ANALYSIS OF CURRENT DUTCH TRAFFIC MANAGEMENT EFFECTIVENESS WITH AUTOMATED VEHICLES: A RAMP-METERING CASE STUDY

SIMULATION STUDY

Thesis

to obtain the degree of master at the Delft University of Technology

by

Moyu ZHOU

Transport & Planning, Civil Engineering

This thesis has been approved by

promotor: Prof. dr. ir. Hans van Lint

Assessment committee:

Prof. dr. ir. Hans van Lint,	Chair
Dr. Simeon Calvert,	Technische Universiteit Delft
Dr. Henk Taale,	Rijkswaterstaat
Dr. Wouter Schakel,	Technische Universiteit Delft
Dr. Wei Pan,	Technische Universiteit Delft

CONTENTS

SUMMARY

Automated vehicles are conventional vehicles equipped with advanced sensors, controller and actuators. They achieve intelligent information exchange with the environment through the onboard sensing and cooperative system. vehicles are possible to have situation awareness and automatically analyze the safety and dangerous state of journeys. Finally vehicles can reach destinations following drivers' willing. The ongoing research on intelligent vehicles is mainly about improving the safety, comfort, efficiency and provide an excellent human-car interface.

As a self-organizing system, the traffic system is quite complicated. There are many disturbance factors to lead to various traffic problems. One of the daily occurring problems is congestion on the motorway. In order to reduce congestion, Rijkswaterstaat applies various dynamic traffic management (DTM) measures to guide the traffic. It works well nowadays in conventional traffic. However, automated vehicles entered the market recently and will start to play an essential role in future traffic. The automated vehicles' reaction to DTM measures may be different from conventional vehicles while the traffic problems still exist. Therefore, it is necessary to research the effectiveness of current Dutch traffic management in automated vehicles.

This thesis aims to investigate the effectiveness of current Dutch DTM measures with driver assistant and partially automated vehicles. Due to the time limitation, only the ramp metering measure will be researched through a simulation study. Therefore the main research question is 'How partial automated driving influences the performance of current Dutch dynamic traffic management system and how can this be evaluated via simulation?'. Three methods are applied, including literature review, simulation and statistical analysis.

The literature part reviews levels of automation, various longitudinal and lateral vehicle motion models, which are chosen and modified in the simulation. Many ramp metering algorithms are also introduced in the literature review. The ramp metering controller in the simulation follows RWS algorithm. Besides, the motorway demand and the penetration rate of level 1 and 2 vehicles are two input of the simulation.

From the simulation results, it is concluded that the level 2 automation consisting of Adaptive Cruise Control (ACC) and Lane Change Assistance (LCA) system brings a negative impact on the motorway capacity. The ramp metering measure remains efficient if the penetration rate of level 2 vehicles is low. However, when the capacity reduces to the critical flow set up in the ramp metering controller, Ramp metering loses its efficiency. The parameters in the ramp metering controller therefore, require an update. For further research, it is recommended to simulate the same scenarios with different ramp

metering algorithms. Since the functions of the algorithms are different, there might be other robust control algorithms for automated vehicles. Besides, another limitation of this thesis is that the automation system in level 2 vehicles is defined as Adaptive Cruise Control (ACC) plus Lane Change Assistance (LCA) system. Other partial automation systems may have a different effect on the performance of ramp metering. This thesis can be expanded by research the ramp metering performance under various types of partial automation systems.

PREFACE

This report concludes my graduation work at the faculty of Civil Engineering and Geosciences at the Delft University of Technology for the master study Transport Planning Infrastructure. The report embodies the research from February 2019 until the end of October 2019 at TU Delft and Rijkswaterstaat.

This thesis report represents the closure of my two-year master study career in this university, this city and this country, where is full of challenges, new experiences, confusion and gratitude. It was a great honour for me to be able to work with Rijkswaterstaat to conduct research on traffic management, with special thanks to Prof. Hans van Lint for offering me this wonderful research opportunity.

I would like to thank my MSc Thesis committee group for all their valuable direction and feedback. Thank you Hans van Lint, Henk Taale for all your constructive guidance and advice, which guided me towards this final report. Thank you Simeon Calvert, as daily supervisor, always kept me on the right track when the moment I am missing direction. I also want to thank Wei Pan for joining my thesis committee and helping me deeply understand the research.

Furthermore, I want to thank Wouter Schakel, my programming supervisor, supported me with programming issues. Thank you my love Zhirun Gan, for encouraging me to keep moving forward and gave me a lot of confidence when I was stuck in the middle of the research. I would like to express my gratitude to my family and friends, who are always standing by me during the hard time of these two years.

Moyu Zhou
Delft, October 2019

LIST OF FIGURES

LIST OF TABLES

1

INTRODUCTION

1

The invention of vehicles brought great convenience to human beings as it largely saved travelling time and improved comfort. However, 'driving' is an indispensable action leading to fatigue and danger to some extent. Drivers need to pay attention to the surrounding environment and vehicles at all times and make corresponding judgements. These judgements mostly depend on the driver's driving experience and concentration. The majority of the incidents are driven by human error. People then came up with the idea of automation to improve safety and comfort by using driving system to replace human drivers.

As the national road authority of the Netherlands, Rijkswaterstaat is always concerned with traffic management in order to best utilize the infrastructure. With the implementation of automation, the behavior of traffic may change therefore the traffic measures might change correspondingly. This thesis will investigate the effectiveness of Ramp Metering in mixed traffic including automated vehicles and conventional vehicles.

Since the current idea of dynamic traffic management is little related to intelligent vehicles, the introduction of these two terms will be a bit dissociated.

1.1. AUTOMATED VEHICLES

Automated vehicles are conventional vehicles equipped with an 'autonomous system' that contains advanced sensors (such as radar, lidar or camera), controller and actuators, so that are possible to move with less human control than conventional vehicles [1]. The autonomous system facilitates information exchange within vehicles and infrastructures through sensing system and controller. Then it precepts the environment and automatically analyze the safety of the possible routes and monitor the unexpected conditions in time. Under the cooperation of autonomous system and human operation, autonomous vehicles reach the destination with smooth interactive experience.

According to SAE standard shown in Figure 1.1, there are overall six levels of automated vehicles. Level 0 is conventional vehicles without the intelligent driving system. Level 1 aims at vehicles that equipped autonomous assistance which may help to control the crosswise or longitudinal distance. Adaptive Cruise Control (ACC) are typical level 1 autonomous system which has been widely applied in commercial vehicles. For example, Toyota has added its laser ACC system towards Japanese and American Market during late 2000 [2]. Audi, BMW and Ford have also developed their ACC systems, which was launched in market recent years. The stepwise development of autonomous vehicles would promote level 2 automation for commercial use in a short time.

The typical level 2 autonomous vehicles (also known as partial automation) are vehicles equipped with assistant systems providing both the longitudinal and the lateral driving assistance. For example, with the aid of Adaptive Cruise Control (ACC) and Lane Change Assist (LCA) systems, level 2 automation could be realized since ACC would control the following distance of the predecessor and LCA would intervene the steering if unexpected object is detected [4].

SAE level	Name	Narrative Definition	Execution of Steering and Acceleration/ Deceleration	*Monitoring* of Driving Environment	Fallback Performance of *Dynamic Driving Task*	System Capability *(Driving Modes)*
***Human driver* monitors the driving environment**						
0	No Automation	the full-time performance by the *human driver* of all aspects of the *dynamic driving task*, even when enhanced by warning or intervention systems	Human driver	Human driver	Human driver	n/a
1	Driver Assistance	the *driving mode*-specific execution by a driver assistance system of either steering or acceleration/deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	Human driver and system	Human driver	Human driver	Some driving modes
2	Partial Automation	the *driving mode*-specific execution by one or more driver assistance systems of both steering and acceleration/ deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	**System**	Human driver	Human driver	Some driving modes
***Automated driving system* ("system") monitors the driving environment**						
3	Conditional Automation	the *driving mode*-specific performance by an *automated driving system* of all aspects of the dynamic driving task with the expectation that the *human driver* will respond appropriately to a *request to intervene*	System	**System**	Human driver	Some driving modes
4	High Automation	the *driving mode*-specific performance by an automated driving system of all aspects of the *dynamic driving task*, even if a *human driver* does not respond appropriately to a *request to intervene*	System	System	**System**	Some driving modes
5	Full Automation	the full-time performance by an *automated driving system* of all aspects of the *dynamic driving task* under all roadway and environmental conditions that can be managed by a *human driver*	System	System	System	**All driving modes**

Figure 1.1: Summery Table of SAE Levels [3]

Overall, low level of automation will have different car following or lane change behaviors from conventional vehicles depending on the its design domain and parameters. Other behaviors such as destination and route choice will be the same as conventional vehicles.

As for high level of automation, level 3 vehicles can monitor the driving environment themselves and only require fallback control when system break down. Level 4 and 5 vehicles achieve driver-less in some driving modes or all the driving modes, i.e. the automated systems control every vehicle behaviors which to large extent are differ from conventional vehicles.

1.2. INTRODUCTION OF DYNAMIC TRAFFIC MANAGEMENT

With the growth of traffic demand, a number of traffic problems such as congestion, high incident rates and increasing environmental pollution are becoming more and more serious. As the above traffic problems restrict people's daily life and national economic development, the possible solutions are driven much attention. The traditional way to solve these traffic problems is to improve or develop the infrastructure. However, due to the scarcity of space and investment, the expansion of traffic infrastructure is not always feasible. In addition, it is hard to solve these problems perfectly for mega transportation

system with high level of complexity. Therefore, the concept of Dynamic Traffic Management (DTM) appears, aiming to solve the above traffic problems at an integrated level thoroughly taking vehicles, infrastructure and the environment into consideration. This section will explain some important dynamic traffic measures.

Dynamic Traffic Management (DTM) appears as the traffic managing system that guides the vehicles, infrastructure and driving environment in order to achieve a more efficient, more effective and safer transportation environment [5]. It could carry out the collection, transmission, storage, analysis, processing and application of traffic information, which is also the transformation from simple static management to intelligent dynamic management. Inside DTM, both static and dynamic information is shared and utilized by travellers, drivers, remote controllers and researchers. Finally, it may realize the dynamic optimization operation of the extensive transportation system and cope with the growing traffic demand. In the following paragraphs an introduction about DTM measures and their aims will be given.

Variable Speed Limit (VSL) is a kind of digital sign with the corresponding appropriate speed limit with respect to the current traffic situations [6]. VSL aims to alert drivers to the incoming road situation such as downhill or accident areas and improve traffic safety by avoiding the individual over speeding. However, over-confident drivers may not follow the road instruction and speeding limits sometimes. The developed automated system in the future should be able to have access to the flexible speed limits and comply with them at the right moment. In other words, current VSL measure is more likely to be effective in mixed traffic.

Dynamic Route Guidance (DRG) is a measure consisting of the Variable Message Sign (VMS) and in-vehicle message. VMS might show the possible delay, jam length, travel time or the combination of free flow travel time and delay of the surrounding roads. Drivers will choose their route according to the above message, so that the traffic flow will be redistribute through VMS. The in-vehicles message is quite similar to VMS but more private based on the vehicle destination and possible routes. It may provide alternative routes avoiding delay and congestion. Note the in-car message functions to optimize the network, which is not necessary for vehicle navigation system. However, level 2 automated system only control the tactical behaviors such as car following and steering. Route choice belongs to strategical behaviors which is still one of the drivers' tasks. Therefore, the performance of DRG will not change in mixed traffic with level 2 vehicles.

Ramp Metering (RM) focuses on maintaining the free flow condition on the mainline of the motorway. Although most individual vehicles at the on-ramp will follow the instruction of metering system, the effect of each metering is quite different due to disturbances during merging maneuver. When it comes to mixed traffic, the effect will become more uncertain since the partially automated vehicles follow different car following and merging algorithms.

Table 1.1: Comparison of Different DTM measures

	Objectives	Effects	Performance in mixed traffic
VSL	Improvement of Traffic Safety and Efficiency	Reduction of Unusual Individual Performances (over speeding)	Better than Conventional Traffic due to the Default Setting
DRG	Network Optimization	Traffic Redistribution of the Whole Network	Similar
RM	Maximum Utilization	Free flow at the Downstream Bottleneck	Unknown due to the Different Microscopic Driving Models

Besides, there are other DTM measures in the Netherlands such as Extra Lanes at Rush hours and Incident Management Camera. Due to the unknown behaviors of automated vehicles, ramp metering will be the focused measure with automated vehicles which is introduced in Section 1.4.

1.3. RESEARCH GAP

The dynamic traffic management system has been applied in the Netherlands since 1980 for solving the increasing demand. Many measures have been implemented on the motorway and achieved great success. For example, the first ramp metering was installed at S101 to the A10-West in 1989 and released the congestion pressure before Coentunnel. Some drivers shift to A10-West because of the benefit from the metering system. In this way the system reorganized the traffic through more even traffic distribution [7]. Dutch motorways have also installed Dynamic Route Information Panels (DRIPS). It is a kind of VMS which is part of the dynamic route guidance measure. According to the traffic data in 2000, the congestion of Amsterdam Ring road reduced 25% [8]. Over years, Dutch government is researching and trying to improve the effectiveness of these measures.

However, the development of autonomous vehicles will lead to mixed traffic flow that contains partial autonomous vehicles and driver assistance vehicles. The mixed traffic may embody different car-following and lane changing behaviors. Thus the macroscopic performance of mixed traffic becomes unknown owing to the unpredicted behavior of automated vehicles and the corresponding reaction of driver assistance vehicles towards the changing traffic. Rijkswaterstaat want to eliminate this uncertainty so that took an interest in the continuation of the DTM success.

1.4. RESEARCH SCOPE

This research is based on the dynamic traffic management case in the Netherlands, while not every detail can be reflected in the model. In order to clarify the boundary of this research, several aspects have to be considered respectively.

- **DTM Measures**
  There are various of DTM measures applied in the Netherlands. The characteristics and effects of each measure are quite different. Table 1.1 illustrates the objec-

tive and effect among VSL, DRG and RM. Automated systems may have less reaction time than drivers if they get the access to detect the speed limit. The aim of DRG is network optimization which influence the route choice of travelling. However, low-level of automation cannot make decisions at strategic level. Human drivers are still the one to make route choice in DRG. Most of time, DRG performs the same with automated vehicles and conventional vehicles in terms of the route choice. Thus VSL and DRG are excluded from this thesis. On the other hand, the effect of ramp metering is quite uncertain. It depends on traffic flow and average speed, which is the accumulative behavior of individual vehicle motion of automated vehicles. Therefore, ramp metering is chosen to research its effectiveness with automated vehicles.

- **Vehicles**
  Literature states that the percentage of automated vehicles will significantly rise from the year 2020 [9]. Although human drivers will change their behaviors when level 2 vehicles appear in the traffic, those changes are difficult to model in the simulation. Besides, ramp metering works efficiently in conventional vehicles nowadays according to the annual report of Rijkswaterstaat [10]. This thesis aims at researching the effectiveness of ramp metering on automated vehicles so there will be only two classes of autonomous vehicles in the simulation. They are level 1 autonomous vehicles and level 2 autonomous vehicles. To some extent, level 1 vehicles will become the one of the main traffic compositions in the future . Therefore, the primary traffic unit is chosen as level 1 autonomous vehicles and level 2 vehicles gradually participate into the traffic.

1.5. RESEARCH OBJECTIVES AND QUESTIONS

In the past few decades, the Netherlands has applied different kinds of motorway measurements such as ramp metering, dynamic route guidance and dedicated lines. Most of them produced high efficient management on motorway traffic.

After automated vehicles pouring into commercial market, traffic on motorway becomes changeable and unpredictable. Rijkswaterstaat (Dutch Ministry of Infrastructure and Water Management) shows great interest on how to manage the mixed traffic as the remote controller, aiming at developing effective and efficient transportation system. Therefore, the objective of this thesis is to *investigate the effectiveness of current Dutch traffic management in mixed traffic including level 1 and level 2 automated vehicles.*

Based on the research gap and objective, the research question is: **How partial automated driving influences the performance of current Dutch dynamic traffic management system and how can this be evaluated via simulation?**

It leads to the following sub-questions.

a) Which Dutch dynamic traffic management measure will be researched in this thesis and how is it developed? How is the chosen Dutch dynamic traffic management measure in this thesis developed from literature?

b) What are the microscopic driving models of partially autonomous vehicles and driver assistant vehicles?

c) How will partial autonomous vehicles influence the macroscopic behaviors of mixed traffic under different penetration rate?

d) How will current measure selected in sub-question (a) influence the macroscopic behaviors of mixed traffic?

e) Could current measure selected in sub-question (a) be improved and how to improve?

1.6. RESEARCH METHODOLOGY

This section explains the general methodology of this research including literature review, simulation study and statistical analysis. The answers to research questions will be obtained through the following methods.

The choice of DTM measure in sub-question (a) has been answered in the introduction part 1.2. For some possible measures, there is a summary table about their objectives, effects and possible problems with automated vehicles. It is difficult to predict the effectiveness of all DTM in mixed traffic due to the time limitation of this research. Based on that table, VSL or DRG are more likely to behave similar or better in the mixed traffic rather than the conventional traffic. However, the performance of ramp metering is influenced by the microscopic behaviors of automated vehicles, which are the reflection of automated system definition and design. For example, ACC vehicles will adjust driving behavior when their system setting of car-following distance is changing. Roadway capacity increases if ACC vehicles have smaller car-following headway and vice versa [4]. Therefore, this research interest is ramp metering measure and the target DTM measure is chosen as ramp metering in this thesis.

1.6.1. LITERATURE REVIEW

Literature Review will give part of the answer to sub-question (a), (b) and also some background knowledge about sub-question (c). In our research, the literature review consists of three parts. The first part will review the traffic impact of intelligent vehicles which is related to sub-question (c). The second part is about ramp metering measure including its theories and worldwide development, answering the last part of sub-question (a). Finally, a list of microscopic car-following models and lane change models will be discussed. Some microscopic models of intelligent vehicles are chosen from this section to simulate vehicle behavior in Section 3.2.2. In addition, some of them are modified based on the existing models..

1.6.2. SIMULATION

Sub question (c) and (d) will be answered by simulation and statistical analysis of simulation outputs 1.6.3. Simulation is a process to imitate real traffic situation based on traffic theory and computer technology to simulate the calculation model of the real traffic system. Traffic simulation reflects the characteristics of the actual traffic system and performs the possible behaviors of traffic in various situations. It can also optimize solutions before real implementation. As the development of intelligent vehicles is still on-going (especially partial autonomous vehicles), some vehicles are not permitted to drive on the road due to the safety concern. Meanwhile, the time cost and vehicle availability are also uncertain. It is therefore difficult to conduct a field test. Therefore, the simulation method with good experimental control and validity is one of the most suitable methods for this research.

The vehicle behaviors will be modelled based on the answer of sub-question (b) in the simulation. The simulation environment, ramp metering controller and some vehicle parameters are designed according to the Dutch motorway situation. The outputs of the simulation will be further processed in statistical analysis.

1.6.3. STATISTICAL ANALYSIS

Statistical analysis is a method to effectively present the results of samples. To guarantee the accuracy of results, it also has some requirements on samples such as the number of samples and simulation runs. This thesis will use both descriptive statistic and statistical inference to analyze the performance indicators. The descriptive statistic is used to analyse the speed (v), total time spent (TTS) and total delays (D). The statistical inference, i.e., the product limit method (PLM) is used to calculate the stochastic capacity. The detailed analysis is explained in Chapter 5. Statistical analysis works together with simulation and gives the idea of sub-question (e)' answer.

1.7. THESIS OUTLINE

This chapter gives an introduction to some elements (intelligent vehicles and dynamic traffic management measures) of this research. Then the research gap is given, forming the research objectives and questions. After that, the proposed methodology is introduced to answer the research questions.

Chapter 2 is a literature review explaining the historical researches about traffic impact of intelligent vehicles. It then introduces the underlying theories and algorithms of ramp metering. In the last section of this chapter, a list of car-following models and lane change models are reviewed.

Chapter 3 explains the design of the simulation including software choice, experimental setup, simulation runs, simulation assumptions and the calculation of performance indicators. In the next chapter 4, the simulation is verified and validated.

Chapter 5 processes the outputs of simulation using the statistical analysis and Chapter 6 conclude this research with some recommendations.

2

Literature Review

2

The previous chapter introduces the research scope and forms some sub-questions to achieve the research objectives. This chapter will introduce some microscopic driving models and answer how ramp metering is developed in sub-question (a). Also, it explains some back-ground knowledge about sub-question (b) and (c). Some researches about traffic impact of intelligent vehicles are also reviewed. Meanwhile, this chapter explains the ramp metering theory, algorithms and historical development.

The following goals are set up as the direction of this chapter.

1) To get the state-of-art about the modelling of vehicle behaviors and some possible traffic impact of intelligent vehicles.

2) To understand how ramp metering measure works.

3) To introduce the development and deployment of ramp metering strategies, understand the existing changes of implementation and possible effectiveness.

4) To identify different algorithms of ramp metering and describe the algorithm of current Dutch traffic management.

2.1. MICROSCOPIC VEHICLE BEHAVIOR MODELS

Microscopic model simulated the behaviors of single vehicles incorporating their dynamics such as velocity, acceleration and position. According to van Wageningen-Kessels et al. [11], all microscopic models have the assumption that drivers would adjust their behaviors based on the performance of their predecessors. This section includes the introduction of both car-following and lane change models.

2.1.1. FAMILY TREE OF CAR FOLLOWING MODEL

The car-following model follows the family tree shown in Figure 2.1. It begins with the safe-distance model branch presented by Pipes, Kometani and Sasaki, Newell and Gipps. Then the stimulus-response models start to develop including widely used models such as Helly model, Optimal Velocity Model (OVM) model and Intelligent Driver Model (IDM). The final branch is about cellular automata (CA) begins at 1986.

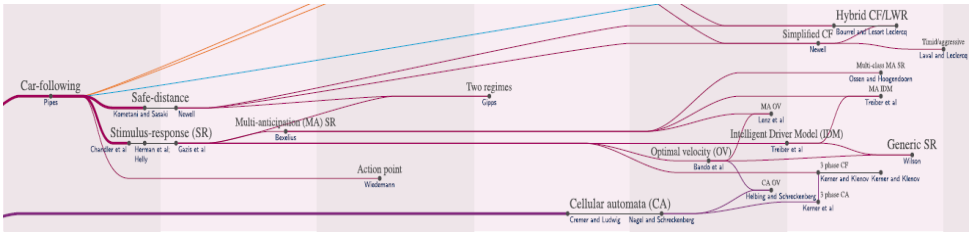


Figure 2.1: Genealogical Tree of Car Following Models [11]

The earliest car following model was published in 1953, which is called Pipes safe-distance model. The distance is the sum of a distance proportional to the velocity of the following vehicle and a certain given minimum distance of separation at standstill state [12]. The function of the Pipes safe-distance model is shown below.

$$x_{n-1} = x_n + d + T * v_n + l_{n-1}^{veh} \tag{2.1}$$

In which:

- d is the distance between two vehicles at standstill
- l_{n-1}^{veh} is the length of the leading vehicle
- $T * v_n$ is the 'legal distance' between vehicle 40 $n-1$ and n

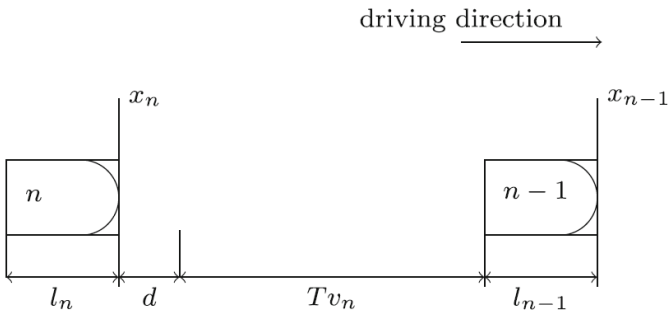


Figure 2.2: Parameters of Pipes' Safe-distance model [12]

The model presented by Kometani and Sasaki is the product of Newtonian equations of motion. The drivers in this model are assumed to react after a certain time according to the unexpected behaviors of their predecessors. In other words, the standstill distance would change to distance based on their individual speed [13]. Gipps model is based on the assumption 'each driver sets limits to his desired braking and acceleration rates'. Then the speed of a vehicle will be chosen from the safe distance according to the driver and the limitations of the vehicle permit. [14].

