The first loop finds 3-phase conflicts by recognizing overlap between two 2-phase groups, but also a difference. Both groups should have one turn flow that is not in the other group. If these two turn flows are crossing themselves, all turn flows from the two groups form a group. This process also applies to larger groups. It is important to discard any conflict group that is a subset of a larger group. The next loop will find 4-phase conflict groups by analysing the 3-phase conflict groups. If larger groups are not found, the process ends. The set of conflicts is a matrix with the dimensions (entrance_link x exit_link x conflict) that is stored at the node level.

**Uncontrolled and priority intersections**
For these intersections, each group exists out of a set of partial flows from a specific lane and all common major flows. Step one is similar as for controlled intersection with the difference that U-turns are taken into account. Step two is to find all major flows per turn flow from a specific link. A turn flow is major if it comes from a priority link while the current link is not a priority link, or if it comes from the right while both links are or are not a priority link. Step three is to loop over many sets of partial flows, from a specific lane, with various sizes. For each set the common major flows should be found. Here it is also important to discard groups that are a subset of another group. Groups are stored as matrices per lane and have the following dimensions for minor (1 x exit_link x conflict) and for major (entrance_link x exit_link x conflict). The '1' for minor sets unifies the dimension order with major sets and is of course related to the link of the specific lane.

**Roundabouts**
At roundabouts conflicts are experience while entering the roundabout. Two flows are important, exit flow and circular flow. Exit flow will only exist if the first link to the left is an exit link. All turn flows towards this link are the exit flow. Circular turn flows are found by looping over all turn flows and analysing the total angle performed by a turn flow. If this angle is larger than the angle towards the entrance link in question, the turn flow goes past the entrance link and is thus part of the circular flow. Both sets of turn flows for exit flow and circular flow are stored in a matrix per link with the following dimensions: (entrance_link x exit_link).

**Turbo roundabouts**
The algorithm for turbo roundabouts involves the creation of lane maps, a turn matrix, beta values per lane, exit flow per lane and circular flow per lane. All of these are found by 'travelling' along the links. A common step in this process is to find all nodes (of all types) either upstream or downstream from a current set of nodes. Important is to also have a set of stop nodes that prevents movements over more than a full circle. For the lane map, the stop nodes are the first downstream set of roundabout nodes from an entrance node. Possible movements are found by travelling further and further downstream. Circular flow is found starting at an entrance node and moving downstream once. Stop nodes are all nodes at this cross section. Nodes with circular flow include all nodes to the right at this cross section, as flow towards these nodes crosses with the entrance movement. From the set of circular nodes, the algorithm travels upstream to find all partial flows that use the nodes. The number of nodes after travelling upstream the first time resembles the number of lanes of the circular flow and thus defines the beta value. Partial flows are not only defined by the lane and link they come from, but also by the link they go to. At every cross section, the algorithm travels downstream once to find exit nodes. The accompanying exit link will, from that moment on, not be an exit link for partial flows, as apparently these partial flows exit the roundabout instead of circulating further. Exit flow can be found by travelling upstream from the correct exit nodes. Stop nodes are defined just

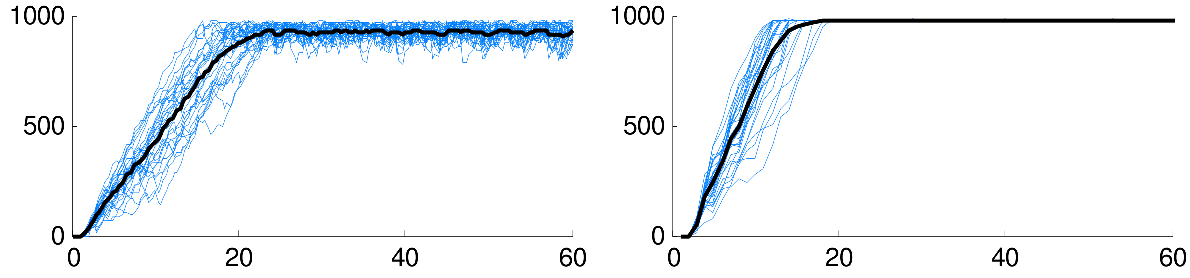
downstream of the exit nodes. What remains is to find the correct exit nodes in the first place. This starts very similar as with circular flow. After having travelled upstream twice over the roundabout nodes, the algorithm travels downstream once to find all exit nodes that hold flow that could potentially intersect with the entrance movement. Finally the turn matrix is derived from the lane maps. For each lane, the set of partial flows for the exit flow and circular flow are stored in matrices with the following dimension: (entrance_link x exit_link x lane).

# Appendix C: VISSIM comparison

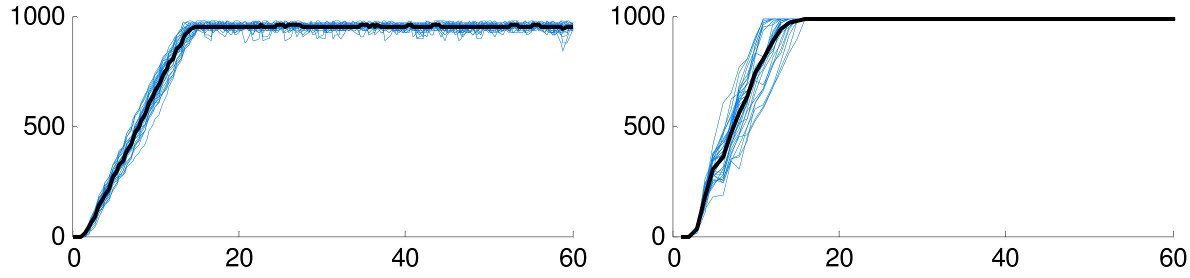
The following plots show the queue length (vertical axis in meters) through time (horizontal axis in minutes). For each link at each node type there is a separate plot that holds 30 model runs. EVAQ plots are to the left while VISSIM plots are to the right. Capacity errors (%) of the node model are also given. These largely explain the differences in slope. The black line is the average queue length.

**Controlled intersection**

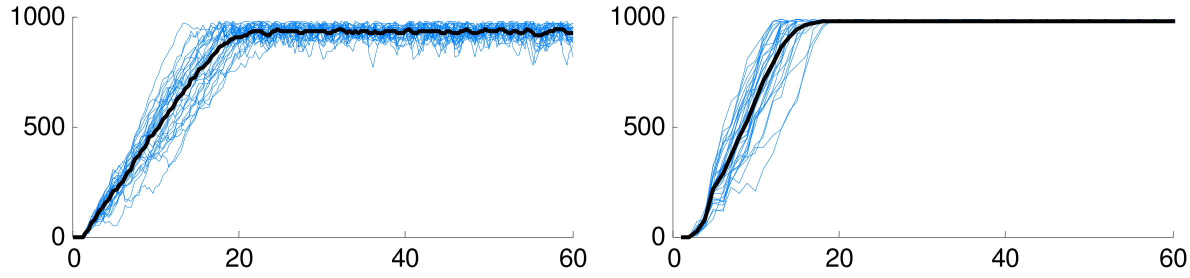
Link 1 (44%)



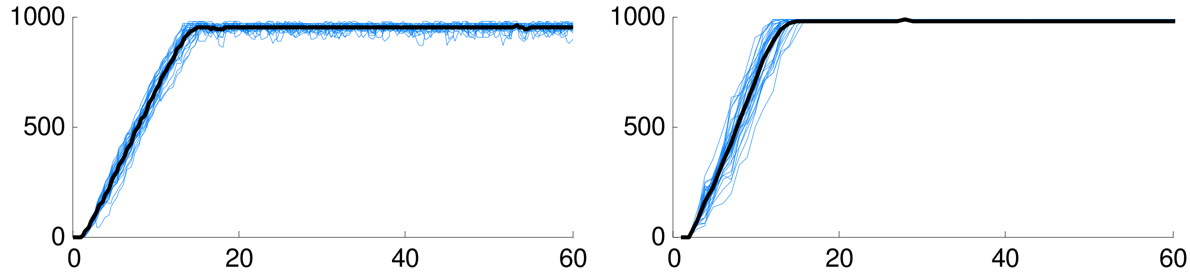
Link 2 (2%)



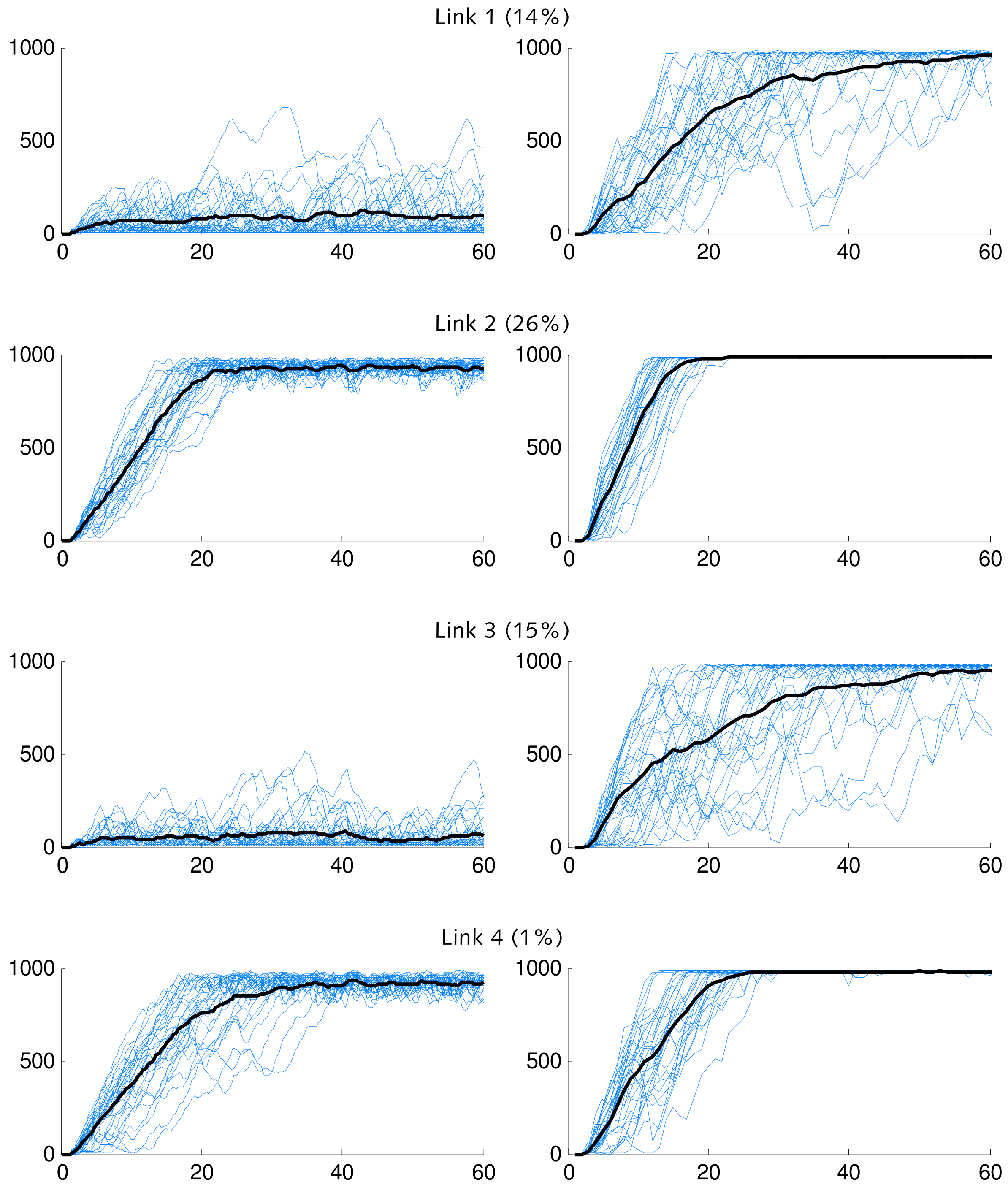
Link 3 (33%)



Link 4 (10%)