Another important car following model based on safe-distance is Newell model [15] which is further transformed into a simple version shown below.

$$x_n(t+\tau_n) = x_{n-1}(t) - d_n \tag{2.2}$$

In which:

- x_{n-1} is the position of the front vehicle
- x_n is the position of the following vehicles

The formula states the position of the vehicles is the function of the position of predecessors, time delays and the individual safe distance. As shown in Figure 2.3, vehicles would copy the trajectories of the front vehicles with the approximate value d_n and τ_n [16].

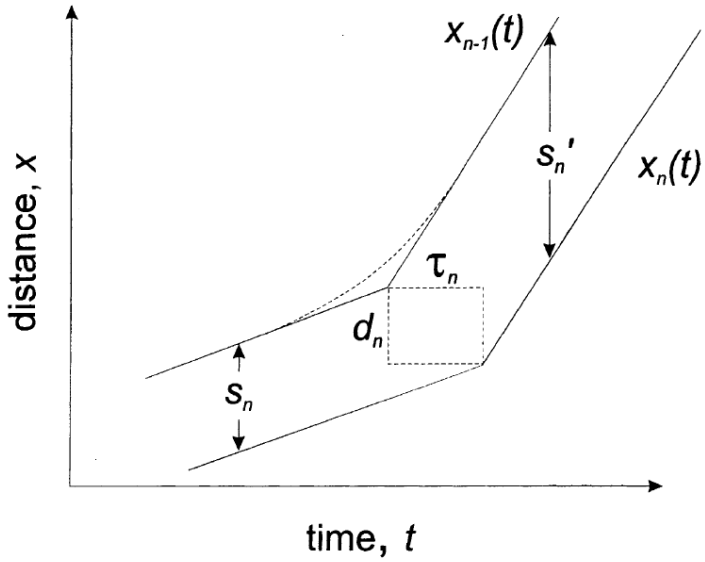


Figure 2.3: Picewise Linear Approximation to Vehicle Trajectories [15]

This branch is further developed into hybrid CF or LWR by Leclercq [17].

The second branch is stimulus-response models that were first built around 1960. The framework is the primary modelling based on follow-up behavior, which regards the behavior of the predecessor as a stimulus to the driver. The driver's perception of the stimulus is a sensitive coefficient as part of the driver's response and then lead to the behavior of the vehicle. Among them, Helly model is quite widely applied in which prescribes a desired spacing s^* as function of the speed v and time headway T. The spacing here is dependent on the standstill distance s_0, net time headway T and the individual speed v.

$$s^* = s_0 + T * v \tag{2.3}$$

The acceleration is determined by a desire to drive at the same velocity as the front vehicle and a desire to drive at the specific individual time headway. The model prescribes the following acceleration.

$$a(t) = \alpha(\Delta v(t-\tau)) + \gamma(s(t-\tau) - s*) \tag{2.4}$$

In which:

- Δv is the velocity difference
- τ is the reactive time
- t is a specific moment

Bando et al come up with the optimal velocity model (OVM) in 1995 to solve the large acceleration or deceleration during the start and stop period of the vehicles in the Newell model. It also tries to investigate the instability of the steady-state traffic flow [18]. The model assumes the drivers would accelerate or decelerate according to difference of optimal speed and the optimal speed is determined as follows [19].

$$a = a_0(v^* - v) \tag{2.5}$$

$$v^* = 16.8(\tanh(0.086(s-25)+0.913)) \tag{2.6}$$

Where a_0 is a reference acceleration (tunable parameter which is constant for a specific vehicle-driver combination)

Another essential car-following model is called Intelligent Drivers Model (IDM) proposed by [20]. This model would present the trend of acceleration in free flow and the consideration of deceleration trend of collision with predecessor during car following state. The function of desired acceleration ($\dot{v_a}$) and desired spacing ($s^*(v,\Delta v)$) is shown below.

$$\dot{v_a} = a^{(\alpha)}[1-(\frac{v_\alpha}{v_0^\alpha})^\sigma - (\frac{s^*(v_\alpha,\Delta v_\alpha)}{s_\alpha})^2] \tag{2.7}$$

$$s^*(v,\Delta v) = s_0^{(\alpha)} + s_1^{(\alpha)}\sqrt{\frac{v}{v_0^\alpha}} + T^\alpha v + \frac{v\Delta v}{2\sqrt{a^{(\alpha)}b^{(\alpha)}}} \tag{2.8}$$

In which:

- Δv is the speed difference between the leader and the follower
- a is acceleration
- α is the vehicle
- b is comfortable deceleration
- a_0 is a reference acceleration (parameter)

The last branch is the Cellular Automata (CA) models. It is defined as a dynamic system that evolves in a discrete-time dimension in a cellular space consisting of a discrete, finite state of cells according to local rules. Since the traffic elements (vehicles) are discrete in reality, the cellular automata theory is used to model the car following behavior, which might avoid the approximation process from discrete to continuous and back to discrete. To some extent, it has its unique advantages. It was first introduced by [21] and

[22] developed some of the most popular [23].

Overall, the safe-distance branch emphasizes the safe vehicle gap between each other. The stimulus-response models underline the follow-up behaviors of drivers. As for the cellular automata models, they highlight vehicles states.

2.1.2. REVIEW OF LANE CHANGE MODEL

Apart from various of car-following models, the lateral behavior of vehicles is also essential since people start to pay attention to multilane behaviors. There are two types of lane changing including mandatory and discretionary lane change. The former one appears during merging, lane closing and obstacle detection while the latter one appears when there are slow-moving vehicles [24]. Some popular lane change model such as MOBIL, LMRS will be listed in the following sections.

Slug and Rabbit model is a model considering the characteristics of different drivers [25]. It divides the drivers into two classes called slugs and rabbits. Slugs would have maximum velocity v_f while rabbits have the maximum velocity $V_f > v_f$. There will be two kinds of regime in this model. First of all, the traffic would have a two-pipe regime in which slugs stay at the right lane operating at their maximum speed and rabbit always stay at left lane in order to overtake slugs.

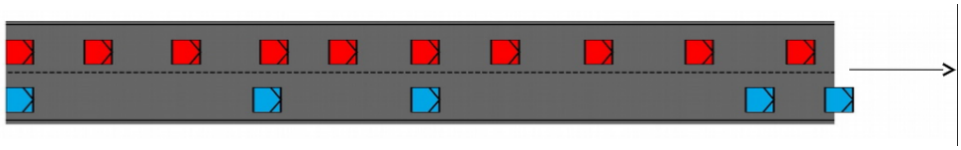


Figure 2.4: Two-pipe Regime (red for rabbits and blue for slugs) [26]

However, a two-pipe regime would only appear at low density. With the increasing density of rabbits, the overall speed of the left lane would reduce and gradually become lower than the right lane. The rabbits would like to achieve their maximum speed by overtaking from the right lane. The above behaviors happen until there is no speed difference of two-lane therefore comes to one-pipe regime.

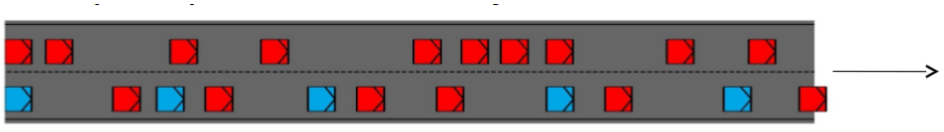


Figure 2.5: One-pipe Regime [26]

MOBIL model is another lane change model based on utility. When the subject ve-

hicle follows another vehicle, the driver would decide if it is good to start lane changing [27]. Assume the current driving environment is three conventional vehicles is accelerating on two lanes and one car at the front side. It will define a utility of subject vehicle as U_x, so the lane-changing vehicle is indicated by x, new follower by b, old follower by a and predecessor by T. The utility function for objective vehicle x is represented as below.

$$U_{changing} = a_x^{'} + p * (a_a^{'} + a_b^{'}) \tag{2.9}$$

$$U_{stay} = a_x + p * (a_a + a_b) \tag{2.10}$$

$$U_{changing} - U_{stay} = a_x^{'} - a_x + p * (a_a^{'} - a_a) + p * (a_b^{'} - a_b) \tag{2.11}$$

(Here p indicates the politeness factor, $p \in [0,1]$)

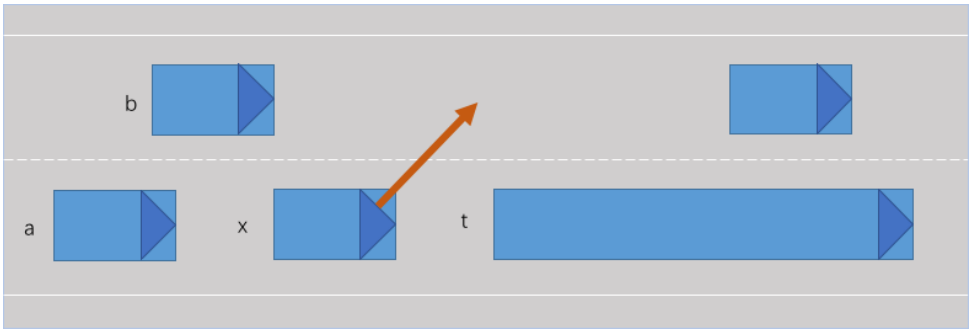


Figure 2.6: MOBIL Lane Changing Model of x

In this case, if the benefit of lane changing larger than a specific threshold Δa, the lane-changing could happen. A trade-off between cost and benefit will be performed when there is a lane changing desire.

Based on the above research, one integrated lane change model called Lane Change model with Relaxation and Synchronisation (LMRS) was proposed [28]. The model assumes that drivers prepare for a lane change in their acceleration, leave gaps for others merging into the traffic stream and only gradually adapt their headways after someone merged in front.

$$d^{ij} = d_r^{ij} + \Theta_r^{ij} * (d_s^{ij} + d_b^{ij}) \tag{2.12}$$

In which:

- d_r is a desire to follow a route
- d_s is a desire to gain speed
- d_b is a desire to keep right (b stands for bias to a particular side)

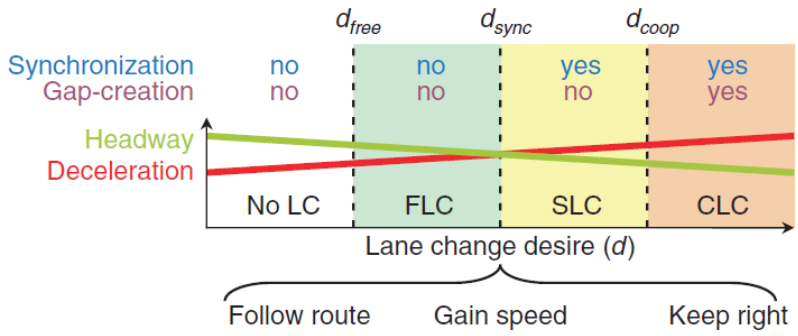


Figure 2.7: Overview of Lane Changing Behavior based on LMRS [28]

- Θ_v is the level at which voluntary (discretionary) incentives are included

Based on the calculated desire, three different thresholds are set up to classify the behaviors of the lane change. They are Free lane changes (d_{free}), synchronized lane changes (d_{sync}), and cooperative lane changes (d_{coop}) which is shown in Figure 2.7.

Generally, lane change models are built from real world behaviors which slow vehicles drive in right lanes and fast vehicles run in left lanes. Details such as trade-off, politeness and relaxation are being added over time in new models.

2.2. LITERATURE ABOUT THE TRAFFIC IMPACT OF INTELLIGENT VEHICLES

It is common to predict the traffic impact of newly implemented technology at the early use of technology in order to find out its rationality and effectiveness. The purpose is to improve the relevant hardware or software before the real application. Although the traffic impact of high automation is a relatively new topic, substantial researches aim at the traffic impact of lower level of intelligent vehicles such as ACC vehicles. Besides, there are some newly developed models describing the behaviors of highly automated vehicles, which will also contribute to the simulation part of this research.

At level 1 stage, vehicles equipped with longitudinal assistance would improve the roadway capacity. Spiliopoulou [29] developed an ACC based control concept with idea of behavior adaptation. Then they simulate the traffic behavior at on- and off-ramp based on this integrated model and found improved motorway traffic efficiency with the increase of ACC penetration rate. The conclusion points out the behavior adaptation in their essay only consider the time gap. The combination of adaptation and other driving systems is worth further researching.

Apart from the automation itself, the penetration rate difference will also influence traffic behavior under RM. There are some researches aiming at the traffic impact of automation without RM. For example, ACC is a kind of systems having quite high requirements of communication technology that is one of the deficiencies of conventional vehicles. The study of the research [30] the influence to capacity at merging bottleneck according to different penetration rate of ACC vehicles in a multi-lane motorway and compare it with single-lane cases. Based on the simulation, the capacity is shown to improve with the increase of ACC vehicle penetration rate but 13% less at multi lanes than at single lane. In other words, the lane change maneuver leads to negative influence on roadway capacity. Therefore, the authors suggest further research conducted in the area of coordinating advanced merging assistance strategies with the ACC string operation to mitigate the capacity reduction at merge areas. Calvert, Schakel, and van Lint [9] investigate the impact of low level of automated vehicles (from level 1 to level 3) to traffic flow. They find the low penetration rate of low level of automated vehicles might have negative impacts and perform further experiment that states that there will be improvement in traffic flow when the penetration rates above 70%. Therefore, the penetration rate of intelligent vehicles is an essential factor when considering the road behavior of mix traffic.

2.3. TRAFFIC FLOW THEORY ABOUT RAMP METERING

In this section, the design principle of ramp metering will be discussed in detail. Ramp metering is one of the vital traffic management methods which aims at controlling the on-ramp traffic to improve the highway traffic away from congestion. One of the main infrastructures of ramp metering is a special kind of traffic light at the on-ramp which has only red and green operating at peak hour (shown in Figure 2.8). Special case might be the existence of yellow light because of the traffic rules. In this case, the time of yellow light after green would become much shorter than conventional one. Then the red light appears after the short flash of yellow light and forms the cycle. The standard traffic light colors including green, yellow and red are used in the Netherlands. The difference with a standard traffic light is the color of the background shield changes to yellow instead of black. The strategy of ramp metering is 'One (two) car per green' independent of the number of lanes at the on-ramp. Usually there is only one lane at the on-ramp and it follows the strategy 'One car per green'. In other words, the traffic light at the on-ramp starts to operate when the loop detector shows the input flow or density (somewhere smaller than threshold values for flow and speed). The on-ramp demand is reduced and the density of highway after the on-ramp is controlled less than critical density by metering strategy. Therefore, it will ensure the smooth operation of mainline traffic. By the aid of ramp metering, the flow of mainline tries to make the full use of infrastructure by avoiding capacity drop.

The traffic flow theory forms the base of ramp metering. There are three variables to describe the traffic state of a Fundamental Diagram (FD). They are time or space mean speed (v)[km/h], density (k)[veh/k] and flow (q)[veh/h]. The relationship between these three variables is $q = u * v$. As mentioned before, ramp metering tries to avoid capacity

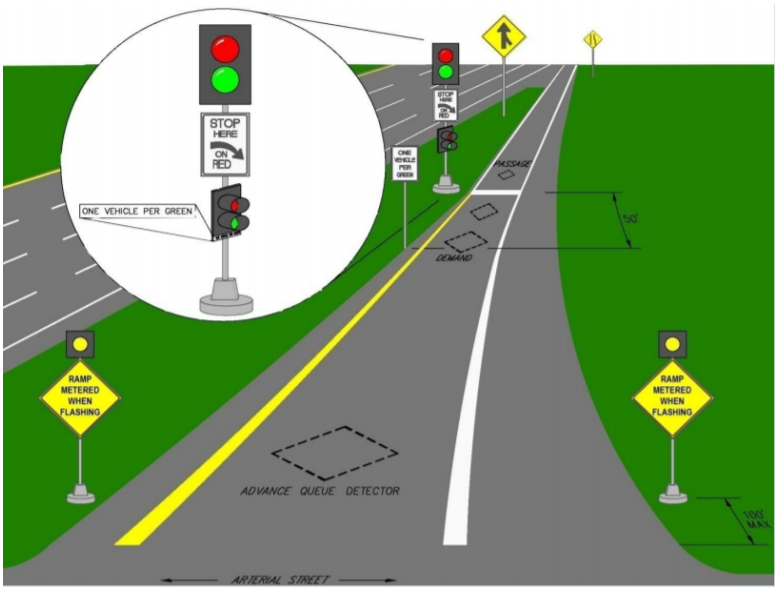


Figure 2.8: Ramp Metering [31]

drop, which is a typical traffic phenomenon that the maximum flow before 'congestion' state is more significant than after at the bottleneck. Kerner [32] explains it as the flow reduction from capacity to free flow condition of the downstream. It is shown clearly in the example FD. Figure 2.9 shows that the maximum flow would reduce about one thousand vehicles per hour in the tested road. The capacity drops results from reduced efficiency during acceleration after the bottleneck. The reduced flow after congestion is called queue discharge rate. Here it is the maximum flow at downstream after the bottleneck [33]. The detailed formulas of ramp metering will be explained in section 2.6.

2.4. WORLDWIDE DEVELOPMENT OF LOCAL RAMP METERING

In this section, the ramp metering will be discussed in detail. Ramp metering is a kind of efficient traffic measures dated back to 1960 in the United States. At the beginning of 1960, it was tested inside Detroit, New York, and St. Louis as different kinds of layouts. Soon afterwards, Chicago, Houston and Los Angeles followed the measure in the middle of 1960's. Log Angeles keeps on developing its ramp metering system including around 1300 meters and achieves the worldwide largest system now. Figure 2.10 shows the amount of ramp metering inside the United States at the end of twentieth century as well as other countries who devote themselves to metering systems. Other countries from North America, Europe, Asia and Australia all started to develop their metering systems.

During the 1980's, the idea of ramp metering started to spread inside Europe. The United Kingdom first installed on the seventh junction of motorway M6 near Walsall at

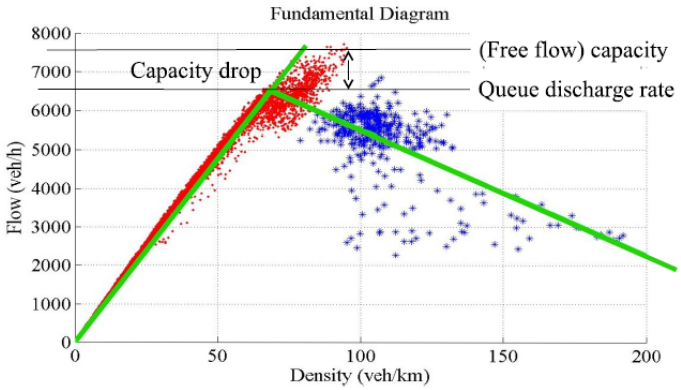


Figure 2.9: Capacity Drop in Fundamental Diagram based on Real Data [19]

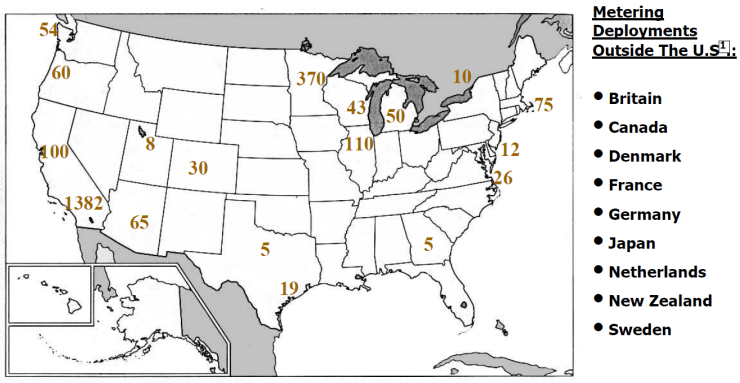


Figure 2.10: Ramp Meter Deployments - USA and Other Parts of the World [34]

1986. However, this ramp metering was removed at late 1990s due to the outdated technique. New metering system (RTIC project) was set up and operated at M3 and M27 from 2000 to 2002. In the next year, the evaluation of UK transportation towards RTIC project shows 1.5% to 5.5% increase of the throughput and 9.5% to 13% decrease in travel time at the morning peak [35]. Those positive effects lead to further development (up to 88 ramp metering installations by the time of 2011) of British ramp metering system.

Soon afterwards, the Netherlands started its trial of ramp metering. Metering was installed firstly as art of circumferential motorway in Amsterdam at around 1989 [37]. Due to the unbalanced demand and support, there is regular congestion at motorway S101. The first trail here releases the traffic pressure and improve the average speed of vehicles. The Rijkswaterstaat wanted to continue this effective traffic management measure and selected Delft-Zuid as the second location of ramp metering. As a result, it solves the

Figure 2.11: Location of Dutch Ramp Metering in 1998 [36]

'stop and go' situation during peak hours and leads to smooth-running traffic. By the time 1998, there are 20 ramp metering systems located around Amsterdam, Rotterdam and Utrecht. Detailed map about Dutch metering at that time is shown in the red area of Figure 2.11. In 2018 there are 136 metering systems operational, mainly in the Western part of the Netherlands.

The Netherlands also start the project called Praktijkproef Amsterdam (Field Operational Test Integrated Network Management Amsterdam, PPA project) from 2013 [38]. Overall, the PPA project has three periods of construction. The aim of first stage is to improve the efficiency of A10 by ramp metering at motorway arterial (the A10-West) and a traffic light at one connecting urban arterial (the S102). The proposed ramp meters are shown in Figure 2.12. The second stage comes to intersection controllers at all the connecting urban arterials of A10-W. The final stage would expand the control to the A10-W network. The last two stages are more relevant to intersection control rather than ramp metering.

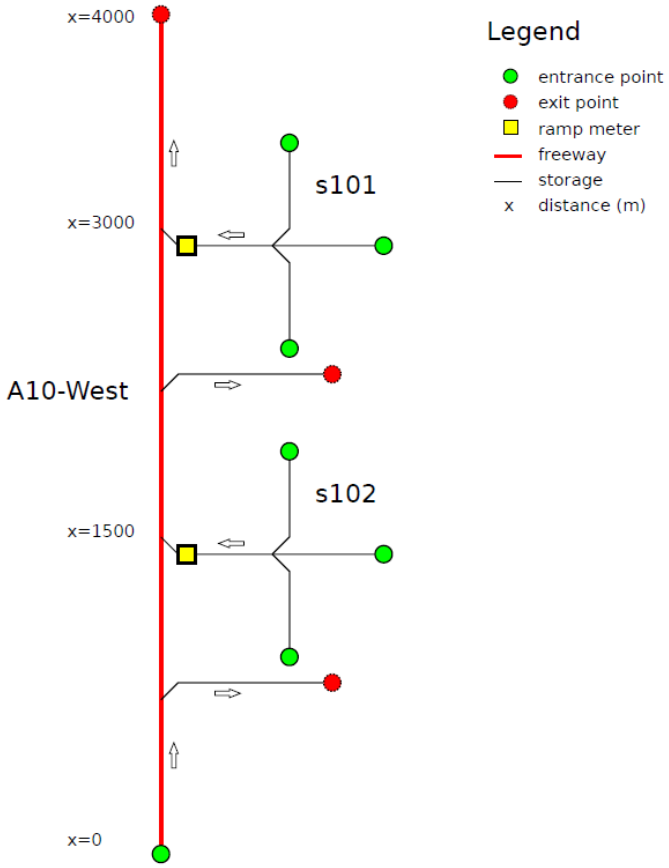


Figure 2.12: Test network showing A10 W(est) freeway, on-ramps s101 and s102 [38]

Apart from the individual development of different countries, Europe has also built up some international research about ramp metering. From 2008 to 2013, several European institutions are united to form a project “Network of Excellence for Advanced Road Cooperative Traffic Management in the Information Society” (NEARCTIS). NEARCTIS has funded 2.5 million euros in order to systemically summery the research of traffic management including ramp metering at institutions such as IFSTTAR, DLR, TU Delft and Imperial College of London. The final aim is to link them with the practical control [39].

Following the world step, New Zealand also developed its ramp metering system at the beginning of the 21st century. It set up a Sydney Coordinated Adaptive Traffic System (SCATS) Ramp Metering System consisting of three novel methods after the positive

feedback of metering at Mahunga Drive. The SCATS Ramp Metering System (SRMS) is constituted by 84 ramp metering installations on 5 motorways in Auckland, which form the most extensive ramp metering system in the southern hemisphere. The system increases 8 per cent of the motorway capacity reduce a quarter of congestion time. At the same time, the safety impact of ramp metering was researched. The evaluation group compare the number of incidents before and after the ramp metering at the same location and time period. Average 22.1% reduction was found at the 25 sites. Due to the positive effects of ramp metering, the SRMS in New Zealand keep developing for smooth and safe traffic [40].

2.5. CONTROL STRATEGY OF RAMP METERING

Control theory is a multipipeline idea combining engineering with mathematics, which mainly deals with the behavior of dynamical systems with input signals. The external input of the system is called "reference value". One or several variables in the system would change together with the reference value and the controller would process the input of the system so that the system output gets the expected value. The main idea of control theory is to maintain the system output by actuators at the reference value.

There are two kinds of control strategy in ramp metering called feed-forward control and feedback control. The structures of them are shown in Figure 2.13 and Figure 2.14.

Feed-forward control (also called open-loop control) is a kind of control only consider the measurements of sensors regardless the estimated measurements and the output. It is particularly suitable for the stable controller and process since it is able to maintain the stability of the system [41].

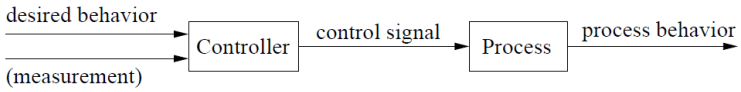


Figure 2.13: Structure of Feed-forward Control [41]

Typical ramp metering in Netherland follows the feed-forward control strategy which will be introduced in Section 2.6 in details. The loop detector is the sensors at the system and measures the traffic flow.

Although feed-forward control is quite easy to apply, it has the disadvantage that could only follow the default process and are not able to adapt the actions according to the situations. In order to overcome this limitation, feedback control (also called closed-loop control) is imported into the control theory. The closed-loop controller uses the feedback to control the state or output of the dynamic system. The actuators influence the outputs and measure them by sensing system. Then sensors proceed the measurement to the controller for processing as the return of the controller as one of the input

signals, becoming a closed loop. Another example of feedback control is cruise control in ADAS [42].

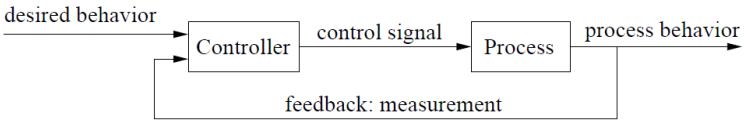


Figure 2.14: Structure of Feedback Control [41]

2.6. RAMP METERING ALGORITHMS

Ramp metering algorithms could be sorted into several types based on different classification methods. For instance, it could be clarified into 'single lane metering' and 'full lane metering' according to the number of operating lanes according to Middelham and Taale. They also mentioned two kinds of ramp metering which either close the on-ramp near the bottleneck or local metering to release congestion [7]. This thesis will only talk about the one with the local metering rate. There have been three control approaches in developing ramp metering strategies: pre-timed, local traffic responsive and coordinated traffic responsive according to the reaction of time [43]. Pre-time (also called Fixed time) strategy is first come up with by [44] to meter the on-ramp demand by the estimation from historical congestion data. However, it overlooks the real-time condition at the motorway mainline and is not able to change the metering rate according to the traffic situation. Therefore, it is soon out of date and the majority of ramp metering algorithms now are mainly local reactive ramp metering strategies.

by number of operating lanes	one lane full lane
by the extent	closed local time metering
by the reaction of time	fixed time local traffic response coordinated traffic response

Table 2.1: Different Classification of Ramp Metering

The following sections would illustrate the different types reactive strategies including demand – capacity (RWS) strategy, occupancy strategy, fuzzy logic strategy, ALINEA strategy.

2.6.1. DEMAND – CAPACITY (RWS) STRATEGY

Demand – Capacity (DC) strategy is a typical feed-forward strategy applied in North America. The structure of demand – capacity strategy is shown in Figure 2.15.

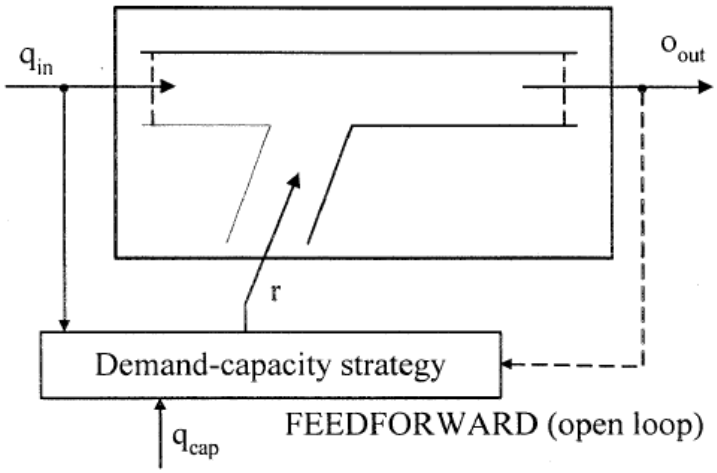


Figure 2.15: Layout of Demand - Capacity Strategy

There are overall two traffic states at DC strategy. When the free flow at upstream is close (but less than) the roadway capacity, the metering rate of each interval would be decided by comparing the measurement from loop detector with the capacity at downstream. If congestion already happens at the downstream, the metering rate would change to the automatically in order to solve or release the congestion. The formula reads as follow [45]:

$$r_k = \begin{cases} q_{cap} - q_{in}(k-1)\,,\ o_{out}(k) \leq o_{cr} \\ r_{min}\,,\ else \end{cases} \tag{2.13}$$

Where:

- q_{cap} is the free flow capacity at the downstream of the on-ramp
- q_{in} is the flow ($q_{measured}$) from loop detector
- o_{out} is the freeway occupancy at the downstream of the on-ramp
- o_{cr} is the critical occupancy at the downstream of the on-ramp
- r_{min} is the default minimum metering vehicle number

The majority of metering in the Netherlands follows similar feed-forward control (RWS strategy). It tries to make full use the existing infrastructure and achieve smooth operation of traffic. It could be seen in Figure 2.16 that the layout of RWS strategy is the same as it of DC strategy. However, it follows a bit different formula:

$$r_k = C - I_{k-1} \tag{2.14}$$

Where:

- r_k is the number of vehicles that are allowed to enter from on-ramp at moment k
- I_{k-1} is the measured flow of the loop detector at the upstream of the on-ramp
- C is the default capacity (according to researches) at the downstream of the on-ramp

$$t = \frac{n * 3600}{r_k} \tag{2.15}$$

Where t is the cycle time, n is the number of operating lanes.

Although DC (RWS) strategy could adapt the metering rate according to the real-time data, the measurement is only from the area before the metering measures [7].. It is still a feed-forward control which is quite unstable due to the non-measurable disturbances, especially on the merging area.

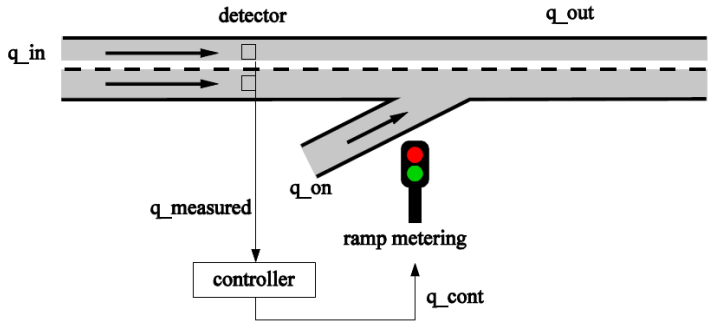


Figure 2.16: Layout of RWS Strategy [41]

2.6.2. OCCUPANCY STRATEGY

The occupancy (OCC) strategy is also an open-loop strategy which is quite sensitive to the disturbances. It would estimate the available flow of downstream by measuring occupancy. Due to the estimating flow q_{in} , ramp metering following this strategy might reduce the initial investment cost to some extent. A g-factor would be estimated through the historical empirical studies [46]. Also, a linear relationship between capacity and critical density is assumed in this strategy where $q_{in} = \frac{v_f * o_{in}}{g}$. The metering number become:

$$r(k) = K_1 - K_2 * o_{in}(k-1) \tag{2.16}$$

Where:

- v_f is the maximum speed in the free flow condition
- $r_k \subseteq [r_{min}, r_{max}]$
- $K_2 = q_{cap}$
- $K_2 = v_f / g$

2.6.3. FUZZY LOGIC STRATEGY

The fuzzy logic strategy is quite different comparing to other strategies. There are five levels of overall three input (speed upstream of on-ramp, speed downstream of on-ramp and the on-ramp queue time). The levels in the controller are set up based on the range such as relatively low, low, middle high and relatively high [47]. The output is cycle time of metering infrastructure which follows the rule 'IF speed upstream is medium AND speed downstream is low, THEN cycle time is long'. It is also the criteria of whether ramp metering operates. Only if the cycle time belongs to the defined range, will the metering light turn on [48].

2.6.4. ALINEA STRATEGY

Asservissement Linéaire d'Entrée Autoroutière (ALINEA) strategy was first to come up with by [49], following the feedback control. The main difference of the layout is the location of loop detector shown in Figure 2.17. It directly measures the flow of downstream bottleneck and uses it as the input of controller in the next step. It is more robust and reactive than the feed-forward control since the controller will react based on the previous results including unknown disturbances. Comparing with other strategies, ALINEA does not have any threshold for switching the metering infrastructure and is applicable to most of the traffic states. The adjustment of metering rate follows smooth variation, which is different from the mutation (on and off) of the previous methods. ALINEA algorithm tries to maintain the optimal traffic state on the mainline by metering at the on-ramp which follows adaption gradually to the metering rate at previous time step based on the subtraction between critical occupancy and downstream occupancy of the motorway.

The ALINEA reads:

$$r(k) = r(k-1) + K_R * [\hat{o} - o_{out}(k)] \tag{2.17}$$

Where:

- $K_R > 0$ is is a regulator parameter
- r_{k-1} is the measured ramp volume in the last time step *(Note it is not equal to the calculated ramp volume in the last time step)*

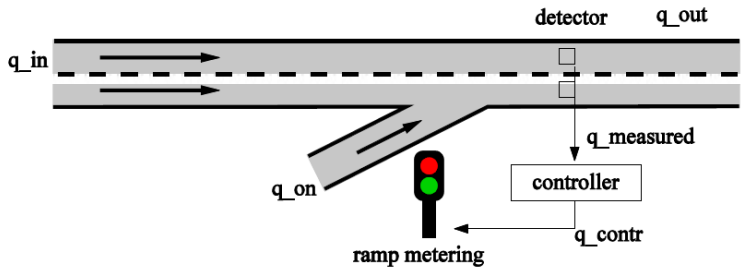


Figure 2.17: Layout of ALINEA Strategy [41]

Although the field test by [50] explains that $r(k)$ is not that sensitive to K_R, an optimal value of K_R = 70 vph was found under different circumstances.

Comparing RWS strategy with ALINEA strategy [51], the ALINEA could improve the motorway capacity, although it is difficult during the design and implementation. It is more applicable to the real world since vehicles do not always show the clam traffic behavior, especially during merging maneuver. Therefore, Rijikswaterstaat has a trail with PPA project starts from 2013 at Amsterdam area follows ALINEA strategy [52].

Measures	Control type	Characteristics
DC(RWS)	Feed forward	Simple for infrastructure, static behavior, less efficient
Occupancy	Feed forward	Least efficient
Fuzzy Logic	Feedback	Shorter metering time and longer cycle times, frequently switching on and off
ALINEA	Feedback	Efficient, complex

Table 2.2: Summery Table of Different Algorithms

2.7. SUMMARY

Chapter 2 forms the state-of-the-art of this research. Section 2.1 introduces some microscopic vehicle behavior models. It gives the background knowledge of vehicle motion modelling, including car-following models and lane change models. The selection on model will be introduced in Chapter 3 according to the characteristics of intelligent vehicles in the simulation. Section 2.2 then reviews some researches searching for the traffic impact of intelligent vehicles. Most of the researches simulate vehicle behaviors and form the conclusion. Various researches study the ACC vehicles' influence on macroscopic behaviors and while there are few researches about partial autonomous vehicles. Therefore the influence of partially autonomous vehicles to mixed traffic macroscopic

behaviors, i.e., sub-question (c), will be answered in Chapter 5. Section 2.3 to 2.6 explain the theories and development of ramp metering.

3

Design of the Simulation

This chapter will introduce the choice of simulation tool, simulation environment, simulation assumptions, scenarios and proposed performance indicators. It answers sub-question (b) in section 3.2.2:

- What are the microscopic driving models of partially autonomous vehicles and driver assistant vehicles?

3.1. CHOICE OF THE SIMULATION SOFTWARE

When the traffic flow reaches or is close to the highway capacity, congestion is likely to occur, especially when there are significant disturbances such as merging. In order to avoid the occurrence of downstream congestion, it is necessary to ensure that the sum of ramp demand and mainline demand entering the downstream is smaller than the downstream capacity. However, the appearance of level 2 autonomous vehicles might influence the performance of ramp metering due to its different lateral vehicle motion. At this moment, the microscopic simulation could be a wise choice due to the following reasons.

First of all, the simulation is an open environment which could have various inputs and outputs. It is flexible to change the inputs and outputs based on individual studies. In this thesis, OpenTrafficSim (OTS), as a single simulator combining all kinds of travel modes (private car, buses, trains, bicycles, pedestrians, airlines, etc.), will be extended with highway ramp metering. By setting up appropriate models of different modes, OTS could simulate possible events. It is a flexible way to reflect the real situation of various road and traffic conditions since it takes every single car as one of the essential elements in the traffic system. So that it is possible to show the behavior of each element per time step. This kind of system is also quite friendly to further researchers due to the easily changed input and output interface. Another significant benefit is the largely reducing time and costs. Furthermore, the result of the simulation could provide comparing data when there are filed tests. Comparing the results from simulation and field test, researchers might find out the deficiency of autonomous motion models and further develop existing models or create new models.

There are various simulation softwares in the market including VISSIM, AnyLogic and some specific-purpose traffic simulation softwares. Each software consists of a series of traffic flow theories and designs traffic simulation modules such as cars, ships, trains or pedestrians to achieve specific simulation purposes. However, it is challenging to develop a standard traffic simulation software that meets the needs of most users. These theories and modules sometimes may not meet the simulation requirements for individual researches. Especially when new technologies such as automation and their models develop, it is necessary to improve existing software or develop new software. The ideal simulation platform should be able to adopt a standard model, parameterize the model, and possibly to edit users own model. This open environment would allow parameters and models that are appropriate for local driver behavior, traffic rules, and environmental conditions to improve the accuracy of simulation.

The reasons why we choose OpenTrafficSim (OTS) are listed as follows.

Orientation OpenTrafficSim (OTS) has an objected-oriented simulation environment which simulates the movement of objects (cars, trains, pedestrians or ships). Each object contains data or attributes (velocity, direction, acceleration, etc.) and could interact with others. In general, those objects are divided into different classes to form complicated interaction and communication. Vehicle moves under the control of vehicle action models and the control of methods in the simulation environment. OTS could simulate vehicle movement as well as the transition of information flow.

Programming Programming function leaves space for users to customize the vehicle action models wherever it requires. Most simulation softwares in the market simulate automated vehicle behaviors based on fixed models that does not match preferable models of level 2 autonomous vehicle. It is important to simulate level 2 autonomous system which is still under development. This thesis contains a few assumptions about level 2 lateral motion model 3.4, which needs to be created in an open environment.

Precision It is important to check and adjust the parameters used in the model are suitable for the local conditions. This research is quite localized – Dutch traffic management environment. Fortunately, the lane change maneuver of level 1 vehicles is modeled by LMRS 2.1.2 in OTS, which calibrates its mathematical models by Dutch data. Therefore, the precision of the vehicle movement could be guaranteed.

Therefore the movement of the level 1 and 2 autonomous vehicle under the control of ramp metering at highway merging area will be simulated in OpenTrafficSim (OTS) based on their own vehicle motion models chosen in Section 3.2.2.

3.2. EXPERIMENT SET-UP

This section explains what the simulation elements are set up in this research, including the simulation environment, vehicles and ramp metering controllers. The simulation environment is fixed in this research while the participation of different vehicles (level 1 and level 2 autonomous vehicles), roadway demand and RWS controller are flexible. The variation of those three aspects generates the 12 scenarios in this research 3.2.4. Overall, each simulation in one scenario has 70 minutes simulation time.

3.2.1. SIMULATION ENVIRONMENT

The environment will show the network of motorway where runs the simulation. It is a primary merging area which has a two-lane motorway and an on-ramp shown in 3.1.

The stretch of the road network (from A to D) is 6 kilometers. The upstream of the motorway (Zone AB) is 3 kilometers long and the downstream of the motorway (Zone CD) is 2.75 kilometers long. The on-ramp acceleration section (Zone EF) is 0.75 kilometers long and length of the merging area (Zone BC) is 0.25 kilometers. Vehicles based on mainline demand are generated from point A and vehicles based on ramp demand are generated from point E. Those vehicles merge at Zone BC. Detectors are added at 0.1 kilometers at upstream and downstream of the merging area. The length of each detector is 1.8 meters. The output statistics from detectors are closely related to ramp metering strategy. The ramp metering installs at the end of the ramp lane, i.e., 250 meters before the merging area. It consists of loop detectors and traffic lights, detecting the speed and flow of detector 1-5. Besides, the width of each lane is set as 3.6 meters and the speed limit of motorway is 120 km/h.

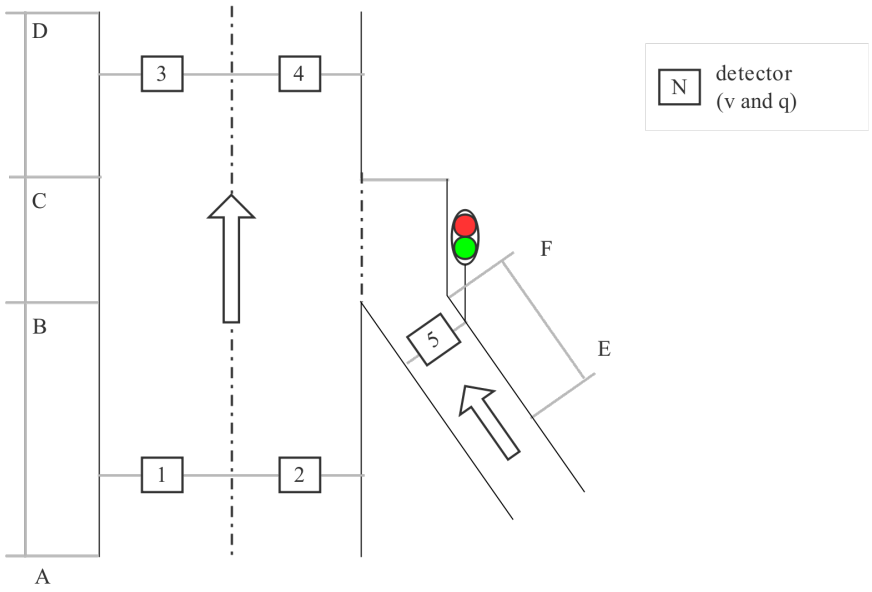


Figure 3.1: Layout of the Environment

In this two-lane motorway with an on-ramp, vehicles in zone AB and EF will drive complying with the rules of a car-following model. Vehicles from the on-ramp and mainline will merge at zone BC under the control of lane change model and then move to zone CD.

3.2.2. VEHICLES IN THE SIMULATION

According to Section 1.4, there will be two types of vehicles in the simulation, including level 1 and level 2 autonomous vehicles. This section will distinguish the tasks of drivers and systems according to the intelligence of autonomous systems.

CLASS: LEVEL 1 AUTONOMOUS VEHICLES

Attribute:

- Attributes: Speed, Acceleration, Directions
- Longitudinal Motion Model: IDM+
- Lateral Motion Model: LMRS

Behavior:

- Generate either from the start of motorway or on-ramp;
- For vehicles generating from motorway: Operate from A to D and change lanes based on willingness and politeness;
- For vehicles generating from on-ramp: Operate from E to F and judge their behavior by ramp metering installation (stop when the traffic light is red) monetarily merge at zone BC;
- For vehicles generating from on-ramp: Operate from F to D and monetarily merge at zone BC.

According to SAE International Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles shown in 3.2 [3], the system function of level 1 automation is defined as follows: *For purposes of DDT performance, level 1 encompasses automation of either lateral vehicle motion control functionality or longitudinal vehicle motion control functionality and limited OEDR associated with the given axis of vehicle motion control.* In this simulation, the level 1 autonomous vehicles have autonomous longitudinal motion control, i.e., equipped Adaptive Cruise Control (ACC) car following system. To be specific, it is an autonomous car following system which allows a vehicle to follow its predecessor at a particular time gap by automatically control the engine, power train or brakes [53]. On the other hand, drivers in level 1 autonomous vehicles should have the responsibility to strategical (destination and route choice), tactical (lane choice) and operational planning (steering). Therefore, the level 1 autonomous vehicles in this simulation have the following features.

- Control the Longitudinal Positions of Vehicles: It could perceive the driving environment by sensing the relative speed of neighbors and further control the acceleration of the ego vehicle. There is another sensor on the pedal to test the force applied by the driver. The system maintains the vehicle speed according to the force applied to the sensors to reduce the driving tasks of drivers.

- Assist Drivers by Following the Predecessor: The system could be set to track the predecessor and adjust the vehicle speed according to the predecessor. It will try

to maintain a safe distance, which can be selected by the setting slot on the control lever near the steering wheel.

- Change Lanes with Human Perception Errors: There are a perception layer for human drivers to perceiving the driving environment. They change lanes according to their desire and perceived information such as neighboring gap, speed limit, road condition and so on. However, it is natural for humans to make mistakes, especially during the perceiving process. They may estimate some inaccurate information about infrastructure and neighbors [54].