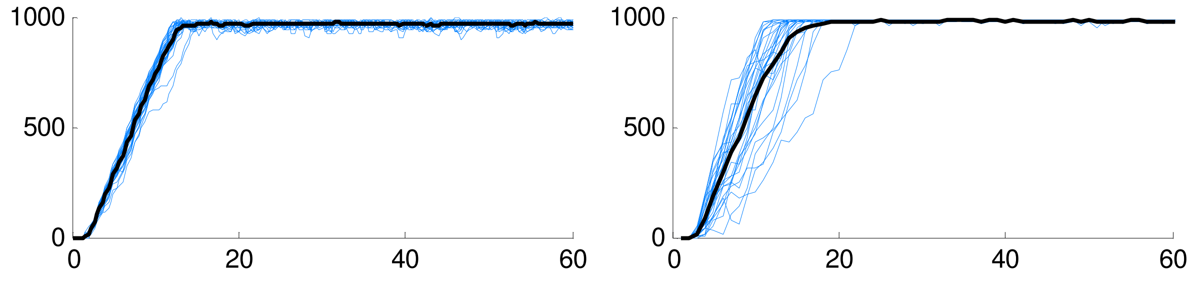


**Uncontrolled intersection**

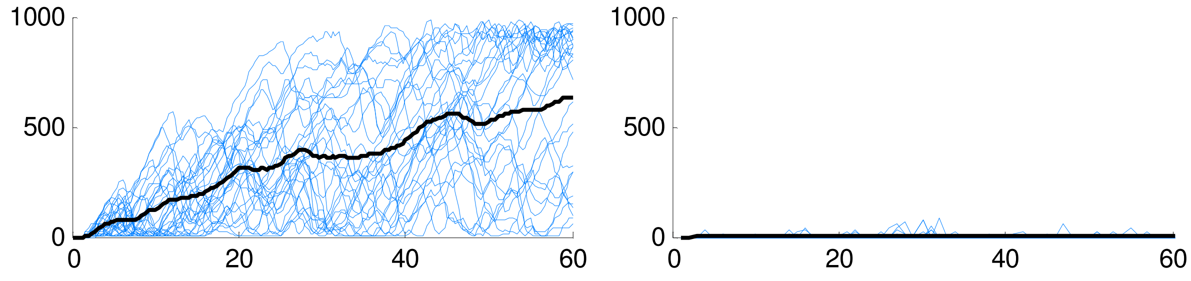


**Priority intersection**

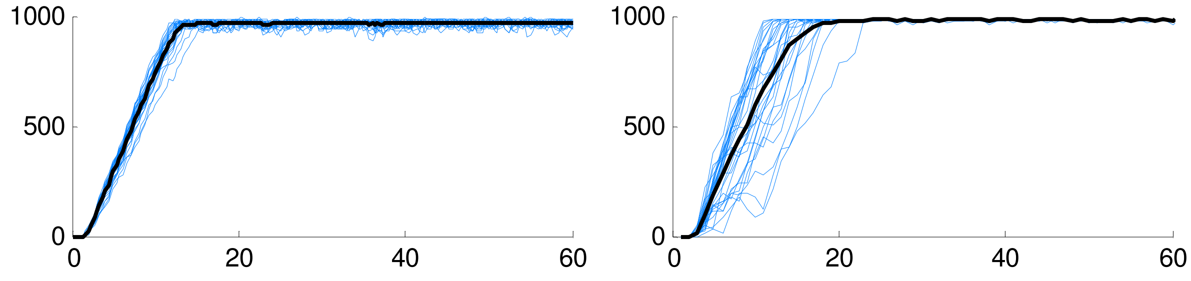
Link 1 (-22%)



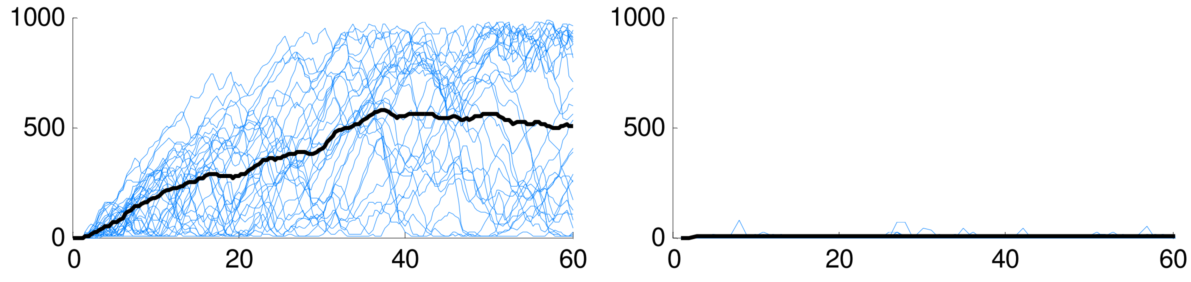
Link 2 (-5%)



Link 3 (-26%)

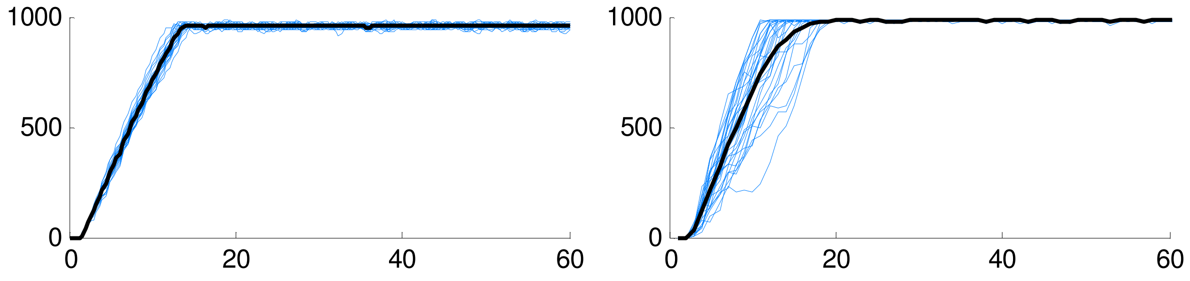


Link 4 (-7%)

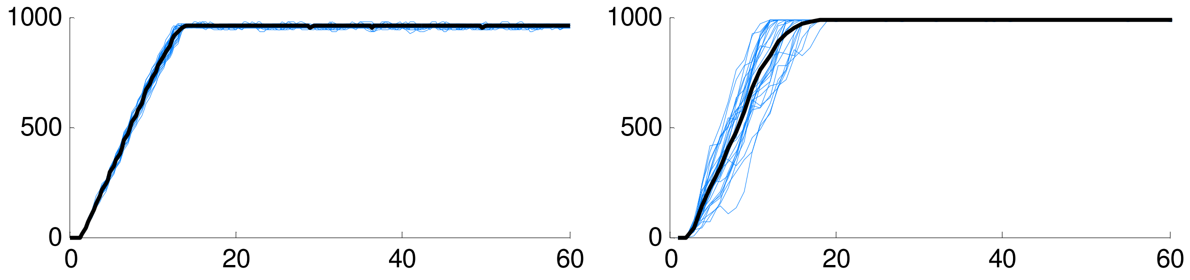


**2-lane roundabout**

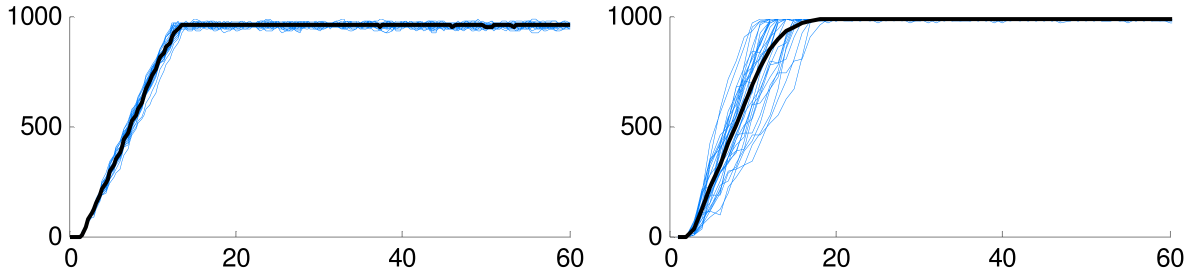
Link 1 (-12%)



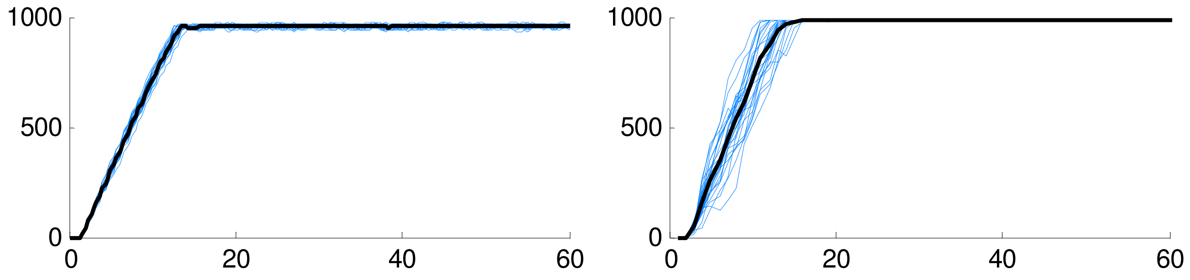
Link 2 (-14%)



Link 3 (-9%)

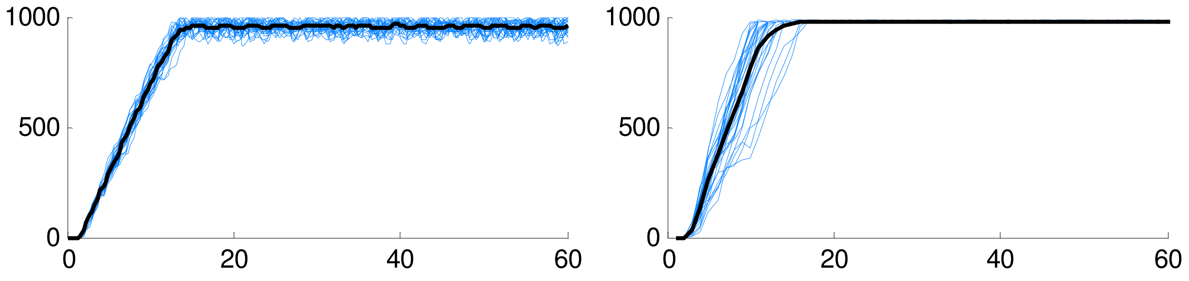


Link 4 (-6%)

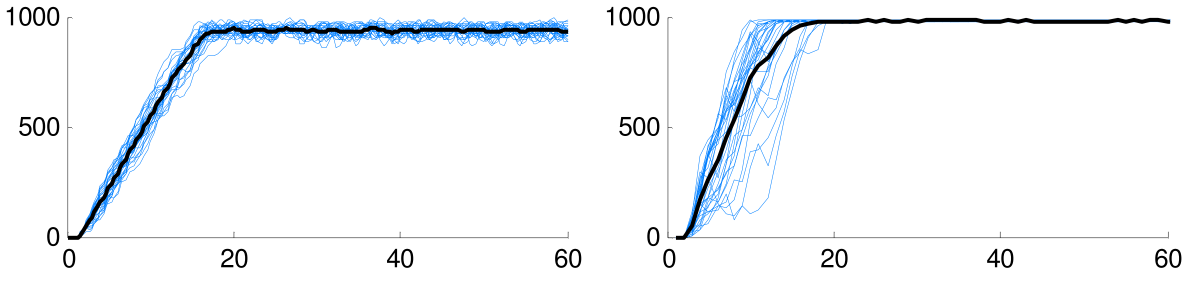


**Turbo roundabout**

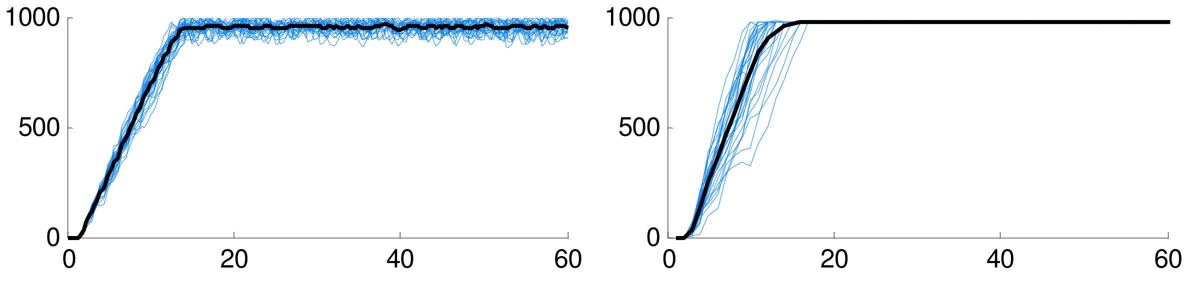
Link 1 (23%)



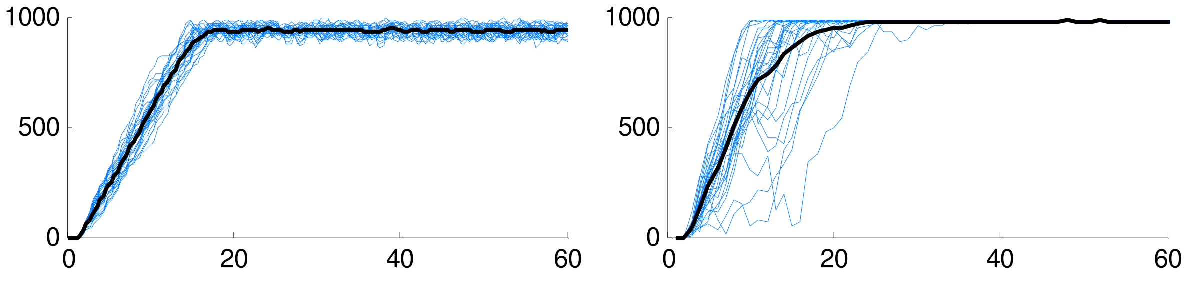
Link 2 (12%)



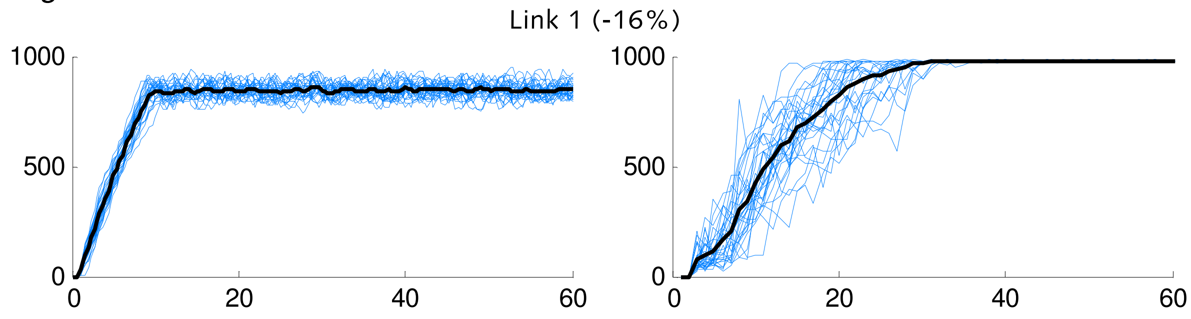
Link 3 (24%)

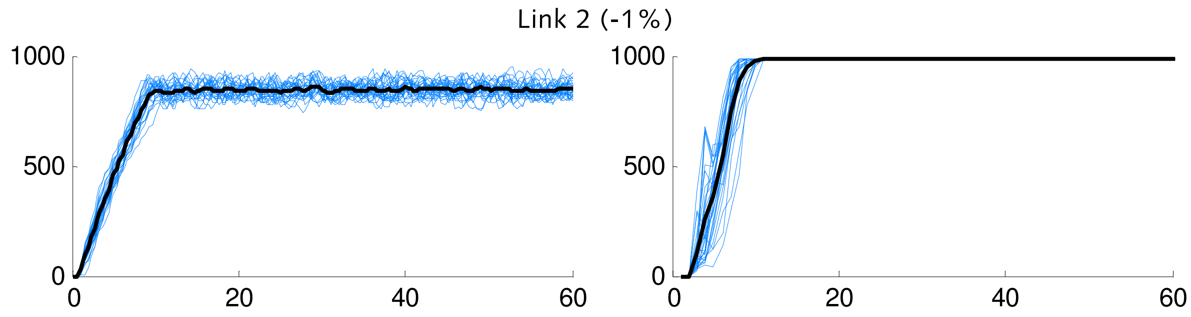


Link 4 (-4%)

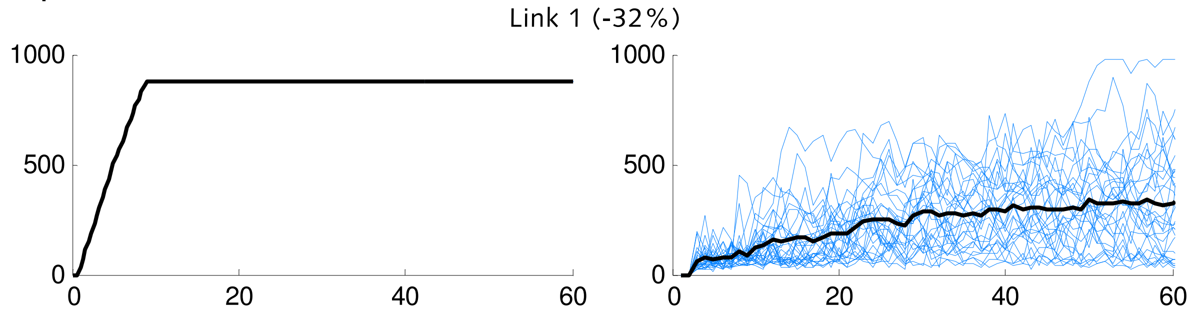


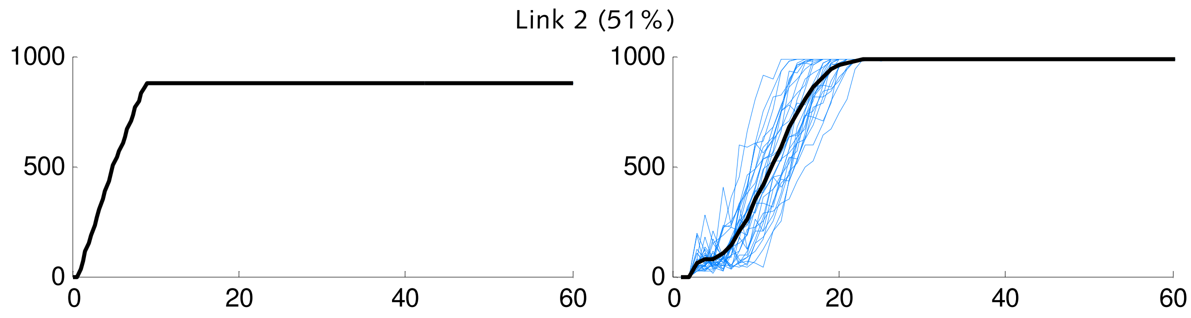
**Weaving section**



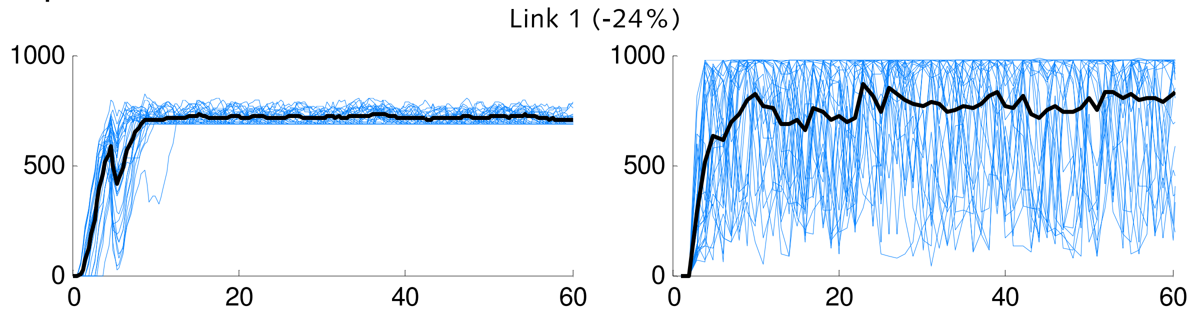


**On ramp**





**Off ramp**



# Appendix D: Matlab code

Appendix D holds most code that resulted from the various developments. Code is given from:

- EVAQ initialisation
- Cell Based Queuing
- Node model
- Node Input Generator

Most code is commented and should be self-explanatory. Matlab experience, or at least some programming experience, is required.

## EVAQ initialisation

Two parts of the initialisation code are specific to the new link and node model. The first part defines the node model parameters while the second part performs initial steps of the link and node model.

```
% Node model parameters
CONSTANTS.ConflictCap = 1300;          % controlled:  conflict group capacity [pcu/h]
CONSTANTS.MinGapAcceptance = 4/3600;   % minor/major: flow gap acceptance [h]
CONSTANTS.AverageHeadway = 2/3600;     % minor/major: average headway of following vehicles [h]
CONSTANTS.RoundaboutAlpha = [0 9 21 27 28 inf; .6 .6 .1 .1 0 0]; % must be full positive range
                                       % roundabout:  multi-linear alpha curve (vs. C-'C [m])
CONSTANTS.RoundaboutBeta = [0.95 .7];  % roundabout:  beta for 1 or 2 roundabout lanes
CONSTANTS.RoundaboutGamma = [1 .65];   % roundabout:  gamma for 1 or 2 link lanes
CONSTANTS.LaneChangeUtil = -.95;       % weave model: utility of a lane change
CONSTANTS.TaperUtil = -.17;            % weave model: utility of taper use, or the merging lane
CONSTANTS.WeaveFraction = 0.79;        % weave model: fraction of weaving traffic over the lane
CONSTANTS.WeaveLaneCap = 3791;         % weave model: (very local) lane capacity [pcu/h]
```

```
% CBQ
% deduce congested wave speeds from triangular fundamental diagrams
WaveSpeed = network.LinkCapacity./(CONSTANTS.QueueDensity*network.LinkLanes -...
    network.LinkCapacity./network.LinkSpeed);
% devide the links into cells
network.CongCellLength = (WaveSpeed*CONSTANTS.DeltaK);
network.CellCount = network.LinkLength./network.CongCellLength;
network.CellLastFactor = rem(network.CellCount, 1);
network.CellCount = ceil(network.CellCount);
network.CellLastFactor(network.CellLastFactor==0) = 1; % integer number of cells
CONSTANTS.MaxCells = max(network.CellCount); % least amount of past flow in memory

% Node model
for n = 1:length(conflicts)
    % calculate controlled capacities from ideal capacity fractions
    if strcmp(conflicts(n).type, 'Controlled')
        conflicts(n).node.capacities = conflicts(n).node.capacities*...
            CONSTANTS.ConflictCap*CONSTANTS.DeltaK;
    end
    % initiate previous time step flows
    if strcmp(conflicts(n).type, 'Uncontrolled') || strcmp(conflicts(n).type, 'Roundabout')
        % scalar will be expanded to matrix size, after the first loop the
        % prevTurnFlows will be overwritten with matrices
        conflicts(n).node.prevTurnFlows1 = 0;
        conflicts(n).node.prevTurnFlows2 = 0;
    end
    % calculate alpha/beta/gamma
    if strcmp(conflicts(n).type, 'Roundabout') && ~strcmp(conflicts(n).node.type, 'Turbo')
        % 1-Lane / 2-Lane
        % node level
        if ~isempty(conflicts(n).node.alpha)
            conflicts(n).node.alpha = interp1q(CONSTANTS.RoundaboutAlpha(1,:)', ...
                CONSTANTS.RoundaboutAlpha(2,:)', conflicts(n).node.alpha);
        end
        conflicts(n).node.beta = CONSTANTS.RoundaboutBeta(conflicts(n).node.beta);
        if ~isempty(conflicts(n).node.gamma)
            conflicts(n).node.gamma = CONSTANTS.RoundaboutGamma(conflicts(n).node.gamma);
        end
        % link level
        for i = 1:length(conflicts(n).node.inlinks)
            if ~isempty(conflicts(n).node.inlink(i).alpha)
                conflicts(n).node.inlink(i).alpha = interp1q(CONSTANTS.RoundaboutAlpha(1,:)', ...
                    CONSTANTS.RoundaboutAlpha(2,:)', conflicts(n).node.inlink(i).alpha);
            end
            if ~isempty(conflicts(n).node.inlink(i).gamma)
                conflicts(n).node.inlink(i).gamma = ...
                    CONSTANTS.RoundaboutGamma(conflicts(n).node.inlink(i).gamma);
            end
        end
    elseif strcmp(conflicts(n).type, 'Roundabout')
        % Turbo
        for i = 1:length(conflicts(n).node.inlinks)
            for j = 1:length(conflicts(n).node.inlink(i).lane)
                conflicts(n).node.inlink(i).lane(j).alpha = ...
                    interp1q(CONSTANTS.RoundaboutAlpha(1,:)', ...
                    CONSTANTS.RoundaboutAlpha(2,:)', conflicts(n).node.inlink(i).lane(j).alpha);
                conflicts(n).node.inlink(i).lane(j).beta = ...
                    CONSTANTS.RoundaboutBeta(conflicts(n).node.inlink(i).lane(j).beta);
                conflicts(n).node.inlink(i).lane(j).gamma = 1;
            end
        end
    end
end

% LCM (1st splitter)
for m =1:length(conflicts)
    if isfield(conflicts(m).node, 'inlink')
        if ~isfield(conflicts(m).node.inlink, 'lanemap')
            continue
        end
        for n = 1:length(conflicts(m).node.inlink)
            lanemap = conflicts(m).node.inlink(n).lanemap;
            % remove impossible turns, but remember correct number
            turns = find(sum(lanemap,2)>0);
            map = lanemap(sum(lanemap,2)>0,:);
            % loop and find independant blocks
            laneSplits = 1;
            flowSplits = 1;
            go = true;
            i = 1;
            j = 1;
            while go
                if j+1 <= size(map,2) && map(i,j+1) == 1
                    % dependant with next lane
                    j = j+1;
                elseif i+1 <= size(map,1) && map(i+1,j) == 1
                    % dependant with next flow
                    i = i+1;
                elseif j+1 <= size(map,2) && i+1 <= size(map,1)
                    % independant block found
                    j = j+1;
                    i = i+1;
                    laneSplits = [laneSplits j];
                    flowSplits = [flowSplits turns(i)];
                else
                    % the end
                    go = false;
                end
            end
            % add additional index as end of last block
            laneSplits = [laneSplits size(lanemap,2)+1];
            flowSplits = [flowSplits size(lanemap,1)+1];
            conflicts(m).node.inlink(n).laneSplits = laneSplits;
            conflicts(m).node.inlink(n).flowSplits = flowSplits;
        end
    end
end
```