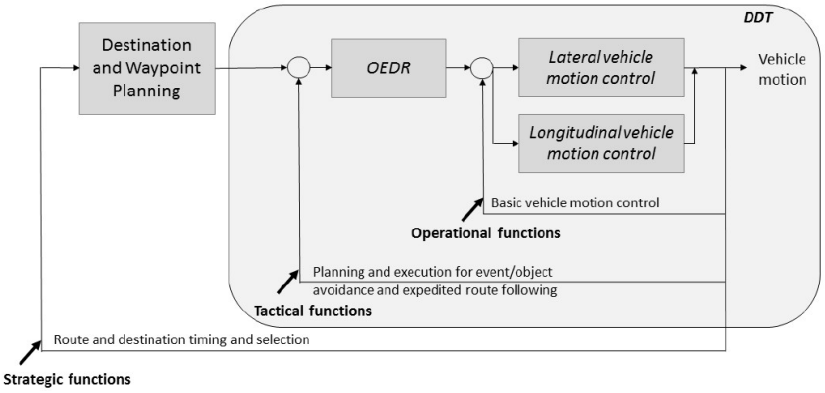


Figure 3.2: Schematic view of driving task showing DDT portion

The behaviors of level 1 autonomous vehicle are modeled by longitudinal and lateral motion models, which have been explained in Section 2.1. This simulation requires fast reaction of to the coming vehicles from the bottleneck which is present clearly in stimulus-response branch in car following models. In stimulus-response branch, the intelligent Driver Model (IDM) has been proved valid to model the longitudinal behavior of autonomous vehicles[55]. Based on the above features, the longitudinal motion model for level 1 autonomous vehicles is set as IDM+ [56] in this simulation. IDM+ is an adaptation vision of the Intelligent Driver Model (IDM) which adapted its parameters according to realistic capacities and stability patterns.

On the other hand, the lateral motion models are expended with more details over time. One of the lastest model is called Lane change Model with Relaxation and Synchronisation (LMRS). The IDM+ is integrated with this lateral motion model considering politeness of the surrounding drivers. It allows similar cut-ins with the reality by relaxation and synchronisation, while other models may reject some gaps or unreasonable acceptance thresholds. The detailed parameters in these models are shown in Table 3.1. Besides, there will be a perception layer for level 1 vehicles to simulate the estimation

errors of neighbor headway, speed and acceleration due to human nature. Comparing the perceived gap and minimum lane-change headway, drivers could decide whether to change lanes or not.

CLASS: LEVEL 2 AUTONOMOUS VEHICLES

Attribute:

- Attributes: Speed, Acceleration, Directions
- Longitudinal Motion Model: IDM+
- Lateral Motion Model: Modified LMRS with Shared Control

Behavior:

- Generate either from the start of motorway or on-ramp;
- For vehicles generating from motorway: Operate from A to D and change lanes based on willingness and politeness;
- For vehicles generating from on-ramp: Operate from E to F and judge their behavior by ramp metering installation (stop when the traffic light is red) monetarily merge at zone BC;
- For vehicles generating from on-ramp: Operate from F to D and monetarily merge at zone BC.

In terms of level 2 autonomous vehicles, the definition in [3] is *level 2 encompasses the automation of lateral and longitudinal vehicle motion control and limited OEDR associated with vehicle motion control.* The level 2 vehicles in the simulation are equipped with Adaptive Cruise Control (ACC) system and Lane Change Assistance (LCA) system. The LCA system in this simulation is a kind of shared control system in which human drivers will decide whether to change lane or not and system finishes the maneuver by detecting neighboring behaviors, acceleration/deceleration and steering.

Since there is the same ACC system in level 1 and 2 autonomous vehicles, they have the same features on longitudinal behaviors (Assist Drivers by Control the Longitudinal Positions of Vehicles; Assist Drivers by Following the Predecessor). Therefore the longitudinal behavior of level 2 vehicles is also modeled by IDM+.

Besides, the shared LCA system has the following features.

- Blending/Mix Input Shared Control: The LCA system has a shared control scheme where drivers generate their lane-change desire based on their strategical (or tactical) planning. Then they request the LCA system to change lanes regardless of the value of desire. At the same time, the LCA system receives the request and detects the gap in target lanes. It changes lanes if the gap is larger (or equals to the

minimum lane-change headway. Otherwise, it waits for enough gap with the synchronization and cooperation of other vehicles until drivers cancel the requests.

- Change Lanes with Machine Detection Errors: The LCA system changes lanes according to drivers' desire and detector errors. During the lane change maneuver, the LCA system will perceive the environment by vehicle detectors. According to the empirical data from [57], the standard error of the position is 13.1 cm and its standard deviation is 8.2 cm. Besides, the standard deviation of the speed is 0.3 cm/s.

- Priority of Human Drivers: Human drivers can always overrule the LCA system to guarantee the safe lane-change maneuver. Level 2 automation still requires the supervision of human drivers during the system switching on. Nowadays, level 2 automation is an area in development. To avoid the machine failing, human drivers can always control the steering wheel and make correction for the LCA system.

The lateral behaviors of level 2 vehicles is modeled by modified LMRS. The desire to change lane will decrease from 4 (no lane-change, free lane-change, synchronized lane-change and cooperative lane-change) to 2 levels (no lane-change and requested lane-change). The LCA system receives the lane change desire and changes from no lane-change level to requested lane-change level. Then it compares the neighbour gap with the minimum lane-change headway and decides whether to change lane immediately. If the gap is less than the minimum lane-change headway, lane change will not perform instantly. The system will synchronize after 1s and then request cooperation 2s later.

Parameter	level 1	level 2
Maximum Speed/[km/hr]	120	120
Car Following Headway/[s]	1.8	1.8
Maximum Comfortable Deceleration/[m/s^2]	2.0	2.0
Maximum Adjustment Deceleration/[m/s^2]	0.5	0.5
Maximum Desired Car-following Acceleration/[m/s^2]]	2.0	2.0
Minimum Lane-change Headway/[s]	0.56	**0.7**
Synchronisation Time/[s]	/	1.0
Cooperation Time/[s]	/	2.0

Table 3.1: Experimental Parameters of Level 1 and 2 Vehicles

The car following headway in both two types of vehicles is chosen according to the autopilot report of car manufactories which is different from default setting (1.2 seconds) of LMRS. Due to safety concerns, manufactories sell ACC vehicles with 1.8-second

car following headway nowadays [58]. Therefore, simulation in this thesis also choose 1.8 seconds considering the real-world situation.

The minimum lane-change headway is chosen as 0.7 seconds owing to safety concerns manufactories as well. It is an assumption of the simulation but there will be a sensitivity analysis in Section 4.1.2 to test its uncertain for maximum capacity.

3.2.3. RAMP METERING CONTROLLER

Ramp metering controllers are set up to prevent or delay congestion. The literature survey shows that currently there are two kinds of ramp metering controllers (RWS and ALINEA) in the Netherlands. However, since 96% of the ramp metering installation follows the RWS algorithms, this simulation will only apply RWS algorithms in the ramp metering installations. Subsection 2.6 only shows the calculation of metering vehicle number and overall cycling time. This section will explain how the controller works.

Figure 3.3 shows the logic of RWS controller. It is initialized at a certain time 1 and starts to get the information from loop detectors at upstream (1 and 2) in Figure 3.1. The controller checks the average speed or flow of detector 1 and 2 every minute. If the average speed is less than 70 km/h or the flow larger than critical flow (3000 veh/h), the installation for ramp metering switches on and works according to RWS formulas. The capacity of motorway used in the formulas is set up as 4000 veh/h. The ramp installation first turns green and then (yellow) red according to the calculated time. It repeats the cycling process until the information from loop detectors updates. Once the installation starts, only if the average speed is larger than 70 km/h and the flow is less than critical flow (3000 veh/h) would it switch off.

3.2.4. SIMULATION SCENARIOS

There are 3 inputs in different scenarios. They are the penetration rate of level 2 autonomous vehicles, the participation of ramp metering installations and demand of motorway (on-ramp).

PENETRATION RATE OF TWO TYPES OF VEHICLES

Level 1 automated vehicles has entered the market now and increase gradually. The basic scenario is defined as scenario with level 1 autonomous vehicles only shown in Table 3.2 and 3.3. It provides a baseline of the simulation output. According to Calvert, Schakel, and van Lint [9], level 2 automated vehicles are supposed to join the traffic at around 2020 and rise up to around 20% share on road at 2035. Since the estimation of vehicle market share may have some errors, this simulation lists 5 types of penetration rate of level 2 autonomous vehicles. It increases from 10% to 50% gradually, with the interval of 10%.

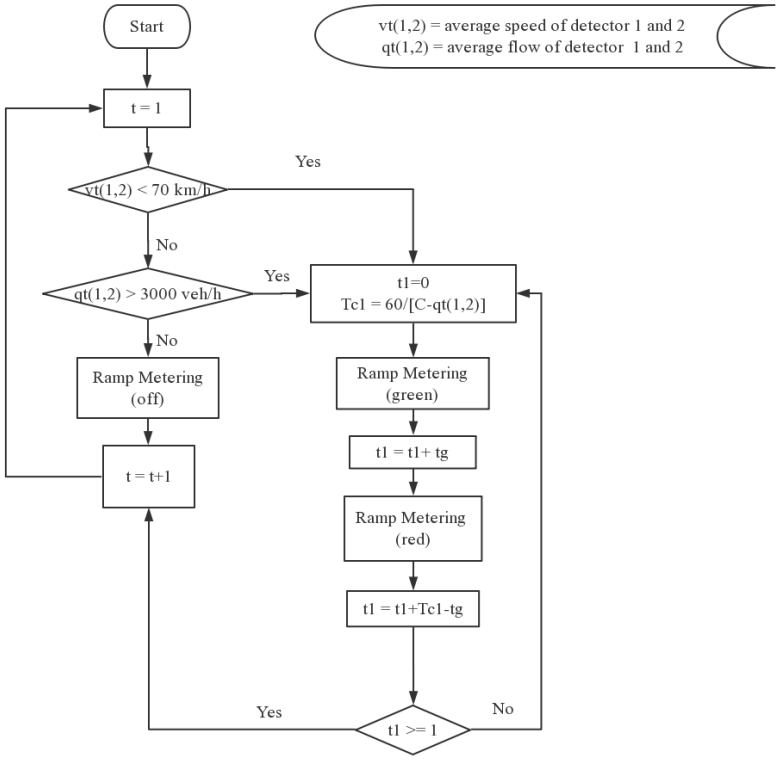


Figure 3.3: Ramp Metering Controller - Following RWS Algorithm

PARTICIPATION OF RAMP METERING INSTALLATIONS

As we am going to evaluate the influence of intelligent vehicles on the effectiveness of ramp metering, all penetration rates are simulated with and without ramp metering. The name of the scenarios is set according to the penetration rate of level 2 autonomous vehicles and whether this scenario has ramp metering installations or not. The designed scenarios are shown in the following tables.

	Base	Low Penetration		Medium Penetration		
Scenario	**0%**	10%	20%	30%	40%	50%
Level 1	**100%**	90%	80%	70%	60%	50%
Level 2	**0%**	10%	20%	30%	40%	50%

Table 3.2: RWS Penetration Scenarios without Ramp Metering

	Base	Low Penetration		Medium Penetration		
Scenario	**RM-0%**	RM-10%	RM-20%	RM-30%	RM-40%	RM-50%
Level 1	**100%**	90%	80%	70%	60%	50%
Level 2	**0%**	10%	20%	30%	40%	50%

Table 3.3: RWS Penetration Scenarios with Ramp Metering

DEMAND

The choice of top demand is essential for the observed flow of motorway. It should have enough flows at the working moment of ramp metering installations. At the same time, the demand should small enough to avoid congestion since ramp metering loses its efficiency once the congestion begins. To meet this requirement, we did several testing without ramp metering and found there is a specific range of 'stop and go' traffic when the mainline demand is 3400 veh/h and ramp demand is 500 veh/h in basic scenarios.

Therefore, the demand for the mainline motorway and on-ramp in Scenario (RM-)0% to (RM-)100% are imported according to Figure 3.4. At the beginning of the simulation runs, there will be a warm-up period for the vehicle generation and smooth operation. The main demand increase from 1000 veh/h to 2200 veh/h within the first 10 minutes. Then there will be a is a slowly increasing demand from 10 min to 40 min at motorway. It is designed for precise observation for the efficiency of ramp metering. The top demand of the motorway is set up as 3400 veh/h and lasts 10 minutes. The on-ramp demand is 500 veh/h and decreases to 100 veh/h during the last 20 minutes of the simulation.

However, due to the reducing capacity at high penetration rates of level 2 autonomous vehicles (i.e., scenario 70% to 100%) explained in Section 5.1.2, congestion appears at the beginning of the motorway (close to Point A) when the mainline demand is close to 3000 veh/h. A sensitivity analysis is done to determine the demand (from 2000 veh/h to 4000 veh/h) in scenario 70%-100%. There is not a top demand lead to congestion at merging area in scenario 0% without ramp metering and will not cause congestion at the beginning of the motorway in scenario 70% to 100% as well. Therefore, there will be another demand graph for high penetration scenarios with the top demand of 2700 veh/h. Other changes of demand are the same as demand in scenario 0% to 100% shown in Figure **??**.

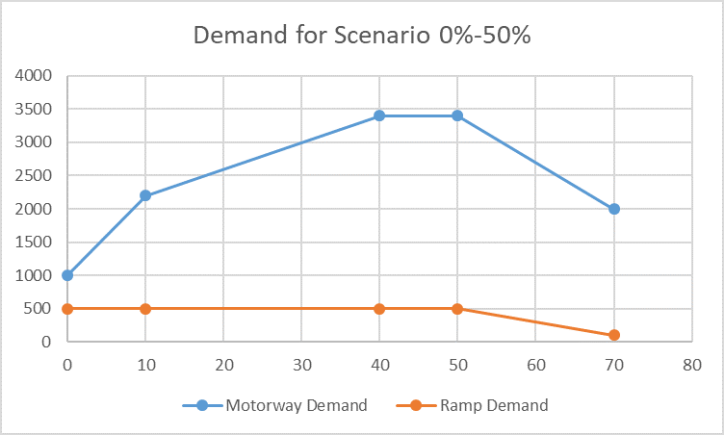


Figure 3.4: Demand of Motorway and Ramp in Scenario 0%-50%

3.3. NUMBER OF RUNS

There will be a random seed for the vehicle generation to initialize the pseudorandom and it will influence traffic behavior and performance. However, the variation based on seed might influence the outputs a lot so that it leads to a weak conclusion. To eliminate the influence of the random seed, seven simulations of the base scenario are performed. The Total Time Spent (TTS) is shown in the following Table and varies from 201 to 313 hours.

Seed	1	2	3	4	5	6	7
TTS [h]	248	264	201	272	300	261	313

Table 3.4: Total Time Spent (TTS) - Basic Scenario (Seed 1-7)

Therefore, a one-step approach [59] is used to calculate the number of simulation runs that should be carried out to ensure that the average value of travel time of each sample in one run within the desired confidence interval.

$$n = (\sigma * Z_{\alpha/2})^2 / d^2 \tag{3.1}$$

where,

n is the number of runs

$Z_{\alpha/2}$ is the two-tailed normal statistic for a level of confidence $1 - \alpha$

σ is the standard deviation

d is the accuracy defined

σ in this case will be estimated according to the short pilot experiments. It is estimated according to the difference of lowest and highest individual travel time (usually 4σ). The average travel time of a vehicle varies from 3.17 minutes to 6.55 minutes. Thus σ is estimated as 0.845. The confidence interval is 95% and accuracy equals to 0.5 minutes. The above formula then becomes:

$$n = (0.845 * 1.96)^2 / 0.5^2 = 11 \tag{3.2}$$

3.4. SIMULATION ASSUMPTIONS AND SIMPLIFICATIONS

This section explains the assumptions and simplifications of autonomous vehicles in the simulation. Due to the developing autonomous technique, not all the behaviors of autonomous vehicles can be modeled. The following assumptions are made according to literature or manufacturer reports. Simplifications are made for the smooth implementation of OTS.

1. All Level 1 autonomous vehicles will turn on their system and follow the instruction of ACC system longitudinally. Laterally, drivers estimate and detect the related objects according to their driving experience. However, human beings always make mistakes due to the limitation of eyesight, attentionfatigue or emotion. Therefore the perceiving positions of infrastructure and neighbors, traffic rules such as speed limit, speed and velocity of neighbors have perception error at the perception layer.

2. The error term of the detectors in LCA system introduced in 3.2.2 will be ignored since their unit is cm and the numbers are quite small compared with the positioning at the simulator.

3. The Lane Change Assistance (LCA) system is assumed to be a mature automation system and will not lead to dangerous lane change. It is a shared system where human drivers generate lane change desire. Then the system will finish the lane-change maneuver automatically. Although there is a rules that human can overrule the LCA system, the system are assumed not to lead incidents that require the taking-over of human drivers.

4. The minimum lane-change headway used in level 2 autonomous vehicles is set at higher value comparing with level 1 autonomous vehicles to make sure the safety of the LCA system. However, this may lead to the difference of simulation results. Therefore, there will be a sensitivity analysis about different minimum lane-change headway.

5. The speed limit of the motorway is set up as 120 km/h and it of the on-ramp is set up as 70 km/h.

6. The vehicle length of level 1 and level 2 autonomous vehicles are all set as 4.19 meters.

7. The ramp metering installation has some requirements. The maximum cycle time of on-ramp should be less than 15 seconds [60]. This setting aims to avoid too large de-

lays at the on-ramp. There might be a critical situation that the flow of motorway is close to its capacity. At that time, the optimum result prevents any disturbance of on-ramp to avoid congestion at the macroscopic level. However, vehicles from on-ramp cannot wait until the flow drops. As a ramp controller, the metering installation should consider the benefit of both motorway and on-ramp. Therefore, the maximum cycle time is set in the simulation.

The following simplifications are made for the more straightforward implementations in OTS.

1. The detector of ramp metering is slightly different from reality. There are two detectors in Dutch ramp metering. The first loop detector is behind the stop line and the signal to turn into yellow after vehicles passing by. Another detector locates after the first one and the signal will turn red after vehicle passing by. In this simulation, the only detector locates at 250m before the merging area and will turn red if vehicles pass. Since yellow times are not relevant for the process of ramp metering, it is included only if the total yellow plus red time is sufficiently long. Otherwise only red light is used and the minimum red time is set as 2 seconds.

The simplification may make different yellow time comparing the simulation with reality. However, it would not influence the simulation result (including fundamental diagram, total delays or total travel time) since these two types of detector setting all obey a strategy called 'One car per green', which means only one vehicle could pass during the green light at the on-ramp. The total cycle time and controlled flow at on-ramp would not be different in these two types and therefore does not influence the macroscopic behaviors.

2. The simulation ignores human interaction when the system is switching on. Currently, human drivers have the authority to overrule the autonomous system, i.e., vehicles would listen to human drivers rather than autonomous system. This rule is designed by manufactory since they are not confident enough about their autonomous system. Human drivers still have the responsibility to overrule the system when they see some inappropriate behaviors. However, human drivers may make perception mistakes and then overrule the system. Those interactions are challenging to predict therefore those human interactions during the system working period are ignored in this thesis.

3.5. PERFORMANCE INDICATORS

There will be four outputs based on each scenario. They are the mean speed of the motorway, capacity drop and total time spent (total delays). Among them, the mean speed, total time spent and total delays will be processed in Excel. The estimation of congestion will be done in MATLAB.

- Mean Speed of Motorway (v)

The mean speed of the motorway at each minute clearly shows the states of the traffic. It is measured at upstream motorway at detectors 1 and 2 which locate at 0.1 km before the merging area BC every 2 minutes. Theoretically, detectors measure the time from the entering to leaving the detecting area of all passing vehicles shown in the following function.

$$v = \frac{\sum (L_{detector} + L_{vehicle}) / T_{measured}}{N} \tag{3.3}$$

$L_{detector}$ is detector length

$L_{vehicle}$ is vehicle length

$T_{measured}$ is the time duration from the vehicle entering to leaving the detector

N is the number of passing vehicles in an aggregate period

However, OTS generates vehicles with their unique characteristics such as vehicle length, speed, acceleration and updates these characteristics during the simulation. Therefore detectors directly obtain vehicle speeds regardless of vehicle and detector length. The function is also simplified.

$$v = \sum V_i / N \tag{3.4}$$

In this thesis, the congestion speed is defined as less than 60 km/h. Since the main disturbance is merging behavior at the on-ramp, the congestion is more likely to happen at the on-ramp and on-ramp adjacent lane, resulting in the decreasing mean speed. Therefore, the mean speed is regarded as a vital output to judge traffic efficiency of the motorway. In this thesis, congested traffic is defined as traffic with mean speed under 60 km/h.

- Capacity

Since the ramp metering aims to postpone the congestion due to the large disturbance, one of the important outputs is the capacity drop. There are various definitions of capacity. The most well-known one is *defined as the maximum flow rate that can reasonably be expected to traverse a uniform segment of road under prevailing roadway, traffic and control conditions [61].* However, the simulation runs with some stochastic events which sometimes are not able to show the maximum flow. Thereby stochastic capacity is defined according to Lorenz and Elefteriadou [62]. *the rate of flow along a uniform freeway segment corresponding to the expected probability of breakdown deemed acceptable under prevailing traffic and roadway conditions in a specific direction.* Another vital parameter for capacity drop is discharge capacity, which is the maximum flow when the traffic is during the congestion period. Similarly, there is also a stochastic discharge capacity in many simulations shown in Figure 3.5 [63]. In this thesis, both the capacity drop (CD)

and the stochastic capacity drop (SCD) will be discussed.

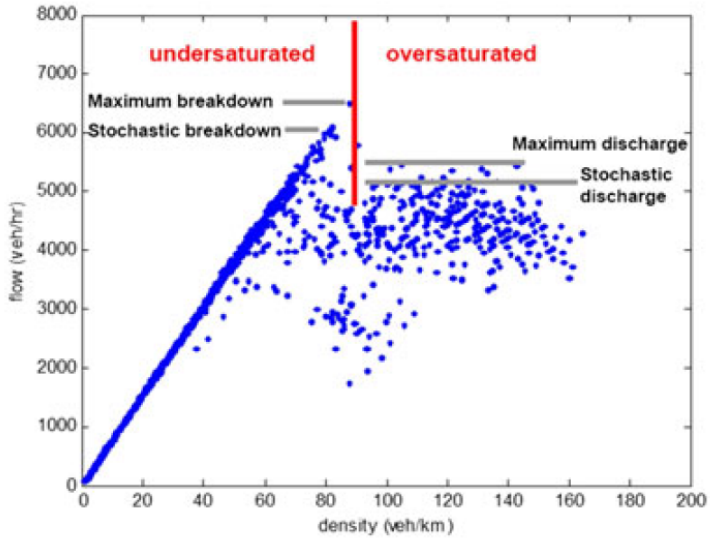


Figure 3.5: Definition of Capacity on Fundamental Diagram [63]

$$CD = \max q_(v_i) - \max q_(v_j) \tag{3.5}$$

$q_(v)$ is the observed flow when speed = v_i or v_j

$v_i \in v > 60$ km/h

$v_j \in v < 60$ km/h

Many methods have been developed for the calculation of stochastic capacity drop such as Empirical distribution method, Product limit method, FOISM method, Simple estimation method, Fundamental diagram method and so on. According to Stanescu [64], the parameterized Product Limit Method (PLM) introduced by Brilon [65] is the most accurate estimation method for Dutch motorways. Since the target of this thesis is Dutch ramp metering, the stochastic capacity drop in the simulation will be calculated through PLM.