## Cell Based Queuing

Cell Based Queuing is performed in a separate function. For the interpolation and cell state determination separate functions are used as these are also used elsewhere.

```matlab
function [MaximumInflow QueueInflow PotentialOutflow] = CBQ(network, CONSTANTS, column,
temporary)

% This function performs the Cell Based Queueing modelling framework to
% derive maximum inflow, queue inflow and potential outflow.
%
% NOTE: The performance of this function heavily relies on keeping the
% temporary structure as is. Any changes to it's fields requires a copy in
% memory. Vectors are thus returned that should be implemented into the
% temporary structure in any function that calls this function.

% Vehicles in queue and on the link
LinkLoad = sum(temporary.LinkInflow(:,column,:),3) - ...
    sum(temporary.LinkOutflow(:,column,:),3);
QueueLoad = sum(temporary.QueueInflow(:,column,:),3) - ...
    sum(temporary.LinkOutflow(:,column,:),3);
QueueLoad(QueueLoad<1e-12) = 0; % rounding issues -> very small negative numbers

% Pre-allocate
MaximumInflow = zeros(CONSTANTS.Links, 1);
QueueInflow = zeros(CONSTANTS.Links, 1, CONSTANTS.Classes);
PotentialOutflow = zeros(CONSTANTS.Links, 1, CONSTANTS.Classes);
QueueLength = zeros(CONSTANTS.Links, 1);

%% Loop the links
for n = 1:CONSTANTS.Links

    % Deduce cell states
    if QueueLoad(n) > 0
        [L T S shockEnable] = cellStates(network, column, temporary, CONSTANTS, n, QueueLoad(n));
        % make cumulative
        L = [0 cumsum(L)];
        T = [0 cumsum(T)];
        S = [0 cumsum(S)];
        % interpolate queue length
        QueueLength(n,1) = interpCBQ(QueueLoad(n), S, L);
    else
        % no queue
        shockEnable = false;
    end

    % Queue inflow
    FreeFlowLength = max(network.LinkLength(n)-QueueLength(n,1), 0);
    Period = min(max(column+1-...
        FreeFlowLength./(network.LinkSpeed(n)*CONSTANTS.DeltaK), 1), column);
    % If Period equals column, the free flow section is transversed within
    % a time step and thus QueueInflow will in reality also have vehicles
    % not on the link yet.
    QueueInflow(n,1,1:CONSTANTS.Classes) = temporary.LinkInflow(n,floor(Period),:) + ...
        rem(Period,1) * (temporary.LinkInflow(n,ceil(Period),:) - ...
        temporary.LinkInflow(n,floor(Period),:));

    % Maximum inflow
    if shockEnable
        MaximumInflow(n) = max(min(network.LinkCapacity(n)*CONSTANTS.DeltaK, ...
            sum(S(1,end,:),3)-LinkLoad(n)), 0);
    else
        % for early time steps the storage does not cover all cells
        MaximumInflow(n) = network.LinkCapacity(n)*CONSTANTS.DeltaK;
    end

    % Potential outflow
    if QueueLoad(n) == 0
        % as there is no queue, the queue inflow is equal to the outflow
        PotentialOutflow(n,:,:) = QueueInflow(n,1,:) - ...
            temporary.QueueInflow(n,column,:);
    else
        % assume a large queue
        AggregatedPotentialOutflow = interpCBQ(CONSTANTS.DeltaK, T, S);
        if AggregatedPotentialOutflow < QueueLoad(n)
            % all vehicles are indeed queued, get potential outflows from
            % queue inflow as all classes have the same speed, find time
            % where the total number of vehicles is equal to x
            x = sum(temporary.LinkOutflow(n,column,:),3) + AggregatedPotentialOutflow;
            xArray = sum(temporary.QueueInflow(n,1:column,:),3);
            d = xArray - x;
            k = length(d(d<0));
            f = (x-xArray(k))/(xArray(k+1)-xArray(k)); % fraction of linear step
            PotentialOutflow(n,:) = temporary.QueueInflow(n,k,:) + ...
                f.*(temporary.QueueInflow(n,k+1,:)-temporary.QueueInflow(n,k,:)) - ...
                temporary.LinkOutflow(n,column,:);
        else
            % small queue, add free flow vehicles
            QueueTravelTime = interpCBQ(QueueLoad(n), S, T);
            Tremainder = CONSTANTS.DeltaK-QueueTravelTime;
            Lab = Tremainder.*network.LinkSpeed(n);
            FreeFlowLength = network.LinkLength(n)-(QueueLength(n,1)+Lab);
            % similar to queue inflow but with a different distance and for the
            % current time step as we need the current vehicles within the range
            Period = max(column-FreeFlowLength./(network.LinkSpeed(n)*CONSTANTS.DeltaK), 1);
            PotentialOutflow(n,:,:) = temporary.LinkInflow(n,floor(Period),:) + ...
                rem(Period,1) * (temporary.LinkInflow(n,ceil(Period),:) - ...
                temporary.LinkInflow(n,floor(Period),:)) - temporary.LinkOutflow(n,column,:);
        end
    end

    % Apply hazard destruction
    PotentialOutflow(n,:,:) = min(PotentialOutflow(n,:,:), ...
        permute(temporary.MaxLinkOutflow(n,:), [1 3 2]));

end
```

```
function [L, T, S, shockEnable] = cellStates(network, column, temporary, CONSTANTS, n, QueueLoad)

% deal with early time steps
if column-network.CellCount(n) < 1
    nCells = column-1;
    fLast = 1;
    shockEnable = false; % ignore storage constraint
else
    nCells = network.CellCount(n);
    fLast = network.CellLastFactor(n);
    shockEnable = true;
end

% get flow pattern (in reversed order), sum classes
pastFlow = sum(temporary.LinkOutflow(n,column:-1:column-nCells,:), 3);
cellFlow = pastFlow(1,1:end-1) - pastFlow(1,2:end); % cumulative to momentary
cellFlow = cellFlow./CONSTANTS.DeltaK; % pcu/dt -> pcu/h
% calculate to density and speed via the fundamental diagram
kjam = CONSTANTS.QueueDensity*network.LinkLanes(n);
kcap = network.LinkCapacity(n)/network.LinkSpeed(n);
cellDens = kcap+((network.LinkCapacity(n)-cellFlow)*(kjam-kcap)/network.LinkCapacity(n));
cellSpeed = cellFlow./cellDens;
% apply saturation flow on the first cell
cellSpeed(1) = max(cellFlow(1), network.LinkSaturationFlow(n))/cellDens(1);
cellSpeed(cellSpeed<0) = 0; % rounding errors
% calculate to travel time and storage
L = ones(1,nCells).*network.CongCellLength(n);
L(end) = L(end).*fLast;
T = L./cellSpeed;
S = L.*cellDens;
```

```
function out = interpCBQ(in, cumulIn, cumulOut)
% This function interpolates as interp1q does, but it works faster as it is
% specific to the linear method and it only works for a single 'in' value.
% The edge values are returned for out of range values.
d = cumulIn - in;
k = length(d(d<0));
if k == 0
```

```
    out = cumulOut(1);
elseif k == length(cumulOut);
    out = cumulOut(k);
else
    sx = in - cumulIn(k);
    dx = cumulIn(k+1)-cumulIn(k);
    dy = cumulOut(k+1)-cumulOut(k);
    out = cumulOut(k) + sx*dy/dx;
end
```

## Node model

The node model is programmed in the nodeModel functions. Sub functions of the Lane Choice Model are contained within the same file.

```
function [LinkInflow LinkOutflow conflicts] = nodeModel(network, CONSTANTS, ...
    column, SplitFractions, MaximumInflow, PotentialOutflow, ...
    Departures, evacscheme, conflicts, temporary)

% This function performs the node model to derive actual inflow and
% outflow.
%
% NOTE: The performance of this function heavily relies on keeping the
% temporary structure as is. Any changes to it's fields requires a copy in
% memory. Vectors are thus returned that should be implemented into the
% temporary structure in any function that calls this function.

% Pre-allocate
LinkInflow = zeros(CONSTANTS.Links, 1, CONSTANTS.Classes);
LinkOutflow = zeros(CONSTANTS.Links, 1, CONSTANTS.Classes);
% permute: [outlink, classes, 1] -> [inlink, outlink, classes]
SplitFractions = permute(SplitFractions, [3 1 2]);

% Connector links
for i = 1:length(network.ConnectorsO)
    % get classes from scheme
    if ~isempty(evacscheme)
        classes = find(evacscheme.Origins == network.LinkTail(network.ConnectorsO(i)));
    else
        classes = 1;
    end
    departingVehicles = SplitFractions(1,network.ConnectorsO(i),classes) .* ...
        ((Departures(network.ConnectorsO(i)) / CONSTANTS.Occupation) - ...
        temporary.LinkInflow(network.ConnectorsO(i),column,classes)); % [1 x 1 x classes]
    ratio = min( MaximumInflow(network.ConnectorsO(i))/sum(departingVehicles), 1);
    LinkInflow(network.ConnectorsO(i),1,classes) = ratio * departingVehicles;
end
affected = intersect(network.ConnectorsO,find(temporary.AffectedLinks));
LinkInflow(affected,1,:) = 0;
% destination links have infinite output capacity
LinkOutflow(network.ConnectorsD,1,:) = ...
    temporary.QueueInflow(network.ConnectorsD,column+1,:) - ...
    temporary.QueueInflow(network.ConnectorsD,column,:);

% Other links, loop nodes they connect to
for n = 1:CONSTANTS.Nodes
    if any(network.Origins == n) || any(network.Destinations == n)
        continue
    end
    store = '';
    % Calculate turnflows
    nIn = length(conflicts(n).node.inlinks);
    nOut = length(conflicts(n).node.outlinks);
    turnFlows = zeros(nIn, nOut, CONSTANTS.Classes);
    if nOut == 1
        % Merge
        % no route choice
        turnFlows(:,1,:) = PotentialOutflow(conflicts(n).node.inlinks,1,1:CONSTANTS.Classes);
    else
        % Intersection
        % get turn matrix
        turnmatrix = conflicts(n).node.turnmatrix;
        % loop links and define turnflows
        fractions = zeros(1,1:nOut,1:CONSTANTS.Classes);
        for i = 1:nIn
            % space = [inlink(=i) x outlink x class]
            % get split fractions for link i that are possible
            if isempty(turnmatrix)
                fractions(1,1:nOut,1:CONSTANTS.Classes) = ...
                    SplitFractions(1,conflicts(n).node.outlinks,:);
            else
                fractions(1,1:nOut,1:CONSTANTS.Classes) = ...
                    SplitFractions(1,conflicts(n).node.outlinks,:) .* ...
                    turnmatrix(i,:,ones(1,CONSTANTS.Classes));
                % scale to sum = 1 per class
                if sum(fractions) == 0
                    fractions = zeros(size(fractions));
                else
                    s = sum(fractions,2);
                    fractions = fractions./s(:,ones(1,nOut),:);
                    fractions(isnan(fractions)) = 0; % 0/0 = nan
                end
            end
            inLink = conflicts(n).node.inlinks(i);
            turnFlows(i,:,:) = PotentialOutflow(inLink,ones(1,nOut),...
                1:CONSTANTS.Classes).*fractions;
        end
    end

    % Apply constraints on the nodes
    % this step will adapt the turn flow matrix , 'None' nodes are skipped
    switch conflicts(n).type
        case 'Controlled'
            % get partial flows
            partialFlows = LCM(sum(turnFlows,3), conflicts(n).node);
            maxPartialFlows = max(partialFlows, [], 3); % [inlink x outlink]
            % loop as long as any constraint is violated
            nConfs = size(conflicts(n).node.conflicts, 3);
            maxRelLoad = inf;
            distrib = [];
            while maxRelLoad > 1
                % calculate conflict demand
                demand = sum(sum( maxPartialFlows(:,:,ones(1,nConfs)) .* ...
                    conflicts(n).node.conflicts, 1), 2);
                % calculate reduction factor
                [maxRelLoad inds] = max(demand./conflicts(n).node.capacities);
                reduction = min(1, 1/maxRelLoad);
                % find inlinks of these conflicts (usually only 1 conflict)
                inLinks = sum(sum(conflicts(n).node.conflicts(:,:,inds), 3), 2) > 0;
                % calculate link time allocation
                distrib(inLinks,end+1) = sum(maxPartialFlows(inLinks,:) .* ...
                    conflicts(n).node.conflicts(inLinks,:,inds(1)),2) / ...
                    demand(inds(1));
                % change all flows from these inlinks
                turnFlows(inLinks,:,:) = reduction*turnFlows(inLinks,:,:);
                maxPartialFlows(inLinks,:) = reduction*maxPartialFlows(inLinks,:);
            end
            % apply reduction by sum of maximum link times
            reduction = min(1, 1/sum(max(distrib,[],2)));
            turnFlows = reduction*turnFlows;

        case 'Uncontrolled'
            % get partial flows
            partialFlows = LCM(sum(turnFlows,3), conflicts(n).node);
            % get turn flows of t-1.5
            prevTurnFlows = .5*conflicts(n).node.prevTurnFlows1 + ...
                .5*conflicts(n).node.prevTurnFlows2;
            % apply reduction per link
            for i = 1:nIn
                reduction = 1;
                % loop lanes
                for j = 1:length(conflicts(n).node.inlink(i).lane)
                    % get minor and major groups
                    minor = conflicts(n).node.inlink(i).lane(j).minor;
                    major = conflicts(n).node.inlink(i).lane(j).major;
                    % loop conflict groups
                    for c = 1:size(minor,3)
```

```
                        % calculate flows
                        majorFlow = sum(sum(major(:,:,c).*prevTurnFlows)) / ...
                            CONSTANTS.DeltaK; %[pcu/h]
                        minorFlow = sum(sum(minor(1,:,c).*partialFlows(i,:,j))) / ...
                            CONSTANTS.DeltaK; %[pcu/h]
                        % calculate reduction
                        capacity = exp(-majorFlow*CONSTANTS.MinGapAcceptance) / ...
                            CONSTANTS.AverageHeadway; %[pcu/h]
                        reduction = min(reduction, capacity/minorFlow);
                    end
                end
                % reduce flows from the link
                turnFlows(i,:,:) = reduction*turnFlows(i,:,:);
            end
            % store for next time step
            store = 'turnflows';

        case 'Roundabout'
            % get turn flows of t-1.5
            prevTurnFlows = .5*conflicts(n).node.prevTurnFlows1 + ...
                .5*conflicts(n).node.prevTurnFlows2;
            % model is type specific
            if ~strcmp(conflicts(n).node.type, 'Turbo')
                % -- 1-Lane / 2-Lane --
                % apply reduction per link
                for i = 1:nIn
                    % calculate flows
                    Vexit = sum(sum(conflicts(n).node.inlink(i).Vexit.*prevTurnFlows))...
                        / CONSTANTS.DeltaK; %[pcu/h]
                    Vcirc = sum(sum(conflicts(n).node.inlink(i).Vcirc.*prevTurnFlows))...
                        / CONSTANTS.DeltaK; %[pcu/h]
                    % get parameters from link, or node if not present
                    alpha = conflicts(n).node.inlink(i).alpha;
                    if isempty(alpha)
                        alpha = conflicts(n).node.alpha;
                    end
                    beta = conflicts(n).node.beta; % type specific
                    gamma = conflicts(n).node.inlink(i).gamma;
                    if isempty(gamma)
                        gamma = conflicts(n).node.gamma;
                    end
                    % calculate and apply reduction
                    capacity = (1500 - (8/9)*(alpha*Vexit + beta*Vcirc))/gamma;
                    capacity = max(capacity, 0); % equilibrium/oscilation
                    demand = sum(sum(turnFlows(i,:,:), 3), 2) / CONSTANTS.DeltaK;
                    reduction = min(1, capacity/demand);
                    turnFlows(i,:,:) = reduction*turnFlows(i,:,:);
                end
                % store for next time step
                store = 'turnflows';
            else
                % -- Turbo --
                % get partial flows
                partialFlows = LCM(sum(turnFlows,3), conflicts(n).node);
                % apply reduction per link
                for i = 1:nIn
                    reduction = 1;
                    % determine reduction by critical lane
                    for j = 1:length(conflicts(n).node.inlink(i).lane)
                        % calculate flows
                        cmax = min(size(prevTurnFlows,3), ...
                            size(conflicts(n).node.inlink(i).lane(j).Vexit,3));
                        Vexit = sum(sum(sum( conflicts(n).node.inlink(i).lane(j).Vexit.* ...
                            prevTurnFlows(:,:,1:cmax) ))) / CONSTANTS.DeltaK; %[pcu/h]
                        Vcirc = sum(sum(sum(conflicts(n).node.inlink(i).lane(j).Vcirc.* ...
                            prevTurnFlows(:,:,1:cmax) ))) / CONSTANTS.DeltaK; %[pcu/h]
                        % get parameters from lane
                        alpha = conflicts(n).node.inlink(i).lane(j).alpha;
                        beta  = conflicts(n).node.inlink(i).lane(j).beta;