The product limit method estimates (stochastic) discharge capacity which requires congested flow (speed is less than 60km/h) in all seeds and look for its cumulative probability curves afterwards. However, the vehicles at downstream have passed the bottleneck congestion and accelerated again. The congested flow then cannot be separated from downstream observations. Therefore, this thesis modifies PLM which differential the free flow and congested flow downstream according to the upstream speed. When

the average speed of upstream is less than 60 km/h, the flow at downstream is considered congested.

It is found a good time interval to gather the information of flow is 5 minutes. However, the overall simulation is 70 minutes and it will not have enough data if the flow is gathered every 5 minutes in 11 runs. Therefore, the time interval in this simulation is chosen as 2 minutes. The free flow and congested flow are separated based on the corresponding measured mean speed. Both of them are assumed to distribute according to a standard probability distribution (Weibull distribution) based on the following function. Parameters α and β fit according to the cumulative probability distribution of observed free flow and congested flow.

$$F(x) = 1 - e^{-}(\frac{x}{\beta})^{\alpha} \tag{3.6}$$

α is shape parameter

β is scale parameter

Two cumulative probability curves will be drawn by MATLAB in each scenario, calculating the stochastic breakdown capacity and stochastic discharge capacity when the cumulative probability is 0.5. Their difference is the stochastic capacity drop (SCD).

- Total Time Spent (TTS)

Total Time Spent (TTS) is the sum of travel time of all vehicles from its origin (A or E) to its destination (D) in the simulation. It indicates the traffic fluency of this network.

$$TTS = \sum (t_i^D - t_i^{Origin}) \tag{3.7}$$

- Total Delays (D)

Delays for a vehicle are the extra time it needs to be compared to the free flow travel time. The free flow travel time is defined as total distance travelled divided by speed limit. It reflects the delays due to the congestion.

$$D = \int (TTS_i - TTS_{freeflow} di) \tag{3.8}$$

3.6. SUMMARY

This chapter summarizes the simulation consisting of traffic simulation, ramp metering control and data processing. Traffic simulation models the behaviors of every single vehicle. The ramp metering is a traffic signal that macroscopically guides some behaviors of vehicles. These two steps happen in OTS based on different scenarios and produce

some outputs. Finally the data processing procedure is done in Excel and Matlab according to the simulation output and functions introduced in section 3.5.

4

VERIFICATION AND VALIDATION

4.1. VERIFICATION

Verification is a process that tests the reliability of the simulation results from different perspectives. First of all, the basic elements (level 1 and 2 vehicles) runs well in the simulation. Other verification will base on the effectiveness of ramp metering and sensitivity analysis.

4.1.1. EFFECTIVENESS OF RAMP METERING

Before running of different scenario, it is essential to verify the correctness of simulation. Specifically, the effectiveness of ramp metering should be checked. The ramp metering installation is supposed to work efficiently in the basic scenario. Otherwise there might be bugs or incorrect behaviors in the simulation. Therefore, the effectiveness of ramp metering is tested based on the capacity of ramp metering and no ramp metering situations. Although the RWS controller and ALINEA controller have different ways to calculate the allowed demand at the on-ramp, they follow similar procedures during ramp metering. Therefore, the test will only take ramp metering following RWS algorithm as an example.

As mentioned before, the basic scenario of RWS algorithm is 100% of level 1 vehicles based on the demand of Figure 3.4. It will be compared with simulation (scenario 0%), which 100% of level 1 vehicles run according to the demand graph without ramp metering installations. The following table shows the congestion period of the motorway scenario RM-0% and 0% according to the period that means speed is less than 60 km/h.

Scenario	RM-0%	0%
Congestion Duration [min]	10.2	29.6
Overall Simulation Time [min]	70	70
Percentage of congestion	14.6%	42.3%

Table 4.1: Congestion Period of Scenario RM-0% and 0%

A motorway with ramp metering will have larger average speed comparing with motorway without ramp metering. The difference is caused by the congestion of the merging area. The vehicles in Scenario 0% confront with the congestion at merging area and accelerate again, especially at the adjacent lane of on-ramp. Most vehicles pass the detectors 2 (on the right lane of the motorway) with lower speed. Scenario RM-0% and 0% have significantly different time of congestion. Ramp metering reduces 27.7% congestion and increases traffic efficiency.

Another essential way to verify the output is simulation animation which visualizes the operation of vehicles on the road network. The flow density diagram represents statistical information such as capacity or speed. What's more, the animation can show the traffic performance in a visualized manner. For example, it is clear to find which part

of the motorway generates severe congestion during the simulation experiments. The strange behaviors of vehicles could also be observed directly from animation.

From the animation of the non-ramp metering scenario (0%), small congestion happens at merging area from minute 4 shown in Figure 4.1 but soon releases at minute 7. More substantial congestion also begins at merging area from minute 22 and remain as 'stop and go' traffic till the end of the simulation.

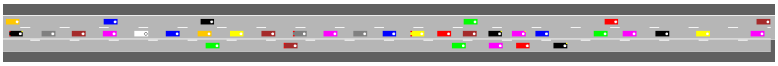


Figure 4.1: The Animation of the 4th minute (Scenario 0%)

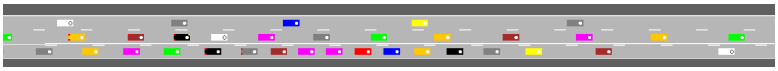


Figure 4.2: The Animation of the 22nd minute (Scenario 0%)

To the opposite, the basic scenario works well during the simulation and no congestion appear. Vehicles operate at a stable speed at merging area and go to the end of the motorway. Based on the comparison of flow, average speed and animation, it is proved that RWS ramp metering is quite effective.

4.1.2. SENSITIVITY ANALYSIS

Sensitivity analysis is to ensure that the outcome of the model does not deviate from the expected value due to the uncertainty of the inputs. As mentioned before, the minimum accepted gap (also called minimum lane change headway) of level 2 automation is a parameter strongly depended on the manufacture of level 2 autonomous system. The primary simulation uses a larger one (0.7s) for level 2 autonomous vehicles. In this part, another two minimum accepted gaps are tested in Medium Penetration (Scenario 50%). One of them (0.56s) is the same as the minimum accepted gap in level 1 autonomous vehicles [28]. Although the other minimum accepted gap is 0.4s which is quite small, it is assumed to have the safe lane changing maneuver in this scenario. The maximum breakdown capacity separates from the flow data is shown in the following table.

Minimum Lane Change Headway [s]	0.7	0.56	0.4
Maximum Breakdown Capacity [veh/h]	4410	4460	4390

Table 4.2: Capacity of Different Minimum Lane Change Headway - Scenario RM-50% (seed=1-11)

The capacity of different minimum lane change headway varies from 4390 veh/h to 4460 veh/h, which is 1.6%. Thus it is not sensitive when the minimum lane change gap

is less than 0.8 seconds. However, tests with minimum lane change headway larger than 0.8 seconds (such as 1.0 seconds or 1.5 seconds) found that severe congestion appears at the on-ramp. It is because that the larger minimum lane change headway makes it difficult for vehicles to find enough gap and merge from the on-ramp. As the cooperation of other vehicles starts after 2 seconds of the lane change desire, the waiting time for a minimum lane change gap is high. At that time, the ramp metering does not switch on as the total demand of motorway and on-ramp is less than the motorway capacity. Congestion at the merging area propagate to the end of the on-ramp and leads to the error of the simulation.

4.2. VALIDATION OF MODEL SETUP

Validation of the model setup is the task of demonstrating that the simulation model is a reasonable representation of the actual motorway traffic, i.e., whether the simulation generates the correct traffic behaviors in the motorway and whether these behaviors are fidelity enough for theoretical analysis. In this section, the fundamental diagram of basic scenario (seed = 1 without ramp metering) will be validated.

Fundamental Diagram analyzes the feature of traffic and has been used for searching the capacity of discontinuities in Dutch motorways [66]. There are various shapes of fundamental diagrams such as Triangular, Inverse Lambda, Smulders, Truncated Triangular, Wu and Greenshields. Since the traffic breaks down at the merging bottleneck, the shape of the fundamental diagram is more likely to be Wu shape theoretically.

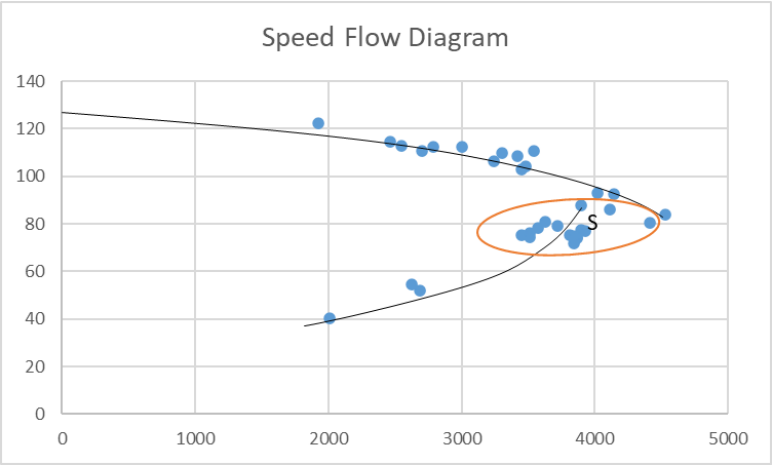


Figure 4.3: Speed Flow Diagram (Scenario 0%)

Figure 4.3 shows the simulated fundamental diagram of the basic scenario without ramp metering installation (scenario 0%) at downstream bottleneck. Generally, it meets the shape of Wu's fundamental diagram, especially during the free flow period. Theo-

retically, the flow should appear according to the grey slant when it comes to the congested flow. However, it appears more stochastic shown in the orange cycle. This flow is called synchronized flow which varies significantly in synchronous streams. The downstream interface of the synchronous stream is usually maintained at the bottleneck, i.e., the merging area in this case. Compared with free flow, the speed of vehicles in synchronous flow will be significantly reduced, but traffic flow can be maintained roughly at the same level as free flow. It could also be less than free flow due to the stochastic of simulation model. If the demand continues to increase, the synchronized flow might change to wide moving jam which results in largely reducing speed and flow. In the basic scenario, there is no wide moving jam since the demand is controlled.

Basic scenario with ramp metering will not be validated here since it has few congested flows which have been shown in Table 4.1 and does not have the complete characteristics of fundamental diagram.

4.3. SUMMARY

This chapter verifies and validates the setup model of simulation. In the verification section, the modelled behaviors of vehicles are found similar to real-world situations. The ramp metering installation works efficiently in basic scenario. Meanwhile, the sensitivity of minimum lane change gap of level 2 autonomous vehicle is tested. The maximum capacity remains at the same level with some variation of the minimum lane change gap. The fundamental diagram is drawn to analyse whether the traffic behaves like the real situations in the validation part. The free flow matches perfectly with the theoretical results while the congested flow to some extent shows some stochastic. However, it is common to have some synchronized flow in simulations which also has the same trend with theoretical results. Therefore I trust the simulation runs well and the outputs will be further processed in the next chapter.

5

Simulation Results and Discussion

In the previous chapter the simulation set-up is structured. This chapter will provide explanation on the model outputs which is the average results of 11 runs per scenario. The following sub-questions will be answered in this chapter.

- How will partial autonomous vehicles influence the macroscopic behaviors of mixed traffic under different penetration rate?
- How will current measure selected in sub-question (a) influence the macroscopic behaviors of mixed traffic?

5.1. Simulation Results

5.1.1. Mean Speed

Statistics of speed is gathered from loop detectors 1 and 2 at motorway upstream. The mean speed is the average speed of all seeds in each scenario.

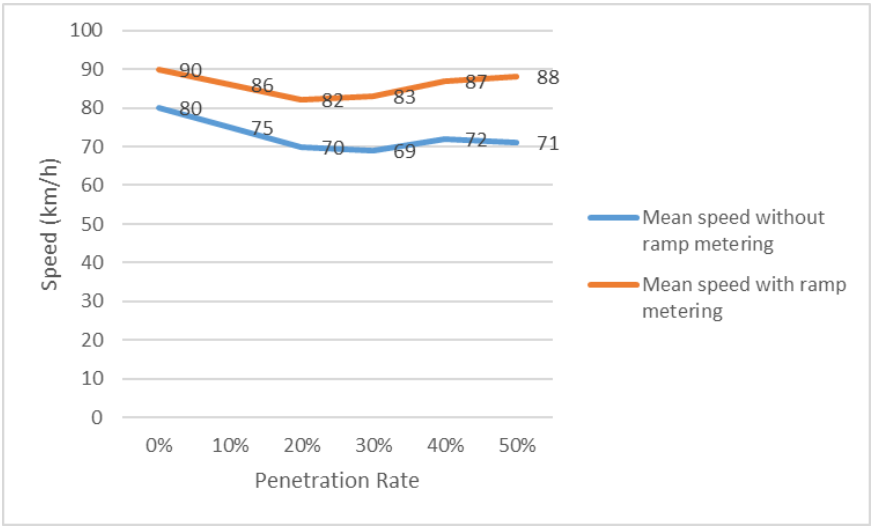


Figure 5.1: Mean speed of Basic, Low and Medium Penetration Rate of Level 2 Vehicles

The experimental outcome that ramp metering works in the basic, low and medium penetration rate scenarios is shown in Figure 5.1. The mean speed in scenarios with ramp metering improves 19.2% on the motorway, i.e., the motorway with ramp metering passing the upstream detectors with significantly higher average speed and therefore results in the larger overall flow in the simulation period.

The mean speed of vehicles benefits from ramp metering. There is a conflict when vehicles from the on-ramp merge to the motorway. When the traffic is at free flow state (far from capacity), vehicles from the on-ramp will wait for an accepted gap and then merge to the motorway. When the flow is close to capacity, it is difficult for them to find

such gap. They need cooperation from other vehicles on the motorway to create an accepted gap for them. Normally the vehicles on the motorway follow the predecessors with a particular headway. If one vehicle decelerates and creates space for the vehicles at the on-ramp, all the followers should decelerate, thus the mean speed of traffic drops. Ramp metering installation controls the number of vehicles from the on-ramp entering the motorway every minute so that the flow would never exceed the capacity.

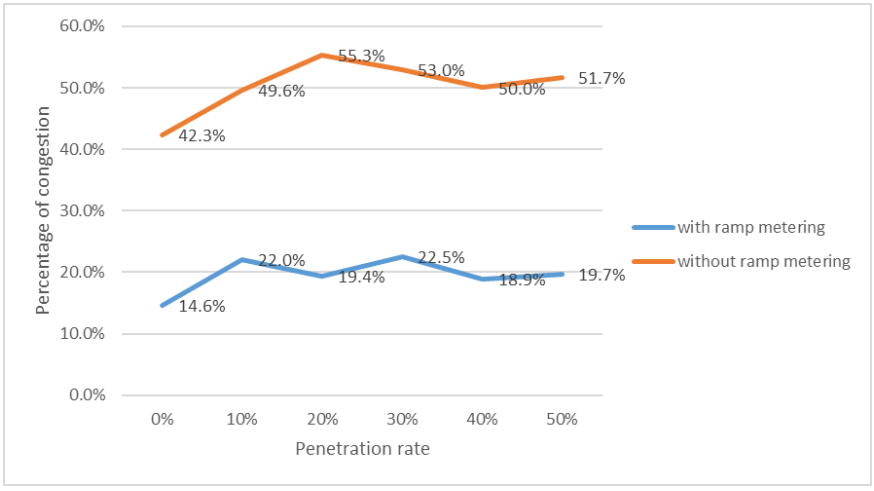


Figure 5.2: Congestion Percentage of Basic, Low and Medium Penetration Rate of Level 2 Vehicles

The congestion percentage is calculated by the average congestion duration of all seeds divided by the overall simulation duration in each scenario. According to Figure 5.2, ramp metering ameliorates 30.8% of congestion. Congestion is more likely to occur from minute 35 to minute 55 when higher demand appears. The congestion duration is expected less than 20 minutes from Demand Graph 3.4. According to simulation outputs, the average congestion duration of scenario 0% - 50% varies from 10.2 minutes to 15.8 minutes (refer to Table A.1), which is in accordance with our expectation. The congestion percentage is around 50% in scenarios without ramp metering so the mean speed is still larger than congested speed (60 km/h). The simulation only gives an insight into congestion since the top demand will merely last for 10 minutes in our study. However, the rush hour of real-world lasts up to 4 hours per day which might results in much more severe congestion without installation of ramp metering in reality.

5.1.2. CAPACITY ANALYSIS

As mentioned before, an experiment has been conducted on both maximum capacity drop shown in Table A.3 and stochastic capacity drop shown in A.4 from downstream detectors.

MAXIMUM CAPACITY DROP

The maximum capacity drop is divided into two parts. The first part is the capacity drop of scenarios without ramp metering.

L2 Penetration Rate	0	10%	20%	30%	40%	50%
Maximum Breakdown Capacity [veh/h]	4410	4200	4170	4380	4530	4380
Maximum Discharge Capacity [veh/h]	3690	3450	3188	3250	3357	3230
Maximum Capacity drop [veh/h]	720	750	982	1130	1173	1150

Table 5.1: Maximum Capacity drop of Scenario 0% - 50% (average of 11 runs)

When the penetration rate of level 2 autonomous vehicles is low or medium (Scenario 0% - 50%), the maximum breakdown capacity fluctuates around 4300 veh/h and maximum discharge capacity fluctuates around 3200 veh/h. In particular, the maximum breakdown and discharge capacity of scenario 40% are larger than the value in other scenarios. According to the detector data-set, all seeds of scenario 40% have larger breakdown and discharge capacity, which is not an extreme case of a specific seed. In other words, the motorway capacity peaks at scenario 40%. Hence the trend of maximum capacity gradually declines at low penetration rate and fluctuates in medium penetration rate. The maximum capacity drop varies from 720 veh/h to 1173 veh/h in different scenarios.

The second part is about the comparison of maximum capacity among scenarios with and without ramp metering. The data of congested flow is limited in scenarios with ramp metering because of short congestion duration, which might results in significant bias when estimating the maximum capacity drop. This applied to all scenarios with ramp metering. The maximum capacity drop is calculated as an example to make a comparison between scenario 0% and RM-0%.

According to Table 5.2, larger capacity drop in scenario RM-0% than 0% is identified. But the maximum breakdown capacity and discharge capacity are at the same level, which indicates ramp metering installation is not able to improve the capacity drop once the congestion emerges.

Scenario	0%	RM-0%
Maximum Breakdown Capacity [veh/h]	4410	4390
Maximum Discharge Capacity [veh/h]	3690	3580
Maximum Capacity drop [veh/h]	720	810

Table 5.2: Maximum Capacity drop of Scenario 0% and RM% (average of 11 runs)

STOCHASTIC CAPACITY DROP

Since the congestion duration of scenarios with ramp metering (Scenario RM-0% to RM-50%) is quite small, there are limited number of model outputs for estimating the stochastic discharge capacity in those scenarios. Therefore, the stochastic capacity drop will be analyzed only from the scenarios without ramp metering (Scenario 0% - 50%).

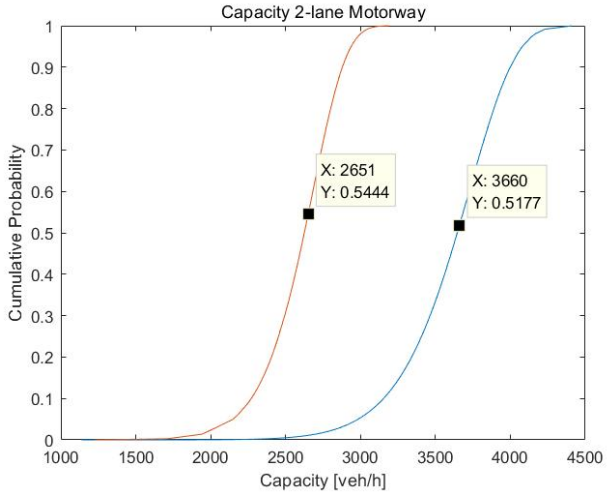


Figure 5.3: Cumulative probability curves

The estimation of the stochastic capacity drop is calculated by the Product Limit Method introduced before. An example of cumulative probability curves is shown in Figure 5.3, which indicates the stochastic breakdown capacity is 3660 veh/h and the stochastic discharge capacity is 2651 veh/h. Other cumulative probability curves for scenarios will be presented in the Appendix A.1, A.2 and A.3.

Figure 5.4 shows the stochastic capacity. The sum of the orange bar and grey bar is the stochastic breakdown capacity. Both stochastic breakdown and discharge capacity decrease when level 2 vehicles first enter the traffic and fluctuate at medium penetration rate. The trend of stochastic capacity is similar to the tendency of maximum breakdown and discharge capacity shown in Table A.3. Stochastic capacity drop remains around 900 veh/h in all scenarios.

The decrease of capacity in scenario 0% to 20% is led by the lateral motion model of level 2 vehicles. As mentioned in Section 3.2.2, the lateral motion of level 2 autonomous vehicles is modelled by the modified LMRS. The lane change assistance of level 2 vehicles is to change lane automatically when the drivers generate the desire and inform the system. It is defined as shared lateral control in this thesis. Therefore, the modified LMRS only has two levels of desires (no lane change desire and lane change desire). Whenever drivers want to change lanes, the desire comes to the lane change level and produces

lane changing even if it is not mandatory. Thus the increase of level 2 vehicles' penetration leads to more lane change behaviors. As for the lateral motion model of level 1 (LMRS), it contains four levels of desire. Drivers change lanes according to their desire and the difficulty of finding an accepted gap. Therefore, the simulation shows decreasing breakdown capacity at low penetration rate.

Along with the rising share of level 2 vehicles (scenario 30% to 40%), the breakdown capacity has a growth and finally stay at around 3700 veh/h (scenario 40% and 50%). At this stage, the accurate perception of level 2 starts to influence the breakdown capacity. There are already over 30% level 2 vehicles involvement of lane change behaviors. They perceive more precise positions or neighboring speed and lead to efficient lane change behaviors. In other words, level 2 vehicles undermine the negative impact of perception errors lead by level 1 vehicle drivers in medium penetration rate.

The stochastic breakdown capacity of the basic scenario is 3660 veh/h, which is a bit lower than the capacity set up in the RWS controller (4000 veh/h). The first reason might be the time interval to gather the detector data. According to Brilon, Geistefeldt and Regler [65], the most accurate capacity is measured with time interval of 5 minutes. However, the time interval in our experiments is chosen as 2 minutes. It is because the overall simulation time is 70 minutes and a time interval of 5 minutes cannot obtain enough data. The second reason might be the rapid rise of demand from 2200 veh/h to 3400 veh/h. The amount of stochastic breakdown capacity is limited and influence the distribution of cumulative probability curves. Also, it may be caused by the defect of simulation models.