                        gamma = conflicts(n).node.inlink(i).lane(j).gamma;
                        % calculate reduction
                        capacity = (1500 - (8/9)*(alpha*Vexit + beta*Vcirc))/gamma;
                        capacity = max(capacity, 0); % may occur due to V from t-1
                        demand = sum(partialFlows(i,:,j)) / CONSTANTS.DeltaK; %[pcu/h]
                        reduction = min(reduction, capacity/demand);
                    end
                    % apply reduction
                    turnFlows(i,:,:) = reduction.*turnFlows(i,:,:);
                    partialFlows(i,:,:) = reduction.*partialFlows(i,:,:);
                end
                % store for next time step
                store = 'partialflows';
            end

        case 'Weaving'
            % gather link info
            pre = conflicts(n).node.pre;
            post = conflicts(n).node.post;
            A = pre(1);
            if pre(end) && pre(end) ~= A
                B = pre(end);
            elseif pre(end-1) ~= A
                B = pre(end-1);
            else
                B = []; % diverge section
            end
            C = post(1);
            if post(end) && post(end) ~= C
                D = post(end);
            elseif post(end-1) ~= C
                D = post(end-1);
            else
                D = []; % merge section
            end
            % calculate demand
            demand = [sum(turnFlows(A,C),3) sum(turnFlows(A,D),3); ...
                sum(turnFlows(B,C),3) sum(turnFlows(B,D),3)];
            % prepare taper lane number adjustment
            preTaperAdj = zeros(size(pre));
            if conflicts(n).node.pretaper
                preTaperAdj(pre==C) = -1;
            end
            postTaperAdj = zeros(size(post));
            if conflicts(n).node.posttaper
                postTaperAdj(post==D) = -1;
            end
            % create utility matrix, start with lane changes
            nlanes = length(pre);
            util = zeros(nlanes);
            for i = 1:nlanes
                for j = 1:nlanes
                    util(i,j) = abs((i+preTaperAdj(i))-(j+postTaperAdj(j))) * ...
                        CONSTANTS.LaneChangeUtil;
                end
            end
            % include taper disutility
            if conflicts(n).node.pretaper
                % deduce first and last lanes of links
                Alast = find(pre==A,1,'last');
                Bfirst = find(pre==B,1,'first');
                util(Alast:Bfirst,:) = util(Alast:Bfirst,:) + CONSTANTS.TaperUtil;
            end
            % exclude lanes that appear/disappear
            util(pre==0,:) = -inf;
            util(:,post==0) = -inf;
            % apply logit
            exps = exp(util);
            flow = zeros(nlanes);
            flow(pre==A,post==C) = demand(1,1) .* exps(pre==A,post==C) ./ ...
                sum(sum(exps(pre==A,post==C)));
            if ~isempty(D)
                flow(pre==A,post==D) = demand(1,2) .* exps(pre==A,post==D) ./ ...
                    sum(sum(exps(pre==A,post==D)));
            end
            if ~isempty(B)
                flow(pre==B,post==C) = demand(2,1) .* exps(pre==B,post==C) ./ ...
                    sum(sum(exps(pre==B,post==C)));
            end
            if ~isempty(B) && ~isempty(D)
                flow(pre==B,post==D) = demand(2,2) .* exps(pre==B,post==D) ./ ...
```

```
                    sum(sum(exps(pre==B,post==D)));
            end
            % calculate lane demand
            laneDemand = zeros(nlanes,1);
            if ~isempty(D) && ~isempty(B) % merges have no weaving traffic
                for i = 2:nlanes-1 % exterior lanes cannot have flow over it
                    from_left = (1:nlanes)' + preTaperAdj < i & pre==A;
                    to_right = (1:nlanes)' + postTaperAdj > i & post==D;
                    from_right = (1:nlanes)' + preTaperAdj > i & pre==B;
                    to_left = (1:nlanes)' + postTaperAdj < i & post==C;
                    laneDemand(i,1) = CONSTANTS.WeaveFraction * ...
                        (sum(sum(flow(from_left,to_right))) + ...
                        sum(sum(flow(from_right,to_left))));
                end
            end
            % merge taper lanes
            if conflicts(n).node.pretaper
                flow(Alast,:) = flow(Alast,:) + flow(Bfirst,:);
                laneDemand(Alast,1) = laneDemand(Alast,1) + laneDemand(Bfirst,1);
                flow(Bfirst,:) = [];
                laneDemand(Bfirst,:) = [];
            end
            % calculate reduction
            reduction = min(min(CONSTANTS.WeaveLaneCap*CONSTANTS.DeltaK./ ...
                (sum(flow,2)+laneDemand)), 1);
            % reduce all flows
            turnFlows = reduction*turnFlows;
    end

    % Apply constraints on the outlinks (MaximumInflow)
    if ~isempty(conflicts(n).node.outlinks)
        PotentialInflow = sum(sum(turnFlows,3),1);
        reduction = max(min(min(MaximumInflow(conflicts(n).node.outlinks)'./...
            PotentialInflow), 1), 0);
        turnFlows = reduction*turnFlows;
        % Add to link inflow
        % permute: [1 (sum inlinks), outlink, classes] -> [outlink, 1 (time slice), classes]
        % (transpose respecting 3rd dimension)
        LinkInflow(conflicts(n).node.outlinks,1,:) = permute(sum(turnFlows, 1), [2 1 3]);
        if strcmp(store, 'partialflows')
            partialFlows = reduction*partialFlows;
        end
    end
    if ~isempty(conflicts(n).node.inlinks)
        % Add to link inflow
        % no permute: [outlink, 1 (time slice), classes]
        LinkOutflow(conflicts(n).node.inlinks,1,:) = sum(turnFlows, 2);
    end

    % Apply restrictions as vehicles have been destroyed by the hazard
    LinkOutflow(temporary.AffectedLinks,1,:) = min( LinkOutflow(temporary.AffectedLinks,1,:),...
        permute(temporary.MaxLinkOutflow(temporary.AffectedLinks,:), [1 3 2]) - ...
        temporary.LinkOutflow(temporary.AffectedLinks,column,:) );

    % Store for next time step
    if strcmp(store, 'turnflows')
        conflicts(n).node.prevTurnFlows2 = conflicts(n).node.prevTurnFlows1;
        conflicts(n).node.prevTurnFlows1 = sum(turnFlows,3);
    elseif strcmp(store, 'partialflows')
        conflicts(n).node.prevTurnFlows2 = conflicts(n).node.prevTurnFlows1;
        conflicts(n).node.prevTurnFlows1 = partialFlows;
    end

end

% Deduce partial flows from lanemaps and turn flows
function partialFlows = LCM(turnFlows, node)
% This function actually performs the 2nd splitter and the assignment. The
% first splitter is allraedy performed. The dimensions of the partial flows
% will be: [inlink, outlink, lane]
turnFlows(turnFlows<0) = 0; % tiny rounding  errors around 0
partialFlows = zeros(length(node.inlinks), length(node.outlinks), 0);
% Loop the inlinks of the node
for i = 1:length(node.inlink)
    flows = [];
    lanemap = node.inlink(i).lanemap;
    for s = 1:length(node.inlink(i).laneSplits)-1
        l1 = node.inlink(i).laneSplits(s);
        l2 = node.inlink(i).laneSplits(s+1)-1;
        f1 = node.inlink(i).flowSplits(s);
        f2 = node.inlink(i).flowSplits(s+1)-1;
        % get flows from this link [outlink x lane];
        flows(f1:f2,l1:l2) = zeroSplitter(lanemap(f1:f2,l1:l2), ...
            turnFlows(i,node.inlink(i).order(f1:f2))', zeros(f2-f1+1,l2-l1+1));
    end
    % permute: [outlink, lane] -> [1 (inlink), outlink, lane]
    partialFlows(i,node.inlink(i).order,1:size(flows,2)) = ...
        permute(flows, [3 1 2]);
end

% Split if lane not used
function flows = zeroSplitter(map, dirflows, flows)
% one lane means no choice
if size(map,2) == 1
    flows = dirflows;
    return
end
% assign flows regardless of the possiblitity of negative flows
tempflows = assign(map, dirflows, flows);
% double precision may result in -1e16 while should be zero
[i, j] = find(tempflows==min(min(tempflows)) & tempflows<-1e-15, 1);
if ~isempty(i)
    % negative flows found, split at most negative
    if j+1 <= size(map,2) && map(i,j+1) == 1
        % (last turn flow i on lane j) < 0, next lane is next group
        j = j+1;
    else
        % (last lane j of turn flow i) < 0, next flow is next group
        i = i+1;
    end
    % re-assign before part
    flows(1:i-1,1:j-1) = zeroSplitter(map(1:i-1,1:j-1), dirflows(1:i-1), flows(1:i-1,1:j-1));
    % re-assign after part
    flows(i:end,j:end) = zeroSplitter(map(i:end,j:end), dirflows(i:end), flows(i:end,j:end));
else
    % assigned flows are all positive
    flows = tempflows;
end

% Assigns flows to the lanes
function flows = assign(map, dirflows, flows)
% average flow per lane (is actual flow per lane if dependency holds)
avelaneflow = sum(dirflows)./size(map,2);
i = find(map(:,1),1,'first'); % there may be impossible turn flows
j = 1;
% deduce first elements
if size(map,1) > i && map(i+1,1) == 1
    % first entry is the first flow as the first flow can only use one lane
    flows(i,j) = dirflows(i);
    i = i+1;
elseif size(map,2) > 1 && map(i,j+1) == 1
    % first entry is aveflow as the first lane has only 1 flow
    flows(i,j) = avelaneflow;
    j = j+1;
end
% add extra row and column to the map to avoid range checks
map(end+1,end+1) = 0;
% walk through the map
while i < size(map,1) && j < size(map,2)
    if map(i,j+1) == 0 && map(i+1,j) == 0
        % we have reached the extra row or column, we can either assign the
        % directional flow remainder or average lane flow remainder (is
        % equal)
        flows(i,j) = avelaneflow - sum(flows(1:i,j));
        i = i+1;
        j = j+1;
    elseif map(i,j+1) == 1
        % next step is a new lane, this flow gets average lane flow remainder
        flows(i,j) = avelaneflow - sum(flows(1:i,j));
        j = j+1;
    else
        % next step is a new flow, this lane gets directional flow remainder
        flows(i,j) = dirflows(i) - sum(flows(i,1:j));
        i = i+1;
    end
end
```