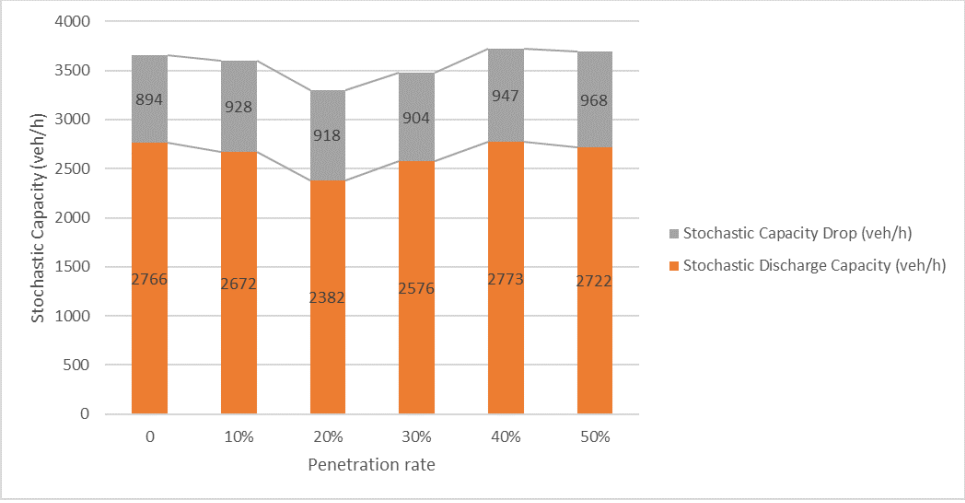


Figure 5.4: Stochastic Capacity drop of Scenario 0% - 50% (average of 11 runs)

5.1.3. TOTAL TIME SPENT AND TOTAL DELAYS

The total time spent and total delays drop significantly in scenarios with ramp metering. The total time spent decreases 9.4% and total delay reduces average 23.9% in scenarios with ramp metering. Horizontally, there is a increasing trend of TTS and D with the growth of level 2 penetration rates at the low penetration rate (scenario 0% to 20%) and a downward tendency at the median penetration rate (scenario 30% to 50%). Their trends are in accordance with the trend of breakdown capacity.

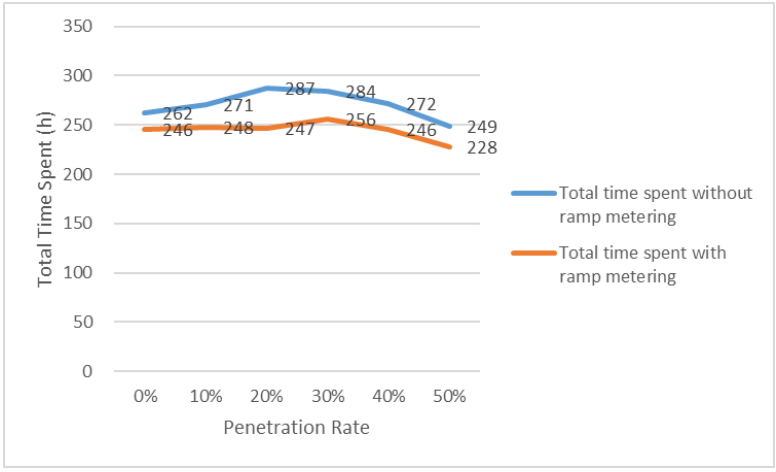


Figure 5.5: Travel Time Spent in Basic, Low and medium Penetration Rate of L2 Vehicles

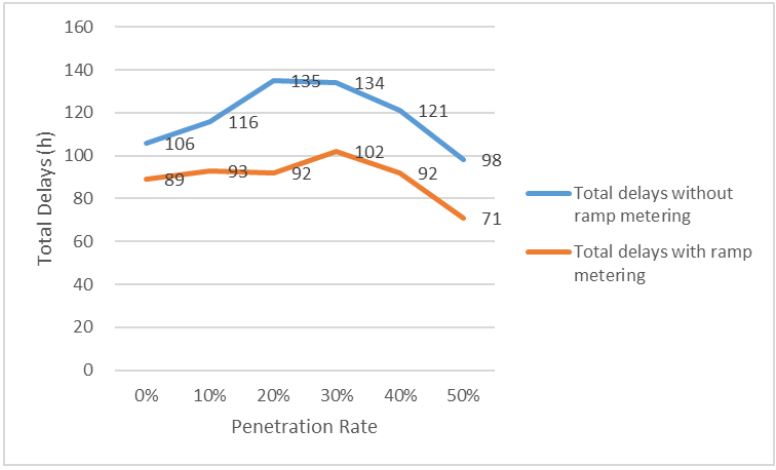


Figure 5.6: Total Delays in Basic, Low and medium Penetration Rate of L2 Vehicles

Rijkswaterstaat reports an average 10% improvement of delays when applying ramp metering measure in the Netherlands [10]. The delays improve 16% in the basic scenario

of this simulation. Thus the simulation is trusted to represent the real-world situation of ramp metering to some extent .

5.2. DISCUSSION

RESULT INTERPRETATIONS

The simulation generates the result that ramp metering keep its effectiveness when the penetration rate of partially autonomous vehicles is less than 50%. Indeed, the macroscopic behaviors of mixed traffic will change due to their different motion behaviors of level 1 and level 2 vehicles. For instance, the motorway capacity fluctuates in the low and medium penetration rate of level 2 vehicles. On the other hand, the performance of ramp metering is evaluated by the drop of congestion duration, total time spent and total delays. All of them have a significantly decrease in every scenario with ramp metering. Therefore, ramp metering measure is quite effective when the penetration rate of level 2 vehicles is less than 50%.

The traffic influence of level 2 vehicles is indicated by road capacity. According to 5.1.2, the reduction of capacity is mainly caused by the setting of lane change assistance system (assumption 4). It only have two kinds of lane change desire. It will perform lane change as long as the drivers input their wishes. While the lane change model of level 1 vehicles have four types of desires and drivers can adjust their behavior by considering the perceiving environment and their desires. Thus level 2 vehicles are more likely to change lanes that require cooperation and then have negative influence on capacity.

At the same time, assumption 1 and 2 lead to the growth of capacity at medium penetration. Although the influence of LCA system still exists, the increasing share of level 2 vehicles brings the benefit of accurate perception. Level 2 vehicles perceive more precise information about the environment and lead to efficient lane change behaviors. They undermine the negative impact of perception errors lead by drivers in level 1 vehicles and result in a moderately growth of capacity in medium penetration rate.

Overall, partially autonomous vehicles may influence the performance of ramp metering by its effect to capacity. Statistical analysis shows they gradually declines the capacity first appear in the traffic and gives rise to capacity at medium penetration rate. However, the ramp metering measure works well when the penetration rate of partially autonomous vehicles is less than 50%.

IMPLICATIONS

These results are built on existing research question 'will automated vehicles negatively impact traffic flow' [9]. This research indicates level 2 automated vehicles in mixed traffic will initially have negative effect on capacities, which verifies the conclusion of the former research. Besides, the research also shows the fluctuation of roadway capacity with the rising level 2 penetration rate. It might further lead to the malfunction of ramp metering in high penetration rate circumstances. For instance, if the breakdown capacity declines to the critical flow (3000 veh/h) set in RWS controller, the ramp metering

installations will never switch on and lose its efficiency. On the other hand, if breakdown capacity becomes larger (such as 5000 veh/h), the ramp metering installations will turn on too early and result in some unnecessary delays.

Level 2 automated vehicles indeed is a milestone for full automation. When it enter the market, they bring complicated impact for traffic capacity. However, the RWS ramp metering measure still works well although level 2 automated vehicles appear. According to the estimation of automated vehicle percentage on the road, the share of level 2 autonomous vehicles will be less than 1% in 2020 and gradually rise to 20% in 2035 [9]. Based on the simulation results, the ramp metering will be likely to be effective before 2035. After that, the development of conditional and high automation (Level 3 and Level 4) should be considered. The market forecast of high automation indicates that level 3 are possible to achieve 8% market share globally in the year 2035 [**?**]. If the share of L3 or L4 automated vehicles continuously grows and stop the increasing share of L2 vehicles, the RM effectiveness should be further studied according to the L3 or L4 vehicles' behaviors. Otherwise Rijkswaterstaat may need to use intelligent RWS parameters and set incentives to high level automation if the development of L3 and L4 vehicles are still on-going.

LIMITATION

There are some limitations of this thesis that might have impact on the results. One of the limitations is the use of the microscopic driving model. IDM+ and (modified) LMRS are used in this thesis to model the motion behaviors of level 1 and level 2 vehicles. However, they do not model all real driving behaviors because of the theoretical constraints. For example, the modified LMRS ignores the human intervene in LCA system. In real case, drivers may lose patient and change lanes manually if the system wait too long for cooperation. At that time, the levels of lane change desire are no longer 2 levels applied in the experiments. Adaptation of these models can provide more detailed behaviors and reliable results. Subsequently, the capacity and critical flow of RWS ramp metering controller are set up by experience. The performance of ramp metering may be better if there is a optimization test for those parameters. Another limitation is the data used for capacity estimation. According to Brilon, Geistefeldt and Regler [65], the most accurate capacity is estimated with flow gathered every 5 minutes. However, the time interval in the experiments is chosen as 2 minutes in order to ensure the abundant number of data set for flow. It may cause some inaccuracy when estimating the stochastic capacity.

5.3. SUMMARY

This chapter processes the outputs of simulation through descriptive statistics and statistical inference. Ramp metering is found to be effective when the penetration rate of level 2 autonomous vehicles is less than 50%. It reduces average 30.8% congestion duration and improves 19.2% the mean speed on the motorway. The total time spent and total delays also reduce when installing ramp metering on the motorway. Level 2 vehicles will result in lower capacity first and then higher capacity when they participate more in

the traffic. In general, ramp metering works well before 2035 and its effectiveness further requires an update based on the vehicle market share at that time.

6

CONCLUSION

The main objective of this thesis is to investigate the effectiveness of current Dutch ramp metering in mixed traffic by estimating the average speed, congestion duration, capacity drop and total time spent (delays). These performance indicators are obtained from statistical analysis based on the outputs of the simulation. This chapter will summarize the research findings and make conclusions regarding the research objectives and research questions. Recommendations for further scientific and technical implications will be given.

6.1. FINDINGS

In this thesis, a simulation study is proposed to analyze the effectiveness of Dutch ramp metering to mixed traffic. The major findings are:

1. The appearance of level 2 autonomous vehicles lead to significant reduction of stochastic breakdown capacity. The stochastic breakdown capacity first drops from 3660 veh/h to 3300 veh/h when the penetration rate gradually reaches 20%. Then it grows up to around 3700 veh/h and remain stable at medium penetration rate. The stochastic discharge capacity, maximum breakdown and discharge capacity also shows the trend.

2. Ramp metering works well if the penetration rate of level 2 autonomous vehicles is less than 50%. It improves 19.2% mean speed on the motorway and profoundly reduces 30.8% congestion duration. Meanwhile, capacity drop disappear in most scenarios with ramp metering.

6.2. ANSWER OF RESEARCH QUESTIONS

Some sub-questions are formulated from the main research question and answered in Chapters 2 to 5. This section will summarize these answers.

a) Which Dutch dynamic traffic management measure will be researched in this thesis and how it is developed?

   There are various Dynamic traffic management measures applied in the Netherlands, including variable speed limit (VSL), dynamic route guidance (DRG), extra/dedicated lanes, incident management camera and ramp metering (RM). Some of them are introduced and compared in section 1.2. It is difficult to predict the effectiveness of all DTM in mix traffic due to the time limitation of this research. Therefore, ramp metering (RM) is chosen as an example of DTM measures and focus on the effectiveness of Dutch RM in mixed traffic. The idea of ramp metering first come up in the United States and then spread in Europe. There are also many developed ramp metering algorithms including Demand - Capacity (DC) strategy, Occupancy strategy, Fuzzy Logic strategy, ALINEA strategy and so on. Each

of them will lead to a influence to the ramp metering effect.

b) What are the microscopic driving models of partially autonomous vehicles and driver assistant vehicles?

The car-following model of driver assistant vehicles and partially autonomous vehicles is an adapted version of the Intelligent Drivers Model (IDM+). The lane change model of driver assistant vehicles is Integrated Lane Change Model with Relaxation and Synchronization (LMRS) and the lane change model of partially autonomous vehicles is modified Integrated Lane Change Model with Relaxation and Synchronization (LMRS) with two types of lane change desires.

c) How will partially autonomous vehicles influence the macroscopic behaviors of mixed traffic under different penetration rates?

The macroscopic behaviors of mixed traffic is evaluated by indicators such as motorway capacity, mean speed and total time spent. The breakdown capacity and discharge capacity of mixed traffic first reduce 9.8% and 13.9% respectively in low participation of level 2 vehicles. Then it turns into an upward trend. The breakdown capacity and discharge capacity reach 3720 veh/h and 2773 veh/h correspondingly. Finally the capacity become stable around 3700 veh/h and 2750 veh/h. The capacity drop remains around 900 veh/h in different scenarios. On the other hand, the mean speed of mixed traffic, total time spent and total delays will have the same trend when participation of partially autonomous vehicles is the same.

d) How will the current measure selected in sub-question (a) influence the macroscopic behavior of mixed traffic?

Ramp metering is found to be quite effective in mixed traffic when the penetration rate of level 2 vehicles is less than 50%. It improves on average 19.2% mean speed, reduces 23.9% total delays and prevents the capacity drop.

e) Could the current measure selected in sub-question (a) be improved and how to improve?

Yes, it can be improved by optimizing the parameter settings. When the penetration rate of partially autonomous vehicles is larger than 50%, the parameters setting of the ramp metering controller may need an update due to the unknown changes of capacity. Nowadays, the RWS controller usually uses the capacity of 4000 veh/h and a critical flow of 3000 veh/h shown in Figure 3.3. For example, if the stochastic capacity reduces to 3000 veh/h in penetration rate above 50%, the capacity parameter should be set as 3000 veh/h and the critical flow changes to 2250 veh/h correspondingly.

Therefore, the answer to the main research question is:

How partial automated driving influences the performance of current Dutch dynamic traffic management system and how could this be evaluated via simulation?

This thesis takes ramp metering as an example DTM measure and research its effectiveness with automated vehicles by simulation. The simulation is structured with a 2-lane motorway, an on-ramp and an RWS ramp controller. The vehicles in the simulation consist of driver assistance (level 1) and partially autonomous (Level 2) vehicles. Level 1 vehicles are equipped with the ACC system and modelled by IDM+ longitudinally and LMRS laterally. Level 2 vehicles are equipped with ACC and LCA systems. Similarly, IDM+ and modified LMRS are used to model the car following and lane change behaviors of level 2 autonomous vehicles. The evaluation of RM effectiveness is based on the statistical analysis of mean speed, capacity, total time spent and total delays. In the simulation, ramp metering is found effective when the penetration rate of level 2 vehicles is less than 50%.

In conclude, the Lane Change Assistance (LCA) system in this thesis brings first a negative and then a positive impact for the motorway capacity. Due to the unknown changes of capacity in high penetration rate, it is important to monitor the capacity changes if level 2 vehicles enter the market. For instance, ramp metering may lose its efficiency when the capacity changes away from the capacity set up in the ramp metering controller. As mentioned in Section 3.2.2, level 2 automation is still developing nowadays. The on-sale level 2 vehicles may have different parameters setup or even different motion models. Thus it requires to find the critical penetration rate of level 2 autonomous vehicles based on the market share in the future. At the same time, Rijkswaterstaat should pay attention to vehicles' share in the future.

6.3. RECOMMENDATIONS

The following section proposes some recommendations for future research. As the research target is Dutch traffic management, the second subsection will give some practical suggestions for Rijkswaterstaat based on the outcome of the research.

6.3.1. RECOMMENDATION FOR FUTURE RESEARCH

One of the limitations of this research is that only ramp metering with RWS algorithm is simulated. It is interesting to look at the optimization of ramp metering strategy in automated traffic. The effectiveness of other ramp metering algorithms introduced in 2.6 can be studied and found out the optimal algorithm applied in mixed traffic.

Another limitation is that partial autonomous vehicles in this thesis are equipped with ACC and LCA, while there are various partial automation systems such as Traffic Jam Assistance (TJA), longitudinal motion control plus Blind Spot Monitoring System

(BSMS) or Automatic Lane Change system (ALC) which are not chosen. Each type of partially autonomous vehicles has their behavior characteristics and might require different modeling paradigm in the simulation. Following the similar process of this study, the performance of ramp metering can be tested when different types of partially autonomous vehicles participate. After that, research on the influence of different ADAS systems to RM effectiveness can be further developed.

In this thesis, the ramp metering is simulated with automated vehicles. While the implementation of automated vehicles will first result in mixed traffic including conventional vehicles and automated vehicles. Researchers can study the performance of ramp metering with automated vehicles combining with conventional vehicles, which further lead to comparison about ramp metering performance with this thesis.

It is also worth looking into the conditions when and how to update the parameter of ramp metering controllers in the future since level 2 vehicles brings changes to capacity. Predictive researches to the time when to update RM parameters might be required. In the future, it will be possible to predict the market share statistically and find out the year that the penetration rate becomes high. Rijkswaterstaat should take action in advance, either optimizing the RWS ramp metering parameters or updating their controller.

Besides, the main research topic 'Analysis of current Dutch traffic management effectiveness with automated vehicles' has not been finished. This thesis only focuses on ramp-metering measure. There are various of DTM measures such as Variable Speed Limit, Dynamic Route Guidance, Extra Lanes at Rush Hours, etc. Further research about different DTM controllers can enrich the answer of our research.

6.3.2. RECOMMENDATIONS FOR RIJKSWATERSTAAT

This thesis studies a ramp metering case to give an insight of current Dutch DTM effectiveness with automated vehicles. In general, ramp metering works effectively with (less than) 50% level 2 automated vehicles. However, other Dutch DTM may have different performance. Due to its effect to traffic behaviors by various motion models and penetration rate, some design algorithms of other DTM may lose its applicability. The changes of traffic behaviors may further reduce the DTM effectiveness. Therefore Rijkswaterstaat should always keep their eyes' on automation design domains and the vehicle market share when different levels of automation enters the traffic in the future.

The suggestions to ramp metering measure are proposed from the following two aspects.

1. **Keep the ramp metering activated and carry out periodic maintenance**.
   Partial autonomous vehicles defined in this thesis have not been put into commercial use. The technique still needs developing and field tests. Therefore Rijkswaterstaat can keep the current ramp metering installations in the near future (before 2035) and continue required maintenance such as regular check-up for detectors

and ramp metering lights.

2. **Use intelligent ramp metering controller**.
   Furthermore, Rijkswaterstaat can also use intelligent ramp metering controllers which can calculate the stochastic breakdown capacity and updating its parameters themselves according to detected data. However, it is costly and might require cost-benefit analysis.

REFERENCES

[3] SAE, *Automated driving - levels of automation,* https://cdn.oemoffhighway.com/files/base/acbm/ooh/document/2016/03/automated_driving.pdf (2016).

[11] F. van Wageningen-Kessels, H. van Lint, K. Vuik, and S. Hoogendoorn, *Genealogy of traffic flow models,* EURO J Transp Logist (2015).

[12] L. Pipes, *An operational analysis of traffic dynamics,* J Appl Phys 24(3) (1953).

[15] G. F. Newell, *Nonlinear effects in the dynamics of car following,* Oper Res 9(2) (1961).

[26] V. Knoop, *Lateral Driving Behavior* (Technology University of Delft, 2018).

[28] W. J. Schakel, V. L. Knoop, and B. van Arem, *Integrated lane change model with relaxation and synchronization,* Transportation Research Record: Journal of the Transportation Research Board (2012).

[31] *Ramp Metering: Operation and Analysis* (VASITE Annual Meeting, 2017).

[19] V. L.Knoop, *Introduction to Traffic Flow Theory: Theory and exercises* (Delft, the Netherlands, 2018).

[34] *Advanced Coordinated Traffic Responsive Ramp Metering Strategies* (UC Berkeley, 1999).

[36] *Ten Years of Ramp-Metering in the Netherlands* (Rijkswaterstaat - Transport Research Centre (AVV), 2000).

[38] S. Hoogendoorn, R. Landman, J. van Kooten, and M. Schreuder, *Integrated network management amsterdam: Control approach and test results,* 16th International IEEE Annual Conference on Intelligent Transportation Systems (ITSC 2013) (2013).

[41] A. Hegyi, *Ramp Metering* (Technology University of Delft, 2018).

[63] S. C. Calvert, H. Taale, and S. P. Hoogendoorn, *Quantification of motorway capacity variation: influence of day type specific variation and capacity drop,* JOURNAL OF ADVANCED TRANSPORTATION **50** (2016).

[1] S. K.Gehrig and F. J.Stein, *Dead reckoning and cartography using stereo vision for an autonomous car,* IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients **3**, 1507 (1999).

[2] IVsource, *Adaptive control arrives in usa,* https://web.archive.org/web/20080908013626/http://www.ivsource.net/archivep/2000/sep/a000929_USacc.html (2000).

[4] J. Freyer, B. Deml, M. Maurer, and B. Färber, *Acc with enhanced situation awareness to reduce behavior adaptations in lane change situations,* IEEE Intelligent Vehicles Symposium **6**, 1000 (2007).

[5] F. Middelham, *State of practice in dynamic traffic management in the netherlands,* Elsevier (2003).

[6] E. Grumert and A. Tapani, *Impacts of a cooperative variable speed limit system,* Procedia - Social and Behavioral Sciences **43**, 595 (2012).

[7] *Ramp Metering in the Netherlands: An Overview* (Rijkswaterstaat AVV Transport Research Center, 2006).

[8] *State of Practice in Dynamic Traffic Management in the Netherlands* (Elsevier, 2003).

[9] S. C. Calvert, W. J. Schakel, and J. W. C. van Lint, *Will automated vehicles negatively impact traffic flow?* Journal of Advanced Transportation **2017**.

[10] H. Taale, *Effecten van benutting in Nederland* (Rijkswaterstaat, 2018).

[13] E. Kometani and T. Sasaki, *Dynamic behaviour of traffic with a nonlinear spacing–speed relationship,* in *Herman R (ed) Theory of traffic flow 1959* (Elsevier, Amsterdam, 1961).

[14] P. G. Gipps, *A behavioural car-following model for computer simulation,* Transp Res Part B Methodol 15(2) (1981).

[16] G. F. Newell, *A simplified car-following theory: a lower order model,* Transp Res Part B Methodol 36(3) (2002).

[17] J. Leclercq, L.and Laval and E. Chevallier, *The lagrangian coordinates and what it means for first order,* in *Allsop RE, Bell MGH, Heydecker BG (eds) Transportation and traffic theory* (Elsevier, Oxford, 2007).

[18] K. N. A. S. A. Bando, M.and Hasebe and Y. Sugiyama, *Dynamical model of traffic congestion and numerical simulation,* Phys Rev E Stat Nonlinear Soft Matter Phys 51 (1995).

[20] M. Treiber, A. Hennecke, and D. Helbing, *Congested traffic states in empirical observations and microscopic simulations,* Phys Rev E Stat Nonlinear Soft Matter Phys 62(2) (2000).

[21] M. Cremer and J. Ludwig, *A fast simulation model for traffic flow on the basis of boolean operations,* Math Comput Simul 28(4) (1986).

[22] S. Knospe, W. and A. S. M. L., Schadschneider, *Empirical test for cellular automaton models of traffic flow,* Phys Rev E Stat Nonlinear Soft Matter Phys 70(1 Pt 2) (2004).

[23] van Wageningen-Kessels, H. F., van Lint, K. Vuik, and S. Hoogendoorn, *Genealogy of traffic flow models,* EURO J Transp Logist (2015) (2015).

[24] Q. Yang and H. Koutsopoulos, *A microscopic traffic simulator for evaluation of dynamic traffic management systems,* Transportation Research Part C: Emerging Technologies 4 (3) (1996).

[25] C. F. Daganzo, *A behavioral theory of multi-lane traffic flow part i: Long homogeneous freeway sections,* Transportation Research Part B: Methodological 36(2) (2002).

[27] A. Kesting, M. Treiber, and D. Helbing, *General lane-changing model mobil for car following models,* Transportation Research Record: Journal of the Transportation Research Board **1999**, 86 (2007).

[29] A. Spiliopoulou, G. Perraki, M. Papageorgiou, and C. Roncoli, *Exploitation of acc systems towards improved traffic flow efficiency on motorways,* 2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS) (2017).

[30] H. Liu, X. Kan, S. E.Shladover, X.-Y. Lu, and R. E.Ferlis, *Impact of cooperative adaptive cruise control on multilane freeway merge capacity,* Journal of Intelligent Transportation Systems **22** (2018).

[32] B. S.Kerner, *Three-phase traffic theory and highway capacity,* Physica A: Statistical Mechanics and its Applications **333**, 379 (2004).

[33] K. Yuan, V. L.Knoop, and S. P.Hoogendoorn, *Capacity drop: a relation between the speed in congestion and the queue discharge rate,* Transportation Research Record **8** (2014).

[35] C. Gould, H. Rayman, and k. McCabe, *Ramp metering development in a uk context,* 12th IEE International Conference on Road Transport Information and Control **12** (2004).

[37] L. Knoop, H. Taale, M. M., E. P., and S. Hoogendoorn, *Ramp metering with real-time estimation of parameters,* International Conference on Intelligent Transportation System (ITSC) (2018).

[39] J. Renaud and N. Faouzi, *Nearctis – european society for traffic management and control,* Transportation Research Procedia 14 (2016).

[40] J. Aydos and A. O'Brien, *Scats ramp metering: Strategies, arterial integration and results,* 17th International IEEE Conference on Intelligent Transportation Systems (ITSC) (2014).

[42] K. Åström and R. Murray, *Feedback System* (Princeton University Press, 2012).

[43] *Advanced Coordinated Traffic Responsive Ramp Metering Strategies* (UC Berkeley, 1999).

[44] *Peak-period Analysis and Control of A Freeway System* (The Texas Highway Department, 1965).

[45] M. Papageorgiou, *Freeway ramp metering: An overview,* IEEE Transaction on Intelligent Transportation System, Vol. 3, NO. 4 (2002).

[46] E. Smaragdis and M. Papageorgiou, *Strategies, series of new local ramp metering,* Transportation Research Record 1856 (2003).

[47] *Ten Years of Ramp-Metering in the Netherlands* (Rijkswaterstaat - Transport Research Centre (AVV), 2000).

[48] *The assessment of ramp metering based on fuzzy logic* (Third ITS World Congress, 1996).

[49] M. Papageorgiou, H. Hadj-Salem, and J. M. Blosseville, *Alinea: A local feedback control law for on-ramp metering,* Transportation Research Record, 1320(1) (1991).

[50] M. Papageorgiou, H. Hadj-Salem, and F. Middelham, *Alinea local ramp metering: Summary of field results,* Transportation Research Record (1997).

[51] M. Stanescu, *Adaptive ramp metering* (2008).

[52] *Integrated Network Management Amsterdam: Control Approach and Test Results* (16th International IEEE Annual Conference on Intelligent Transportation Systems (ITSC 2013), Hague, 1976).

[53] SAE, *Adaptive cruise control (acc) operating characteristics and user interface,* https://saemobilus.sae.org/content/j2399_201409 (2014).

[54] M. Treiber, A. Kesting, and D. Helbing, *Delays, inaccuracies and anticipation in microscopic traffic models,* Physica A: Statistical Mechanics and its Applications **360** (2006).

[55] A. Kesting, M. Treiber, M. Sch¨onhof, F. Kranke, and D. Helbing, *Driver assistance systems for the active congestion avoidance in road traffic,* VDI Berichte **22** (2006).

[56] W. J. Schakel, B. van Arem, and B. D. Netten, *Effects of cooperative adaptive cruise control on traffic flow stability,* 13th International IEEE Conference on Intelligent Transportation Systems (2010).

[57] J. C. McCall and M. M. Trivedi, *Video-based lane estimation and tracking for driver assistance: Survey, system, and evaluation,* IEEE Transactions on Intelligent Transportation Systems, 7(1) (2006).

[58] Tesla, *Autopilot features,* https://www.tesla.com/autopilot (2019).

[59] R. Groenveld, *Analysis of input and output data* (2001).

[60] Z. Wang and C. Liu, *The minimum yellow timing for ramp meters,* Transportation Science and Technology **3** (2014).

[61] *Highway capacity manual,* http://onlinepubs.trb.org/onlinepubs/trnews/trnews273hcm2010.pdf (2010).

[62] M. Lorenz and L. Elefteriadou, *A probabilistic approach to defining freeway capacity and breakdown,* in *Transportation Research Circular E-CO18, (4th International Symposium on Highway Capacity) Transportation Research Board, National Research Council* (2000) pp. 84–85.

[64] M. Stanescu, *Adaptive ramp metering,* http://its-edulab.nl/files/MSc%20Thesis%20Marc%20Stanescu.pdf (2008).

[65] W. Brilon, J. Geistefeldt, and M. Regler, *Reliability of freeway traffic flow: A stochastic concept of capacity,* in *6th International Symposium on Transportation and Traffic Theory* (College Park, Maryland, 2005) pp. 125–144.

[66] Arane, *Onderzoek verkeersafwikkeling en capaciteitswaarden discontinuïteiten,* RWS AVV (2007).

Appendices

A

STATISTICS ANALYSIS OF MODEL OUTPUTS

A.1. MEAN SPEED

L2 Penetration Rate	0%	RM-0%	10%	RM-10%	20%	RM-20%	30%	RM-30%
Mean Speed [km/h]	80	90	75	86	70	87	83	69
S.D. of Speed [km/h]	5.8	5.0	5.1	5.2	6.3	6.6	4.3	6.9
Mean S.D. of Speed [km/h]	31.6	20.2	33.9	23.1	36.6	25.9	36.9	25.4
Congestion Duration [min]	29.6	10.2	34.7	15.4	38.7	13.6	37.1	15.8

Table A.1: Congestion Period L2 Basic and Low Penetration Rate (seed = 1 - 11)

L2 Penetration Rate	40%	RM-40%	50%	RM-50%
Mean speed [km/h]	72	87	71	88
S.D. of Speed [km/h]	5.6	6.4	6.6	6.9
Mean S.D. of Speed [km/h]	37.8	25.3	37.4	24.2
Congestion Duration [min]	35.0	13.2	36.2	13.8

Table A.2: Congestion Period L2 Medium Penetration Rate (seed = 1 - 11)

A.2. CAPACITY DROP

- **Maximum Capacity**

L2 Penetration Rate	0	10%	20%	30%	40%	50%	60%
Maximum Breakdown Capacity [veh/h]	4200	4410	4170	4200	4380	4110	4620
Maximum Discharge Capacity [veh/h]	3450	4410	3188	3281	3146	3285	3215
Maximum Capacity Drop [veh/h]	750	0	982	919	1234	815	1405

Table A.3: Maximum Capacity Drop of Scenario 0% - 60% (average of 11 runs)

- **Stochastic Capacity**

L2 Penetration Rate	0	10%	20%	30%	40%	50%
Stochastic Breakdown Capacity [veh/h]	3660	3600	3300	3480	3720	3690
Stochastic Discharge Capacity [veh/h]	2651	2672	2677	2678	2671	2677
Stochastic Capacity Drop [veh/h]	1009	928	623	802	1049	1013

Table A.4: Stochastic Capacity Drop of Scenario 0% - 60% (average of 11 runs)

The estimation of stochastic breakdown capacity and discharge capacity are based on Product Limit Method. The cumulative probability curve of each scenario is shown below.

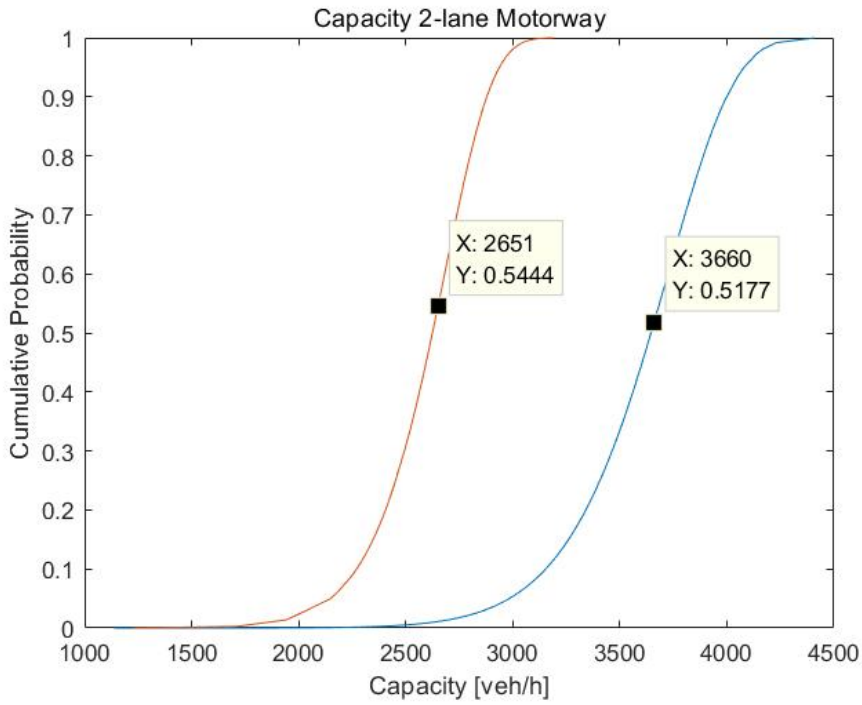
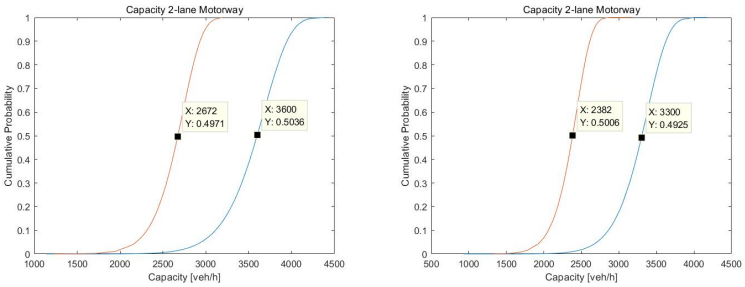


Figure A.1: Cumulative Probability Curves for Stochastic Capacity-0%

A.3. TOTAL TIME SPENT AND TOTAL DELAYS

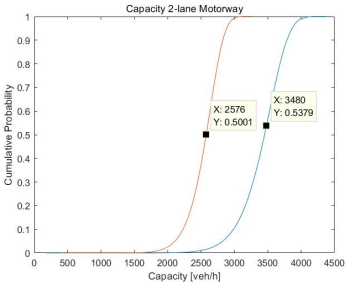
L2 Penetration Rate	Average TTS [h]	L2 Penetration Rate	Average TTS [h]
0%	262	RM-0%	246
10%	271	RM-10%	248
20%	287	RM-20%	247
30%	284	RM-30%	256
40%	272	RM-40%	246
50%	249	RM-50%	228

Table A.5: Average Total Travel Time of All Scenario (average of 11 runs)

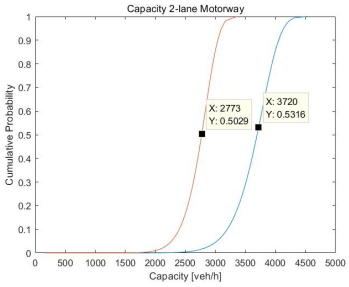


(a) 10%

(b) 20%



(c) 30%



(d) 40%

Figure A.2: Cumulative Probability Curves for Stochastic Capacity

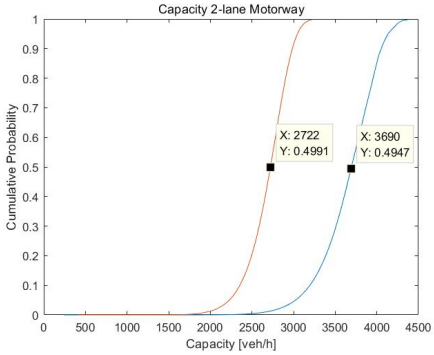


Figure A.3: Cumulative Probability Curves for Stochastic Capacity-50%

L2 Penetration Rate	Average D [h]	L2 Penetration Rate	Average D [h]
0%	106	RM-0%	89
10%	116	RM-10%	93
20%	135	RM-20%	92
30%	134	RM-30%	102
40%	121	RM-40%	92
50%	98	RM-50%	71

Table A.6: Average Total Travel Time of All Scenario (average of 11 runs)