## Node Input Generator

The Node Input Generator is in total 2182 lines of code. Much of this code is related to graphical elements and is not related to the actual generation of groups, which is the core of the program. The conflict group generation sub function (genGroups) will be shown, together with the used sub functions.

```
function genGroups(src, evt)
% apply lane maps to turn matrix
fig = gcbf;
udat = get(fig, 'UserData');
type = get(findobj(fig, 'Tag', 'nodeType'), 'UserData');
if (strcmp(type, 'Controlled') || strcmp(type, 'Uncontrolled or priority')) && ...
        get(findobj(fig, 'Tag', 'uselanemaps'), 'Value') == 1
    [udat ok] = lanemaps2turnmatrix(udat);
    if ~ok
        return
    end
end
% empty lanemap if all movements are possible
if get(findobj(fig, 'Tag', 'allmatrix'), 'Value') == 1
    udat.node.turnmatrix = [];
end
% based on the type, generate groups (and maybe some parameters)
switch type
    case {'None', 'Weaving section'}
        % no groups
    case 'Controlled'
        % gather intersecting turnflows
        Uturn = false; % exclude U-turns (permitted conflict)
        crosses = getCrosses(udat, Uturn);
        % initiate conflicts
        conflicts = zeros(length(udat.inlink), length(udat.outlink), 0);
        foundany = true; % while loop
        prev_groups = crosses; % previous loop groups, start with 2-conflicts
        % waitbar tracking
        h = waitbar(0,'');
        g = 2; % group size
        hc = 0; % waitbar update counter
        while foundany
            % update waitbar
            g = g+1;
            waitbar(0,h,['Creating ' num2str(g) '-phase groups']);
            % initialize loop
            foundany = false; % get out of while loop unless it is changed
            cur_groups = zeros(length(udat.inlink), length(udat.outlink), 0);
            had = false(size(prev_groups,3),1); % had groups = subgroups
            % Loop previous groups
            for i = 1:size(prev_groups,3)
                % update waitbar, works per group size from 0 to 1
                hc = hc+1;
                if hc == 20
                    waitbar(i/size(prev_groups,1),h)
                    hc = 0;
                end
                % Per previous group, loop previous groups further along
                for j = i+1:size(prev_groups,3)
                    % get the non-matching turn flows
                    group1 = prev_groups(:,:,i);
                    group2 = prev_groups(:,:,j);
                    tf1 = (group1-group1.*group2);
                    tf2 = (group2-group1.*group2);
                    if sum(sum(tf1))~=1 || sum(sum(tf2))~=1
```

```
                        % per group 1 should not match, skip otherwise
                        continue
                    end
                    % is this combination a conflict?
                    this = getConflict(tf1|tf2, crosses);
                    if any(this)
                        foundany = true;
                        % find sub groups
                        these = getConflict(group1|group2, prev_groups);
                        had(these) = true;
                        % find duplicates
                        these = getConflict(group1|group2, cur_groups);
                        if ~any(these)
                            % only add if not present allready
                            cur_groups(:,:,end+1) = group1|group2;
                        end
                    end
                end
            end
            % if any prev_group is not part of a current group, keep it
            if any(~had)
                conflicts = cat(3, conflicts, prev_groups(:,:,~had));
            end
            % prepare for next loop
            prev_groups = cur_groups;
        end
        % capacities, determine right turns
        waitbar(1,h,'Determining capacities');
        rightTurns = zeros(length(udat.inlink), length(udat.outlink));
        for i = 1:length(udat.inlink)
            minang = inf;
            lanemap = udat.inlink(i).lanemap;
            % find outlink closest to the right
            for j = 1:length(udat.outlink)
                if udat.inlink(i).angle == udat.outlink(j).angle
                    continue
                end
                ang = 360-innerAngle(udat.inlink(i).angle, udat.outlink(j).angle);
                if ang < minang;
                    minang = ang;
                    right = j;
                end
            end
            % search for inlink from the right that is closer
            for j = 1:length(udat.inlink)
                if udat.inlink(i).angle == udat.inlink(j).angle
                    continue
                end
                ang = 360-innerAngle(udat.inlink(i).angle, udat.inlink(j).angle);
                if ang < minang
                    % no right turn as inlink is closer
                    right = [];
                end
            end
            % does the right turn have dedicated lanes only?
            if any(sum(lanemap(:,lanemap(end,:)>0),1) > 1)
                % their is a shared lane
                right = [];
            end
            % store dedicated right turns
            if ~isempty(right)
                rightTurns(i,right) = 1;
            end
        end
        maxConf = max(sum(sum(conflicts,1),2));
        capacities = ones(1,1,size(conflicts,3));
        for c = 1:size(conflicts,3)
            if sum(sum(conflicts(:,:,c) .* rightTurns)) > 0 && ...
                    sum(sum(conflicts(:,:,c),1),2) < maxConf;
                % no right turn, this conflict can only use a fraction of time
                capacities(1,1,c) = (maxConf + sum(sum(conflicts(:,:,c),1),2)) / (2*maxConf);
            end
        end
        if ishandle(h)
            delete(h) % delete waitbar
        end
        % store conflicts
        udat.node.conflicts = conflicts;
        udat.node.capacities = capacities;
        msgbox([num2str(size(conflicts, 3)) ' conflict groups generated.'])
    case 'Uncontrolled or priority'
        % gather intersecting turnflows
        Uturn = true; % include U-turns
        crosses = getCrosses(udat, Uturn);
        % create waitbar
        h = waitbar(0,'');
        % group counter (information for user only)
        ngroups = 0;
        % Loop the inlinks
        for i1 = 1:length(udat.inlink)
            % update waitbar
            waitbar(i1/length(udat.inlink), h, ['Creating minor/major groups for link '...
                num2str(i1) ' of ' num2str(length(udat.inlink))])
            % we need a lanemap
            if isempty(udat.inlink(i1).lanemap)
                errordlg(['Link ' num2str(i1) ' does not have a lanemap.'], '')
                delete(h)
                return
            end
            % find major flows of all turn flows from i1
            for j1 = 1:length(udat.outlink)
                % initiate majors matrix of flows towards j1
                majors(:,:,j1) = zeros(length(udat.inlink), length(udat.outlink));
                % loop inlinks to find major flows
                for i2 = 1:length(udat.inlink)
                    if i2 == i1
                        continue % major flow not from the same link
                    end
                    % loop outlinks to find major flows
                    for j2 = 1:length(udat.outlink)
                        % i1-j1 conflicting with i2-j2?
                        comb = zeros(length(udat.inlink),length(udat.outlink));
                        comb(i1, j1) = 1;
                        comb(i2, j2) = 1;
                        this = getConflict(comb, crosses);
                        if ~isempty(find(this,1))
                            if udat.inlink(i1).priority && ~udat.inlink(i2).priority
                                % i2-j2 is not major
                            elseif ~udat.inlink(i1).priority && udat.inlink(i2).priority
                                % i2-j2 is major
                                majors(i2,j2,j1) = 1; % *)
                            else
                                % both priority or both not, major if from the right
                                ai1 = udat.inlink(i1).angle;
                                aj1 = udat.outlink(j1).angle;
                                ai2 = udat.inlink(i2).angle;
                                ai1_j1 = innerAngle(aj1, ai1); % angle between in1 and out1
                                if ai1_j1 == 0
                                    ai1_j1 = 360;
                                end
                                ai1_i2 = innerAngle(ai2, ai1); % angle between in1 and in2
                                if ai1_i2 < ai1_j1 % in2 comes from the right
                                    % i2-j2 is major
                                    majors(i2,j2,j1) = 1; % *)
                                end
                            end
                            % *) i2-j2 is a major turn flow for flow from
                            % the current link i1 towards j1
                        end
                    end
                end
            end
            % loop lanes at link i1
            for i = 1:size(udat.inlink(i1).lanemap, 2)
                % initiate minor and major for this lane
                minor = zeros(1,length(udat.outlink),0);
                major = zeros(length(udat.inlink),length(udat.outlink),0);
                % get available turns from the turn lane
                turns = find(udat.inlink(i1).lanemap(:,i));
                order = udat.inlink(i1).order;
                % find groups of size j = 1, 2, 3 etc. partial flows
                for j = 1:length(turns)
                    % get all combinations of size j
                    group_minors = nchoosek(turns,j);
                    % loop partial groups and find common majors
                    for g = 1:size(group_minors,1)
                        % assume all majors
                        group_majors = ones(length(udat.inlink), length(udat.outlink));
                        % loop minors in group
                        for p = 1:size(group_minors,2)
                            % major only if major for all
                            these_majors = majors(:,:,order(group_minors(g,p)));
                            group_majors = group_majors & these_majors;
                        end
                        % keep non-empty groups
                        if sum(sum(group_majors)) > 0
                            these_minors = zeros(1,length(udat.outlink),1);
                            these_minors(order(group_minors(g,:))) = 1;
                            minor = cat(3, minor, these_minors);
                            major = cat(3, major, group_majors);
                            ngroups = ngroups+1;
                        end
                    end
                end
                % delete groups that are a subset of others
                these = true(1, size(minor, 3));
                for j = 1:size(minor, 3)
                    others = 1:size(minor, 3);
                    others(j) = [];
                    ind1 = getConflict(minor(:,:,j), minor(:,:,others));
                    ind2 = getConflict(major(:,:,j), major(:,:,others));
                    these(ind1&ind2) = false;
                end
                minor = minor(:,:,these);
                major = major(:,:,these);
                % store groups per lane
                udat.inlink(i1).lane(i).minor = minor;
                udat.inlink(i1).lane(i).major = major;
            end
        end
        delete(h)
        msgbox([num2str(ngroups) ' conflict groups generated.'])
    case 'Roundabout'
        % get roundabout type
        rtype = get(findobj(fig, 'Tag', 'rounType'), 'UserData');
        switch rtype
            case {'1-Lane', '2-Lane'}
                % Loop inlink to generate parameters for
                for i1 = 1:length(udat.inlink)
                    % initiate Vexit
                    Vexit = zeros(length(udat.inlink), length(udat.outlink));
                    % find nearest outlink (to the left)
                    i1a = udat.inlink(i1).angle;
                    minang = inf;
                    % loop outlinks to find nearest
                    for j = 1:length(udat.outlink)
                        ja = udat.outlink(j).angle;
                        ang = innerAngle(i1a, ja);
                        if ang < minang
                            minang = ang;
                            exit = j;
                        end
                    end
                    % loop inklinks to see if any is actually closer
                    for j = 1:length(udat.inlink)
                        if i1 == j
                            continue % not the same link
                        end
                        ja = udat.inlink(j).angle;
                        ang = innerAngle(i1a, ja);
                        if ang < minang
                            exit = []; % no exit flow is closest link is an inlink
                        end
                    end
                    % all flow to the exit link is exit flow
                    if ~isempty(exit)
                        Vexit(:,exit) = 1;
                    end
                    % initiate Vcirc
                    Vcirc = zeros(length(udat.inlink), length(udat.outlink));
                    % loop inlinks to get turnflows
                    for i2 = 1:length(udat.inlink)
                        i2a = udat.inlink(i2).angle;
                        % loop outlinks to get turnflows
                        for j2 = 1:length(udat.outlink)
                            j2a = udat.outlink(j2).angle;
                            ang1 = 360-innerAngle(i2a, i1a); % angle to the right
                            ang2 = 360-innerAngle(i2a, j2a); % angle to the right
                            if ang1 < ang2
                                % turnflow goes past inlink
                                Vcirc(i2, j2) = 1;
                            end
                        end
                    end
                    % store per link
                    udat.inlink(i1).Vexit = Vexit;
                    udat.inlink(i1).Vcirc = Vcirc;
                end
                msgbox('Vexit and Vcirc generated for all inlinks.', '')
            case 'Turbo'
                % Get all nodes and links from the plot
                ax = findobj(fig, 'Tag', 'rounTurboAxes');
                allLines = findobj(ax, 'Type', 'line');
                m = get(allLines, 'Marker');
                % get links (without marker)
                linkObjs = allLines(strcmp(m, 'none'));
                if length(linkObjs) <= 1
                    msgbox('Please draw the roundabout first.')
                    return
                end
                % get nodes by the tag
                originObjs = findobj(ax, '-regexp', 'Tag', 'origin');
                destinationObjs = findobj(ax, '-regexp', 'Tag', 'destination');
                roundaboutObjs = findobj(ax, '-regexp', 'Tag', 'roundabout');
                % store links with the XY coordinates
                objs.links = [cell2mat(get(linkObjs, 'XData')),...
                    cell2mat(get(linkObjs, 'YData'))];
                % set right order (against the clock)
                for l = 1:size(objs.links,1)
                    x1 = objs.links(l,1);
                    x2 = objs.links(l,2);
                    y1 = objs.links(l,3);
                    y2 = objs.links(l,4);
                    ang1 = getAng(x1, y1);
                    ang2 = getAng(x2, y2);
                    ang = 360-innerAngle(ang1, ang2); % angle to the right
                    if ang > 180
                        % no link can span more than 180 degrees, must be
                        % the other way around
                        objs.links(l, :) = [x2 y2 x1 y1];
                    else
                        objs.links(l, :) = [x1 y1 x2 y2];
                    end
                    set(linkObjs(l), 'Color', [1 0 0])
                    p = plot(objs.links(l, 1), objs.links(l, 2), 'Marker', '.');
                    delete(p)
                    set(linkObjs(l), 'Color', [0 0 0])
                end
                % store nodes with the XY coordinates
                objs.origins = [cell2mat(get(originObjs, 'XData')),...
                    cell2mat(get(originObjs, 'YData'))];
                objs.destinations = [cell2mat(get(destinationObjs, 'XData')),...
                    cell2mat(get(destinationObjs, 'YData'))];
                objs.roundabouts = [cell2mat(get(roundaboutObjs, 'XData')),...
                    cell2mat(get(roundaboutObjs, 'YData'))];
                % Loop inlinks
                h = waitbar(0, '');
                for i = 1:length(udat.inlink)
                    % update waitbar
                    waitbar(i/length(udat.inlink), h, ['Generating lanemap, ',...
                        'Beta''s, Vexit''s & Vcirc''s for link ' num2str(i)])
                    % Loop lanes of link i
                    nlanes = length(findobj(originObjs, 'Tag', ['origin' num2str(i)]));
                    % initiate lanemap
                    lanemap = zeros(length(udat.outlink), nlanes);
                    for l = 1:nlanes
                        % find origin node
                        for n = 1:length(originObjs)
                            % link and lane number in the userdata
                            dat = get(originObjs(n), 'UserData');
                            if dat.inlink == i && dat.lane == l
                                % node found, n stays at current value
                                break
                            end
                        end
                        % == lanemap ==
                        % move downstream
                        [origins, destinations, roundabouts] = ...
                            getNodes('down', objs, objs.origins(n,:), []);
                        if isempty(roundabouts)
                            % origin not connected properly
                            errordlg('An origin node is not connected to the roundabout')
                            delete(h)
                            return
                        end
                        % get stop nodes at this crossection
                        % find onject
                        obj = findobj(roundaboutObjs, 'XData', roundabouts(1,1),...
                            'YData', roundabouts(1,2));
                        % find objects with the same tag
                        groupobjs = findobj(roundaboutObjs, 'Tag', get(obj, 'Tag'));
                        % get XY coordinates
                        if length(groupobjs) == 1
                            stopnodes = [get(groupobjs, 'XData') get(groupobjs, 'YData')];
                        else
                            stopnodes = [cell2mat(get(groupobjs, 'XData')),...
                                cell2mat(get(groupobjs, 'YData'))];
                        end
                        % Travel downstream in a loop
                        % lanemap is from left to right, links will be found
                        % in opposite order, start counter at the right
                        n_out = length(udat.outlink)+1;
                        ok = true; % keep on looping
                        while ok
                            % Find exit link that may not be accessible to
                            % update exit link counter
                            % find object of first roundabout node
                            obj = findobj(roundaboutObjs, 'XData', roundabouts(1,1),...
                                'YData', roundabouts(1,2));
                            % find all objects in this set
                            groupobjs = findobj(roundaboutObjs, 'Tag', get(obj, 'Tag'));
                            % get XY coordinates
                            if length(groupobjs) == 1
                                r_nodes = [get(groupobjs, 'XData') get(groupobjs, 'YData')];
                            else
                                r_nodes = [cell2mat(get(groupobjs, 'XData')),...
                                    cell2mat(get(groupobjs, 'YData'))];
                            end
                            % move downstream from this set
                            [origins2, destinations2, roundabouts2] = ...
                                getNodes('down', objs, r_nodes, []);
                            if ~isempty(destinations2)
                                n_out = n_out-1; % exit link is found, adapt lane counter
                            end
                            % Find actual downstream nodes
                            [origins, destinations, roundabouts] = ...
                                getNodes('down', objs, roundabouts, stopnodes);
                            if ~isempty(destinations)
                                % exit accessible from lane l
                                lanemap(n_out,l) = 1;
                            end
                            % stop loop as dead ends (stop nodes) are found
                            if isempty(roundabouts)
                                ok = false;
                            end
                        end
                        % create order array
                        order = 1:length(udat.outlink);
                        order = [order(i:end) order(1:i-1)]; % shift
                        % == Vcirc ==
                        % get downstream nodes
                        [origins, destinations, roundabouts] = ...
                            getNodes('down', objs, objs.origins(n,:), []);
                        % find objects in this set
                        obj = findobj(roundaboutObjs, 'XData', roundabouts(1,1),...
                            'YData', roundabouts(1,2));
                        groupobjs = findobj(roundaboutObjs, 'Tag', get(obj, 'Tag'));
                        % get XY coordinates as stopnodes
                        if length(groupobjs) == 1
                            stopnodes = [get(groupobjs, 'XData') get(groupobjs, 'YData')];
                        else
                            stopnodes = [cell2mat(get(groupobjs, 'XData')),...
                                cell2mat(get(groupobjs, 'YData'))];
                        end
                        % find all right-hand nodes as these also cross
                        r_all = sqrt(stopnodes(:,1).^2+stopnodes(:,2).^2);
                        r_linked = min(sqrt(roundabouts(:,1).^2+roundabouts(:,2).^2));
                        roundabouts = stopnodes(r_linked<=r_all,:);
                        % store these nodes as they are also needed for Vexit
                        rgroup = roundabouts;
                        % travel upstream 1 roundabout set
                        [origins, destinations, roundabouts] = ...
                            getNodes('up', objs, roundabouts, []);
                        % the number of roundabout lanes here determines beta
                        beta = size(roundabouts,1);
                        % initiate Vcirc
                        Vcirc = zeros(length(udat.inlink),length(udat.outlink),0);
                        % keep track of available outlinks
                        out_links = true(1,length(udat.outlink));
                        % travel further upstream
                        ok = true;
                        while ok
                            % move upstream
                            [origins, destinations, roundabouts] = ...
                                getNodes('up', objs, roundabouts, stopnodes);
                            % add these origin nodes to partial flows
                            if ~isempty(origins)
                                for o = 1:size(origins,1)
                                    % find object and its data
                                    obj = findobj(originObjs, 'XData', origins(o,1),...
                                        'YData', origins(o,2));
                                    dat = get(obj, 'UserData');
                                    % out_links may include impossible
                                    % turns from dat.inlink, these flows
                                    % are zero anyway
                                    Vcirc(dat.inlink, out_links, dat.lane) = 1;
                                end
                            end
                            if ~isempty(roundabouts)
                                % move downstream and find destinations,
                                % these may not be destinations of any
                                % partial flow to be found later as flow
                                % does not pass the current link
                                [origins2, destinations2, roundabouts2] = ...
                                    getNodes('down', objs, roundabouts, []);
                                if ~isempty(destinations2)
                                    % find object and its data
                                    obj = findobj(destinationObjs, 'XData', ...
                                        destinations2(1,1), 'YData', destinations2(1,2));
                                    dat = get(obj, 'UserData');
                                    out_links(dat.outlink) = false;
                                end
                            else
                                % stop if no roundabout nodes anymore
                                ok = false;
                            end
                        end
                        % store per lane
                        udat.inlink(i).lane(l).Vcirc = Vcirc;
                        % == Vexit ==
                        % data from Vcirc can be used
                        roundabouts = rgroup;
                        % move upstream
                        [origins, destinations, roundabouts] = ...
                            getNodes('up', objs, roundabouts, []);
                        % find set as stopnodes
                        if ~isempty(roundabouts)
                            obj = findobj(roundaboutObjs, 'XData', roundabouts(1,1),...
                                'YData', roundabouts(1,2));
                            groupobjs = findobj(roundaboutObjs, 'Tag', get(obj, 'Tag'));
                            if length(groupobjs) == 1
                                stopnodes = [get(groupobjs, 'XData'),...
                                    get(groupobjs, 'YData')];
                            else
                                stopnodes = [cell2mat(get(groupobjs, 'XData')),...
                                    cell2mat(get(groupobjs, 'YData'))];
                            end
                        else
                            % roundabout apparently not a full circle, thus
                            % no need for stopnodes
                            stopnodes = [];
                        end
                        % move upstream
                        [origins, destinations, roundabouts] = ...
                            getNodes('up', objs, roundabouts, []);
                        % move downstream once to find exit nodes
                        [origins, destinations, roundabouts] = ...
                            getNodes('down', objs, roundabouts, []);
                        % get exit link number through the node object
                        out_link = false(1,length(udat.outlink));
                        if ~isempty(destinations)
                            obj = findobj(destinationObjs, 'XData', destinations(1,1),...
                                'YData', destinations(1,2));
                            dat = get(obj, 'UserData');
                            out_link(dat.outlink) = true;
                        end
                        % travel upstream 1 roundabout set
                        [origins, destinations, roundabouts] = ...
                            getNodes('up', objs, destinations, []);
                        % initiate Vexit
                        Vexit = zeros(length(udat.inlink),length(udat.outlink),0);
                        % travel further upstream
                        ok = true;
                        while ok
                            % move upstream
                            [origins, destinations, roundabouts] = ...
                                getNodes('up', objs, roundabouts, stopnodes);
                            % add these origin nodes to partial matrix
                            if ~isempty(origins)
                                for o = 1:size(origins,1)
                                    % find object and its data
                                    obj = findobj(originObjs, 'XData', origins(o,1),...
                                        'YData', origins(o,2));
                                    dat = get(obj, 'UserData');
                                    % add in partial matrix
                                    Vexit(dat.inlink, out_link, dat.lane) = 1;
                                end
                            end
                            % stop if no roundabout nodes left
                            if isempty(roundabouts)
                                ok = false;
                            end
                        end
                        % store Vexit and beta per link
                        udat.inlink(i).lane(l).Vexit = Vexit;
                        udat.inlink(i).lane(l).beta = beta;
                    end
                    % store lanemap and inlink order per link
                    if any(sum(lanemap,1) == 0)
                        errordlg(['Some lanes are a dead end at link ' num2str(i) '.']);
                        delete(h)
                        return
                    end
                    udat.inlink(i).lanemap = lanemap;
                    udat.inlink(i).order = order;
```

```
                end
                % delete waitbar
                delete(h)
                % translate generated lanemaps to turn matrix
                [udat ok] = lanemaps2turnmatrix(udat);
                if ~ok
                    return
                end
                if any(sum(udat.node.turnmatrix, 1) == 0)
                    errordlg('Some inlinks are a dead end')
                    return
                end
                if any(sum(udat.node.turnmatrix, 2) == 0)
                    errordlg('Some outlinks are unreachable')
                    return
                end
        end
end
set(fig, 'UserData', udat)

% Translate lanemaps to turnmatrix
function [udat ok] = lanemaps2turnmatrix(udat)
ok = true; % status for lane map existance
% initiate turnmatrix
udat.node.turnmatrix = zeros(length(udat.inlink), length(udat.outlink));
% loop the links
for i = 1:length(udat.inlink)
    if isempty(udat.inlink(i).lanemap)
        errordlg(['Link ' num2str(i) ' does not have a lanemap.'], '')
        ok = false;
        return
    end
    % get turns in lanemap and select indices from the corresponding link order
    these = sum(udat.inlink(i).lanemap,2) > 0;
    udat.node.turnmatrix(i,udat.inlink(i).order(these)) = 1;
end
if sum(sum(udat.node.turnmatrix)) == length(udat.inlink)*length(udat.outlink)
    % all movements defined in the lane maps
    udat.node.turnmatrix = [];
end

% Find conflicting turnflows
function crosses = getCrosses(udat, Uturn)
d = 1; % deviation between in- and outlinks in degrees
maxUturnAngle = 2; % degrees, excluding d, otherwise just another link
% initiate crosses
crosses = zeros(length(udat.inlink), length(udat.outlink), 0);
% loop inlinks
for i1 = 1:length(udat.inlink)
    ai1 = udat.inlink(i1).angle - d;
    x11 = sind(ai1);
    y11 = cosd(ai1);
    % loop outlinks
    for j1 = 1:length(udat.outlink)
        aj1 = udat.outlink(j1).angle + d;
        if innerAngle(ai1, aj1) <= maxUturnAngle + 2*d && ~Uturn
            continue % exclude U-turn
        end
        if ~isempty(udat.node.turnmatrix) && udat.node.turnmatrix(i1,j1) == 0
            continue % impossible movement
        end
        x12 = sind(aj1);
        y12 = cosd(aj1);
        % straight line coefficients
        b1 = (y12-y11)/(x12-x11);
        a1 = y11 - b1*x11;
        % loop inlinks again
        for i2 = 1:length(udat.inlink)
            ai2 = udat.inlink(i2).angle - d;
            if ai1 == ai2
                continue % not from same link
            end
            x21 = sind(ai2);
            y21 = cosd(ai2);
            % loop outlinks again
            for j2 = 1:length(udat.outlink)
                aj2 = udat.outlink(j2).angle + d;
                if innerAngle(ai2, aj2) <= maxUturnAngle + 2*d  && ~Uturn
                    continue % exclude U-turn
                end
                if ~isempty(udat.node.turnmatrix) && udat.node.turnmatrix(i2,j2) == 0
                    continue % impossible movement
                end
                x22 = sind(aj2);
                y22 = cosd(aj2);
                % straight line coefficients
                b2 = (y22-y21)/(x22-x21);
                a2 = y21 - b2*x21;
                % find intersecting point
                if isinf(b1) && isinf(b2)
                    % parallel
                    continue
                elseif isinf(b1)
                    x = x11;
                    y = a2 + b2*x;
                elseif isinf(b2)
                    x = x21;
                    y = a1 + b1*x;
                else
                    x = (a1-a2)/(b2-b1);
                    y = a1+b1*x;
                end
                p = 1000000000; % precision factor, their may be very tiny rouding issues
                if round(sqrt(x^2+y^2)*p)/p <= 1
                    % find duplicates
                    this_conflict = zeros(length(udat.inlink), length(udat.outlink));
                    this_conflict(i1, j1) = 1;
                    this_conflict(i2, j2) = 1;
                    ind = getConflict(this_conflict, crosses);
                    if ~any(find(ind,1))
                        % add if unique
                        crosses(i1, j1, end+1) = 1;
                        crosses(i2, j2, end) = 1;
                    end
                end
            end
        end
    end
end

% Return conclicts, if any
function ind = getConflict(cur, conflicts)
% return indices c where conflicts(:,:,c) is subset of cur(:,:)
% get x and y indices
a = find(cur);
x = rem(a,size(conflicts,1));
x(x==0) = size(conflicts,1);
y = (a-x)/size(conflicts,1) + 1;
% loop conflicts and set cur elements to zero
for c = 1:length(a)
    conflicts(x(c), y(c), :) = 0;
end
% subsets have no elements left (sum = 0)
ind = sum(sum(conflicts, 1), 2) == 0;

% Move upstream or downstream along turbo roundabout map
function [origins, destinations, roundabouts] = getNodes(direction, objs, nodes, stopnodes)
origins = [];
destinations = [];
roundabouts = [];
% get indices: links go from 1,2 to 3,4; movements may go the other way
switch direction
    case 'up'
        from = [3 4];
        to = [1 2];
    case 'down'
        from = [1 2];
        to = [3 4];
end
% remove links to stopnodes
```

```
for n = 1:size(stopnodes, 1)
    objs.links(objs.links(:,to(1))==stopnodes(n,1) &...
        objs.links(:,to(2))==stopnodes(n,2), :) = [];
end
% keep links from a current node
these = false(size(objs.links, 1),1);
for n = 1:size(nodes, 1)
    these = these | (objs.links(:,from(1))==nodes(n,1) &...
        objs.links(:,from(2))==nodes(n,2));
end
objs.links = objs.links(these,:);
% find other nodes per node
for n = 1:size(nodes, 1)
    for l = 1:size(objs.links, 1)
        these = objs.origins(:,1)==objs.links(l,to(1)) & ...
            objs.origins(:,2)==objs.links(l,to(2));
        origins = [origins; objs.origins(these,:)];
        these = objs.destinations(:,1)==objs.links(l,to(1)) & ...
            objs.destinations(:,2)==objs.links(l,to(2));
        destinations = [destinations; objs.destinations(these,:)];
        these = objs.roundabouts(:,1)==objs.links(l,to(1)) & ...
            objs.roundabouts(:,2)==objs.links(l,to(2));
        roundabouts = [roundabouts; objs.roundabouts(these,:)];
    end
end
% keep unique only
origins = unique(origins, 'rows');
destinations = unique(destinations, 'rows');
roundabouts = unique(roundabouts, 'rows');
```