B

OTS Code

- Road Network

```
package org.opentrafficsim.demo;

import java.io.BufferedWriter;
import java.io.IOException;
import java.rmi.RemoteException;
import java.util.ArrayList;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;
import java.util.Set;

import org.djunits.unit.FrequencyUnit;
import org.djunits.unit.SpeedUnit;
import org.djunits.unit.TimeUnit;
import org.djunits.value.StorageType;
import org.djunits.value.vdouble.scalar.Acceleration;
import org.djunits.value.vdouble.scalar.Direction;
import org.djunits.value.vdouble.scalar.Duration;
import org.djunits.value.vdouble.scalar.Length;
import org.djunits.value.vdouble.scalar.Speed;
import org.djunits.value.vdouble.scalar.Time;
import org.djunits.value.vdouble.vector.FrequencyVector;
import org.djunits.value.vdouble.vector.TimeVector;
import org.djutils.cli.CliUtil;
import org.djutils.exceptions.Throw;
import org.djutils.exceptions.Try;
```

```
import org.opentrafficsim.base.CompressedFileWriter;
import org.opentrafficsim.base.parameters.ParameterException;
import org.opentrafficsim.base.parameters.ParameterSet;
import org.opentrafficsim.base.parameters.ParameterTypeDuration;
import org.opentrafficsim.base.parameters.ParameterTypes;
import org.opentrafficsim.base.parameters.Parameters;
import org.opentrafficsim.base.parameters.constraint.NumericConstraint;
import org.opentrafficsim.core.animation.gtu.colorer.AccelerationGTUColorer;
import org.opentrafficsim.core.animation.gtu.colorer.GTUColorer;
import org.opentrafficsim.core.animation.gtu.colorer.IDGTUColorer;
import org.opentrafficsim.core.animation.gtu.colorer.SpeedGTUColorer;
import org.opentrafficsim.core.animation.gtu.colorer.SwitchableGTUColorer;
import org.opentrafficsim.core.compatibility.Compatible;
import org.opentrafficsim.core.dsol.OTSSimulatorInterface;
import org.opentrafficsim.core.geometry.OTSPoint3D;
import org.opentrafficsim.core.gtu.GTU;
import org.opentrafficsim.core.gtu.GTUCharacteristics;
import org.opentrafficsim.core.gtu.GTUDirectionality;
import org.opentrafficsim.core.gtu.GTUException;
import org.opentrafficsim.core.gtu.GTUType;
import org.opentrafficsim.core.gtu.perception.DirectEgoPerception;
import org.opentrafficsim.core.gtu.perception.EgoPerception;
import org.opentrafficsim.core.gtu.perception.Perception;
import org.opentrafficsim.core.gtu.plan.operational.OperationalPlan;
import
    org.opentrafficsim.core.gtu.plan.operational.OperationalPlanException;
import org.opentrafficsim.core.network.LateralDirectionality;
import org.opentrafficsim.core.network.LinkType;
import org.opentrafficsim.core.network.Network;
import org.opentrafficsim.core.network.NetworkException;
import org.opentrafficsim.core.network.Node;
import org.opentrafficsim.core.network.route.Route;
import org.opentrafficsim.core.parameters.ParameterFactoryByType;
import org.opentrafficsim.road.gtu.colorer.GTUTypeColorer;
import
    org.opentrafficsim.road.gtu.generator.characteristics.LaneBasedGTUCharacteristics;
import
    org.opentrafficsim.road.gtu.generator.od.DefaultGTUCharacteristicsGeneratorOD;
import
    org.opentrafficsim.road.gtu.generator.od.GTUCharacteristicsGeneratorOD;
import org.opentrafficsim.road.gtu.generator.od.ODApplier;
import org.opentrafficsim.road.gtu.generator.od.ODOptions;
import org.opentrafficsim.road.gtu.lane.LaneBasedGTU;
import org.opentrafficsim.road.gtu.lane.VehicleModel;
import
    org.opentrafficsim.road.gtu.lane.perception.CategoricalLanePerception;
import org.opentrafficsim.road.gtu.lane.perception.LanePerception;
import org.opentrafficsim.road.gtu.lane.perception.PerceptionCollectable;
import org.opentrafficsim.road.gtu.lane.perception.RelativeLane;
```

```
import
    org.opentrafficsim.road.gtu.lane.perception.categories.AnticipationTrafficPerception;
import
    org.opentrafficsim.road.gtu.lane.perception.categories.DirectInfrastructurePerception;
import
    org.opentrafficsim.road.gtu.lane.perception.categories.DirectIntersectionPerception;
import
    org.opentrafficsim.road.gtu.lane.perception.categories.InfrastructurePerception;
import org.opentrafficsim.road.gtu.lane.perception.categories.neighbors
.DirectNeighborsPerception;
import
    org.opentrafficsim.road.gtu.lane.perception.categories.neighbors.HeadwayGtuType;
import
    org.opentrafficsim.road.gtu.lane.perception.categories.neighbors.NeighborsPerception;
import org.opentrafficsim.road.gtu.lane.perception.headway.HeadwayGTU;
import org.opentrafficsim.road.gtu.lane.plan.operational.LaneChange;
import
    org.opentrafficsim.road.gtu.lane.plan.operational.LaneOperationalPlanBuilder;
import
    org.opentrafficsim.road.gtu.lane.plan.operational.SimpleOperationalPlan;
import org.opentrafficsim.road.gtu.lane.tactical.LaneBasedTacticalPlanner;
import
    org.opentrafficsim.road.gtu.lane.tactical.LaneBasedTacticalPlannerFactory;
import org.opentrafficsim.road.gtu.lane.tactical.following.AbstractIDM;
import
    org.opentrafficsim.road.gtu.lane.tactical.following.CarFollowingModel;
import org.opentrafficsim.road.gtu.lane.tactical.following.IDMPlus;
import
    org.opentrafficsim.road.gtu.lane.tactical.lmrs.AbstractIncentivesTacticalPlanner;
import org.opentrafficsim.road.gtu.lane.tactical.lmrs.AccelerationIncentive;
import org.opentrafficsim.road.gtu.lane.tactical.util.CarFollowingUtil;
import org.opentrafficsim.road.gtu.lane.tactical.util.TrafficLightUtil;
import org.opentrafficsim.road.gtu.lane.tactical.util.lmrs.Cooperation;
import org.opentrafficsim.road.gtu.lane.tactical.util.lmrs.Desire;
import org.opentrafficsim.road.gtu.lane.tactical.util.lmrs.Incentive;
import org.opentrafficsim.road.gtu.lane.tactical.util.lmrs.LmrsParameters;
import org.opentrafficsim.road.gtu.lane.tactical.util.lmrs.LmrsUtil;
import org.opentrafficsim.road.gtu.strategical.LaneBasedStrategicalPlanner;
import
    org.opentrafficsim.road.gtu.strategical.LaneBasedStrategicalPlannerFactory;
import org.opentrafficsim.road.gtu.strategical.od.Categorization;
import org.opentrafficsim.road.gtu.strategical.od.Category;
import org.opentrafficsim.road.gtu.strategical.od.Interpolation;
import org.opentrafficsim.road.gtu.strategical.od.ODMatrix;
import
    org.opentrafficsim.road.gtu.strategical.route.LaneBasedStrategicalRoutePlannerFactory;
import org.opentrafficsim.road.network.OTSRoadNetwork;
import
    org.opentrafficsim.road.network.control.rampmetering.CycleTimeLightController;
```

```
import org.opentrafficsim.road.network.control.rampmetering.RampMetering;
import
    org.opentrafficsim.road.network.control.rampmetering.RampMeteringLightController;
import
    org.opentrafficsim.road.network.control.rampmetering.RampMeteringSwitch;
import org.opentrafficsim.road.network.control.rampmetering.RwsSwitch;
import org.opentrafficsim.road.network.control.rampmetering.AlineaSwitch;
import org.opentrafficsim.road.network.factory.LaneFactory;
import org.opentrafficsim.road.network.lane.Lane;
import org.opentrafficsim.road.network.lane.LaneType;
import org.opentrafficsim.road.network.lane.OTSRoadNode;
import org.opentrafficsim.road.network.lane.Stripe.Permeable;
import org.opentrafficsim.road.network.lane.changing.LaneKeepingPolicy;
import org.opentrafficsim.road.network.lane.object.sensor.Detector;
import
    org.opentrafficsim.road.network.lane.object.sensor.Detector.CompressionMethod;
import org.opentrafficsim.road.network.lane.object.sensor.SinkSensor;
import
    org.opentrafficsim.road.network.lane.object.trafficlight.SimpleTrafficLight;
import
    org.opentrafficsim.road.network.lane.object.trafficlight.TrafficLight;
import org.opentrafficsim.road.network.speed.SpeedLimitInfo;
import org.opentrafficsim.road.network.speed.SpeedLimitProspect;
import org.opentrafficsim.swing.script.AbstractSimulationScript;

import nl.tudelft.simulation.dsol.logger.SimLogger;
import nl.tudelft.simulation.event.EventInterface;
import nl.tudelft.simulation.jstats.distributions.DistNormal;
import nl.tudelft.simulation.jstats.streams.StreamInterface;
import nl.tudelft.simulation.language.d3.DirectedPoint;
import picocli.CommandLine.Option;

/**
 * <p>
 * Copyright (c) 2013-2019 Delft University of Technology, PO Box 5, 2600
     AA, Delft, the Netherlands. All rights reserved. <br>
 * BSD-style license. See <a
     href="http://opentrafficsim.org/node/13">OpenTrafficSim License</a>.
 * <p>
 * @version $Revision$, $LastChangedDate$, by $Author$, initial version 12
     jun. 2019 <br>
 * @author <a href="http://www.tbm.tudelft.nl/averbraeck">Alexander
     Verbraeck</a>
 * @author <a href="http://www.tudelft.nl/pknoppers">Peter Knoppers</a>
 * @author <a href="http://www.transport.citg.tudelft.nl">Wouter Schakel</a>
 */
public class RampMeteringDemo extends AbstractSimulationScript
{
```

```
/** Controlled car GTU type id. */
private static final String CONTROLLED_CAR_ID = "controlledCar";

/** Parameter factory. */
private ParameterFactoryByType parameterFactory = new
    ParameterFactoryByType();

/** Ramp metering. */
@Option(names = { "-r", "--rampMetering" }, description = "Ramp metering
    on or off", defaultValue = "true")
private boolean rampMetering;

/** Whether to generate output. */
@Option(names = "--output", description = "Generate output.", negatable
    = true, defaultValue = "false")
private boolean output;

/** Accepted gap. */
@Option(names = "--acceptedGap", description = "Accepted gap.",
    defaultValue = "0.5s")
private Duration acceptedGap;

/** L1 percentage. */
@Option(names = "--l1percentage", description = "Level 1 Percentage.",
    defaultValue = "0.6")
private Double l1percentage;

/** L2 percentage. */
@Option(names = "--l2percentage", description = "Level 2 Percentage.",
    defaultValue = "0.4")
private Double l2percentage;

/** Main demand. */
private FrequencyVector mainDemand;

/** Main demand string. */
@Option(names = "--mainDemand", description = "Main demand in veh/h.",
    defaultValue = "2000,3000,3900,3900,3000")
private String mainDemandString;

/** Ramp demand. */
private FrequencyVector rampDemand;

/** Ramp demand string. */
@Option(names = "--rampDemand", description = "Ramp demand in veh/h.",
    defaultValue = "500,500,500,500,500")
private String rampDemandString;

/** Demand time. */
```

```
    private TimeVector demandTime;

    /** Demand time string. */
    @Option(names = "--demandTime", description = "Demand time in min.",
        defaultValue = "0,10,40,50,70")
    private String demandTimeString;

    /** Scenario. */
    @Option(names = "--scenario", description = "Scenario name.",
        defaultValue = "test")
    private String scenario;

    /** GTUs in simulation. */
    private Map<String, Double> gtusInSimulation = new LinkedHashMap<>();

    /** Total travel time, accumulated. */
    private double totalTravelTime = 0.0;

    /** Total travel time delay, accumulated. */
    private double totalTravelTimeDelay = 0.0;

    /**
     * Constructor.
     */
    protected RampMeteringDemo()
    {
        super("Ramp metering 1", "Ramp metering 2");
    }

    /**
     * @param args String[] command line arguments
     * @throws Exception any exception
     */
    public static void main(final String[] args) throws Exception
    {
        RampMeteringDemo demo = new RampMeteringDemo();
        CliUtil.changeOptionDefault(demo, "simulationTime", "4200s");
        CliUtil.execute(demo, args);
        demo.mainDemand =
                new FrequencyVector(arrayFromString(demo.mainDemandString),
                    FrequencyUnit.PER_HOUR, StorageType.DENSE);
        demo.rampDemand =
                new FrequencyVector(arrayFromString(demo.rampDemandString),
                    FrequencyUnit.PER_HOUR, StorageType.DENSE);
        demo.demandTime = new
            TimeVector(arrayFromString(demo.demandTimeString),
            TimeUnit.BASE_MINUTE, StorageType.DENSE);
        demo.start();
    }
```

```
/**
 * Returns an array from a String.
 * @param str String; string
 * @return double[] array
 */
private static double[] arrayFromString(final String str)
{
    int n = 0;
    for (String part : str.split(","))
    {
        n++;
    }
    double[] out = new double[n];
    int i = 0;
    for (String part : str.split(","))
    {
        out[i] = Double.valueOf(part);
        i++;
    }
    return out;
}

/** {@inheritDoc} */
@Override
protected OTSRoadNetwork setupSimulation(final OTSSimulatorInterface
    sim) throws Exception
{
    SimLogger.setSimulator(sim);

    OTSRoadNetwork network = new OTSRoadNetwork("RampMetering", true);
    if (this.output)
    {
        network.addListener(this, Network.GTU_ADD_EVENT);
        network.addListener(this, Network.GTU_REMOVE_EVENT);
    }
    GTUType car = network.getGtuType(GTUType.DEFAULTS.CAR);
    GTUType controlledCar = new GTUType(CONTROLLED_CAR_ID, car);

    GTUColorer[] colorers =
            new GTUColorer[] { new IDGTUColorer(), new
                SpeedGTUColorer(new Speed(150, SpeedUnit.KM_PER_HOUR)),
                   new
                       AccelerationGTUColorer(Acceleration.createSI(-6.0),
                       Acceleration.createSI(2)),
                   new GTUTypeColorer().add(car).add(controlledCar) };
    SwitchableGTUColorer colorer = new SwitchableGTUColorer(0, colorers);
    setGtuColorer(colorer);
```

```
// parameters
StreamInterface stream =
    sim.getReplication().getStream("generation");
this.parameterFactory.addParameter(ParameterTypes.FSPEED, new
    DistNormal(stream, 123.7 / 120.0, 12.0 / 1200));

OTSRoadNode nodeA = new OTSRoadNode(network, "A", new OTSPoint3D(0,
    0), Direction.ZERO);
OTSRoadNode nodeB = new OTSRoadNode(network, "B", new
    OTSPoint3D(3000, 0), Direction.ZERO);
OTSRoadNode nodeC = new OTSRoadNode(network, "C", new
    OTSPoint3D(3250, 0), Direction.ZERO);
OTSRoadNode nodeD = new OTSRoadNode(network, "D", new
    OTSPoint3D(6000, 0), Direction.ZERO);
OTSRoadNode nodeE = new OTSRoadNode(network, "E", new
    OTSPoint3D(2000, -25), Direction.ZERO);
OTSRoadNode nodeF = new OTSRoadNode(network, "F", new
    OTSPoint3D(2750, 0.0), Direction.ZERO);

LinkType freeway = network.getLinkType(LinkType.DEFAULTS.FREEWAY);
LaneKeepingPolicy policy = LaneKeepingPolicy.KEEPRIGHT;
Length laneWidth = Length.createSI(3.6);
LaneType freewayLane =
    network.getLaneType(LaneType.DEFAULTS.FREEWAY);
Speed speedLimit = new Speed(120, SpeedUnit.KM_PER_HOUR);
Speed rampSpeedLimit = new Speed(70, SpeedUnit.KM_PER_HOUR);
List<Lane> lanesAB = new LaneFactory(network, nodeA, nodeB, freeway,
    sim, policy)
        .leftToRight(1.0, laneWidth, freewayLane,
            speedLimit).addLanes(Permeable.BOTH).getLanes();
List<Lane> lanesBC = new LaneFactory(network, nodeB, nodeC, freeway,
    sim, policy)
        .leftToRight(1.0, laneWidth, freewayLane,
            speedLimit).addLanes(Permeable.BOTH,
            Permeable.LEFT).getLanes();
List<Lane> lanesCD = new LaneFactory(network, nodeC, nodeD, freeway,
    sim, policy)
        .leftToRight(1.0, laneWidth, freewayLane,
            speedLimit).addLanes(Permeable.BOTH).getLanes();
List<Lane> lanesEF =
       new LaneFactory(network, nodeE, nodeF, freeway, sim,
            policy).setOffsetEnd(laneWidth.multiplyBy(1.5).neg())
               .leftToRight(0.5, laneWidth, freewayLane,
                   rampSpeedLimit).addLanes().getLanes();
List<Lane> lanesFB = new LaneFactory(network, nodeF, nodeB, freeway,
    sim, policy)
        .setOffsetStart(laneWidth.multiplyBy(1.5).neg())
        .setOffsetEnd(laneWidth.multiplyBy(1.5).neg())
        .leftToRight(0.5, laneWidth, freewayLane,
```

```
            speedLimit).addLanes().getLanes();
for (Lane lane : lanesCD)
{
    new SinkSensor(lane,
        lane.getLength().minus(Length.createSI(50)),
        GTUDirectionality.DIR_PLUS, sim);
}
// detectors
Duration agg = Duration.createSI(120.0);
// TODO: detector length affects occupancy, which length to use?
Length detectorLength = Length.createSI(6.1);
Detector det1 = new Detector("1", lanesAB.get(0),
    Length.createSI(2900), detectorLength, sim, agg,
    Detector.MEAN_SPEED,
       Detector.OCCUPANCY);
Detector det2 = new Detector("2", lanesAB.get(1),
    Length.createSI(2900), detectorLength, sim, agg,
    Detector.MEAN_SPEED,
       Detector.OCCUPANCY);
Detector det3 = new Detector("3", lanesCD.get(0),
    Length.createSI(100), detectorLength, sim, agg,
    Detector.MEAN_SPEED,
       Detector.OCCUPANCY);
Detector det4 = new Detector("4", lanesCD.get(1),
    Length.createSI(100), detectorLength, sim, agg,
    Detector.MEAN_SPEED,
       Detector.OCCUPANCY);
List<Detector> detectors12 = new ArrayList<>();
detectors12.add(det1);
detectors12.add(det2);
List<Detector> detectors34 = new ArrayList<>();
detectors34.add(det3);
detectors34.add(det4);
if (this.rampMetering)
{
    // traffic light
    TrafficLight light = new SimpleTrafficLight("light",
        lanesEF.get(0), lanesEF.get(0).getLength(), sim);
    List<TrafficLight> lightList = new ArrayList<>();
    lightList.add(light);
    // ramp metering
    RampMeteringSwitch rampSwitch = new AlineaSwitch(detectors34);
    RampMeteringLightController rampLightController =
           new CycleTimeLightController(sim, lightList,
               Compatible.EVERYTHING);
    new RampMetering(sim, rampSwitch, rampLightController);
}

// OD
```

```
        List<OTSRoadNode> origins = new ArrayList<>();
        origins.add(nodeA);
        origins.add(nodeE);
        List<OTSRoadNode> destinations = new ArrayList<>();
        destinations.add(nodeD);
        Categorization categorization = new Categorization("cat",
            GTUType.class);// , Lane.class);
        Interpolation globalInterpolation = Interpolation.LINEAR;
        ODMatrix od = new ODMatrix("rampMetering", origins, destinations,
            categorization, this.demandTime, globalInterpolation);
        // Category carCatMainLeft = new Category(categorization, car,
            lanesAB.get(0));
        // Category carCatMainRight = new Category(categorization, car,
            lanesAB.get(1));
        Category carCatRamp = new Category(categorization, car);// ,
            lanesEB.get(0));
        Category controlledCarCat = new Category(categorization,
            controlledCar);
        // double fLeft = 0.6;
        od.putDemandVector(nodeA, nodeD, carCatRamp, this.mainDemand,
            this.l1percentage);
        od.putDemandVector(nodeA, nodeD, controlledCarCat, this.mainDemand,
            this.l2percentage);
        // od.putDemandVector(nodeA, nodeD, carCatMainLeft, mainDemand,
            fLeft);
        // od.putDemandVector(nodeA, nodeD, carCatMainRight, mainDemand, 1.0
            - fLeft);
        od.putDemandVector(nodeE, nodeD, carCatRamp, this.rampDemand,
            this.l1percentage);
        od.putDemandVector(nodeE, nodeD, controlledCarCat, this.rampDemand,
            this.l2percentage);
        ODOptions odOptions = new ODOptions();
        odOptions.set(ODOptions.GTU_TYPE, new
            ControlledStrategicalPlannerGenerator()).set(ODOptions.INSTANT_LC,
            true);
        ODApplier.applyOD(network, od, sim, odOptions);

        return network;
    }

    /**
     * Returns the parameter factory.
     * @return ParameterFactoryByType; parameter factory
     */
    final ParameterFactoryByType getParameterFactory()
    {
        return this.parameterFactory;
    }
```

```
/** {@inheritDoc} */
@Override
public void notify(final EventInterface event) throws RemoteException
{
    if (event.getType().equals(Network.GTU_ADD_EVENT))
    {
        this.gtusInSimulation.put((String) event.getContent(),
            getSimulator().getSimulatorTime().si);
    }
    else if (event.getType().equals(Network.GTU_REMOVE_EVENT))
    {
        measureTravelTime((String) event.getContent());
    }
    else
    {
        super.notify(event);
    }
}

/**
 * Adds travel time and delay for a single GTU.
 * @param id String; id of the GTU
 */
private void measureTravelTime(final String id)
{
    double tt = getSimulator().getSimulatorTime().si -
        this.gtusInSimulation.get(id);
    double x = getNetwork().getGTU(id).getOdometer().si;
    // TODO: we assume 120km/h everywhere, including the slower ramps
    double ttd = tt - (x / (120 / 3.6));
    this.totalTravelTime += tt;
    this.totalTravelTimeDelay += ttd;
    this.gtusInSimulation.remove(id);
}

/** {@inheritDoc} */
@Override
protected void onSimulationEnd()
{
    if (this.output)
    {
        // detector data
        String file = String.format("%s_%02d_detectors.txt",
            this.scenario, getSeed());
        Detector.writeToFile(getNetwork(), file, true, "%.3f",
            CompressionMethod.NONE);

        // travel time data
        for (GTU gtu : getNetwork().getGTUs())
```

```
            {
                measureTravelTime(gtu.getId());
            }
            Throw.when(!this.gtusInSimulation.isEmpty(),
                 RuntimeException.class,
                    "GTUs remain in simulation that are not measured.");
            file = String.format("%s_%02d_time.txt", this.scenario,
                 getSeed());
            BufferedWriter bw = CompressedFileWriter.create(file, false);
            try
            {
                bw.write(String.format("Total travel time: %.3fs",
                     this.totalTravelTime));
                bw.newLine();
                bw.write(String.format("Total travel time delay: %.3fs",
                     this.totalTravelTimeDelay));
                bw.close();
            }
            catch (IOException exception)
            {
                throw new RuntimeException(exception);
            }
            finally
            {
                try
                {
                    if (bw != null)
                    {
                        bw.close();
                    }
                }
                catch (IOException ex)
                {
                    throw new RuntimeException(ex);
                }
            }
        }
    }

    /**
     * Strategical planner generator. This class can be used as input in
         {@code ODOptions} to generate the right models with
     * different GTU types.
     */
    private class ControlledStrategicalPlannerGenerator implements
        GTUCharacteristicsGeneratorOD
    {

        /** Default generator. */
```

```
private DefaultGTUCharacteristicsGeneratorOD defaultGenerator = new
    DefaultGTUCharacteristicsGeneratorOD();

/** Controlled planner factory. */
private
    LaneBasedStrategicalPlannerFactory<LaneBasedStrategicalPlanner>
    controlledPlannerFactory;

/** Constructor. */
ControlledStrategicalPlannerGenerator()
{
   // anonymous factory to create tactical planners for controlled
       GTU's
   LaneBasedTacticalPlannerFactory<?> tacticalPlannerFactory =
           new
               LaneBasedTacticalPlannerFactory<LaneBasedTacticalPlanner>()
           {
               @Override
               public Parameters getParameters() throws
                   ParameterException
               {
                   ParameterSet set = new ParameterSet();
                   set.setDefaultParameter(ParameterTypes.PERCEPTION);
                   set.setDefaultParameter(ParameterTypes.LOOKBACK);
                   set.setDefaultParameter(ParameterTypes.LOOKAHEAD);
                   set.setDefaultParameter(ParameterTypes.S0);
                   set.setDefaultParameter(ParameterTypes.TMAX);
                   set.setDefaultParameter(ParameterTypes.TMIN);
                   set.setDefaultParameter(ParameterTypes.DT);
                   set.setDefaultParameter(ParameterTypes.VCONG);
                   set.setDefaultParameter(ParameterTypes.T0);
                   set.setDefaultParameter(TrafficLightUtil.B_YELLOW);
                   set.setDefaultParameters(LmrsParameters.class);
                   set.setDefaultParameters(AbstractIDM.class);
                   return set;
               }

               @SuppressWarnings("synthetic-access")
               @Override
               public LaneBasedTacticalPlanner create(final
                   LaneBasedGTU gtu) throws GTUException
               {
                   // here the lateral control system is initiated
                   ParameterSet settings = new ParameterSet();
                   try
                   {
                       // system operation settings
                       settings.setParameter(SyncAndAccept.SYNCTIME,
                           Duration.createSI(1.0));
```

```
                        settings.setParameter(SyncAndAccept.COOPTIME,
                            Duration.createSI(2.0));
                        // parameters used in car-following model for
                            gap-acceptance
                        settings.setParameter(AbstractIDM.DELTA, 1.0);
                        settings.setParameter(ParameterTypes.S0,
                            Length.createSI(3.0));
                        settings.setParameter(ParameterTypes.TMAX,
                            Duration.createSI(1.8));
                        settings.setParameter(ParameterTypes.A,
                            Acceleration.createSI(2.0));
                        settings.setParameter(ParameterTypes.B,
                            Acceleration.createSI(2.0));
                        settings.setParameter(ParameterTypes.T,
                            RampMeteringDemo.this.acceptedGap);
                        settings.setParameter(ParameterTypes.FSPEED,
                            1.0);
                        settings.setParameter(ParameterTypes.B0,
                            Acceleration.createSI(0.5));
                        settings.setParameter(ParameterTypes.VCONG,
                            new Speed(60, SpeedUnit.KM_PER_HOUR));
                    }
                    catch (ParameterException exception)
                    {
                        throw new GTUException(exception);
                    }
                    return new ControlledTacticalPlanner(gtu, new
                        SyncAndAccept(gtu, new IDMPlus(), settings));
                }
            };
        // standard strategical planner factory using the tactical
            factory and the simulation-wide parameter factory
        this.controlledPlannerFactory = new
            LaneBasedStrategicalRoutePlannerFactory(tacticalPlannerFactory,
               RampMeteringDemo.this.getParameterFactory());
    }

    /** {@inheritDoc} */
    @Override
    public LaneBasedGTUCharacteristics draw(final Node origin, final
        Node destination, final Category category,
           final StreamInterface randomStream) throws GTUException
    {
        GTUType gtuType = category.get(GTUType.class);
        // if GTU type is a controlled car, create characteristics for a
            controlled car
        if (gtuType.equals(getNetwork().getGtuType(CONTROLLED_CAR_ID)))
        {
            Route route = null;
```

```
            VehicleModel vehicleModel = VehicleModel.MINMAX;
            GTUCharacteristics gtuCharacteristics =
                    GTUType.defaultCharacteristics(gtuType,
                        origin.getNetwork(), randomStream);
            return new LaneBasedGTUCharacteristics(gtuCharacteristics,
                this.controlledPlannerFactory, route, origin,
                    destination, vehicleModel);
        }
        // otherwise generate default characteristics
        return this.defaultGenerator.draw(origin, destination, category,
            randomStream);
    }

}

/** Tactical planner. */
private static class ControlledTacticalPlanner extends
    AbstractIncentivesTacticalPlanner
{
    /** */
    private static final long serialVersionUID = 20190731L;

    /** Lane change system. */
    private AutomaticLaneChangeSystem laneChangeSystem;

    /** Lane change status. */
    private final LaneChange laneChange;

    /** Map that {@code getLaneChangeDesire} writes current desires in.
        This is not used here. */
    private Map<Class<? extends Incentive>, Desire> dummyMap = new
        LinkedHashMap<>();

    /**
     * Constructor.
     * @param gtu LaneBasedGTU; gtu
     * @param laneChangeSystem AutomaticLaneChangeSystem; lane change
         system
     */
    ControlledTacticalPlanner(final LaneBasedGTU gtu, final
        AutomaticLaneChangeSystem laneChangeSystem)
    {
        super(new IDMPlus(), gtu, generatePerception(gtu));
        setDefaultIncentives();
        this.laneChangeSystem = laneChangeSystem;
        this.laneChange = Try.assign(() -> new LaneChange(gtu),
            "Parameter LCDUR is required.", GTUException.class);
    }
```

```
/**
 * Helper method to create perception.
 * @param gtu LaneBasedGTU; gtu
 * @return LanePerception lane perception
 */
private static LanePerception generatePerception(final LaneBasedGTU
    gtu)
{
    CategoricalLanePerception perception = new
        CategoricalLanePerception(gtu);
    perception.addPerceptionCategory(new
        DirectEgoPerception<LaneBasedGTU,
        Perception<LaneBasedGTU>>(perception));
    perception.addPerceptionCategory(new
        DirectInfrastructurePerception(perception));
    // TODO: perceived GTUs as first type
    perception.addPerceptionCategory(new
        DirectNeighborsPerception(perception, HeadwayGtuType.WRAP));
    perception.addPerceptionCategory(new
        AnticipationTrafficPerception(perception));
    perception.addPerceptionCategory(new
        DirectIntersectionPerception(perception,
        HeadwayGtuType.WRAP));
    return perception;
}

/** {@inheritDoc} */
@Override
public OperationalPlan generateOperationalPlan(final Time startTime,
    final DirectedPoint locationAtStartTime)
       throws OperationalPlanException, GTUException,
           NetworkException, ParameterException
{
    // get some general input
    Speed speed =
        getPerception().getPerceptionCategory(EgoPerception.class).getSpeed();
    SpeedLimitProspect slp =
        getPerception().getPerceptionCategory(InfrastructurePerception.class)
           .getSpeedLimitProspect(RelativeLane.CURRENT);
    SpeedLimitInfo sli = slp.getSpeedLimitInfo(Length.ZERO);

    // LMRS desire
    Desire desire =
        LmrsUtil.getLaneChangeDesire(getGtu().getParameters(),
        getPerception(), getCarFollowingModel(),
           getMandatoryIncentives(), getVoluntaryIncentives(),
               this.dummyMap);

    // other vehicles respond to these 'interpreted' levels of lane
```

```
    change desire
getGtu().getParameters().setParameter(LmrsParameters.DLEFT,
    desire.getLeft());
getGtu().getParameters().setParameter(LmrsParameters.DRIGHT,
    desire.getRight());

// car-following
Acceleration a = getGtu().getCarFollowingAcceleration();

// cooperation
Acceleration aCoop =
    Cooperation.PASSIVE.cooperate(getPerception(),
    getGtu().getParameters(), sli,
       getCarFollowingModel(), LateralDirectionality.LEFT,
           desire);
a = Acceleration.min(a, aCoop);
aCoop = Cooperation.PASSIVE.cooperate(getPerception(),
    getGtu().getParameters(), sli, getCarFollowingModel(),
       LateralDirectionality.RIGHT, desire);
a = Acceleration.min(a, aCoop);

// compose human plan
SimpleOperationalPlan simplePlan =
       new SimpleOperationalPlan(a,
           getGtu().getParameters().getParameter(ParameterTypes.DT));
for (AccelerationIncentive incentive :
    getAccelerationIncentives())
{
    incentive.accelerate(simplePlan, RelativeLane.CURRENT,
        Length.ZERO, getGtu(), getPerception(),
           getCarFollowingModel(), speed,
               getGtu().getParameters(), sli);
}

// add lane change control
if (!this.laneChange.isChangingLane())
{
    double dFree =
        getGtu().getParameters().getParameter(LmrsParameters.DFREE);
    if (this.laneChangeSystem.initiatedLaneChange().isNone())
    {
        if (desire.leftIsLargerOrEqual() && desire.getLeft() >
            dFree)
        {
            this.laneChangeSystem.initiateLaneChange(LateralDirectionality.LEFT);
        }
        else if (desire.getRight() > dFree)
        {
            this.laneChangeSystem.initiateLaneChange(LateralDirectionality.RIGHT);
```

```
                }
            }
            else
            {
                if ((this.laneChangeSystem.initiatedLaneChange().isLeft()
                    && desire.getLeft() < dFree)
                        ||
                            (this.laneChangeSystem.initiatedLaneChange().isRight()
                            && desire.getRight() < dFree))
                {
                    this.laneChangeSystem
                .initiateLaneChange(LateralDirectionality.NONE);
                }
            }
        }
        simplePlan = this.laneChangeSystem.operate(simplePlan,
            getGtu().getParameters());
        simplePlan.setTurnIndicator(getGtu());

        // create plan
        return
            LaneOperationalPlanBuilder.buildPlanFromSimplePlan(getGtu(),
            startTime, simplePlan, this.laneChange);
    }
}

/** Interface allowing tactical planners to use an automatic lane change
    system. */
private interface AutomaticLaneChangeSystem
{

    /**
     * Update operational plan with actions to change lane. This method
         should be called by the tactical planner always.
     * @param simplePlan SimpleOperationalPlan; plan
     * @param parameters Parameters; parameters
     * @return SimpleOperationalPlan; adapted plan
     * @throws OperationalPlanException if the system runs in to an error
     * @throws ParameterException if a parameter is missing
     */
    SimpleOperationalPlan operate(SimpleOperationalPlan simplePlan,
        Parameters parameters)
            throws OperationalPlanException, ParameterException;

    /**
     * Returns the direction in which the system was initiated to
         perform a lane change.
     * @return LateralDirectionality; direction in which the system was
         initiated to perform a lane change, {@code NONE} if
```

```
     *        none
     */
    LateralDirectionality initiatedLaneChange();

    /**
     * Initiate a lane change.
     * @param dir LateralDirectionality; direction, use {@code NONE} to
         cancel
     */
    void initiateLaneChange(LateralDirectionality dir);

}

/** Implementation of an automatic lane change system. */
private static class SyncAndAccept implements AutomaticLaneChangeSystem
{
    /** Parameter of time after lane change command when the system will
        start synchronization. */
    public static final ParameterTypeDuration SYNCTIME = new
        ParameterTypeDuration("tSync",
           "Time after which synchronization starts.",
               Duration.createSI(1.0), NumericConstraint.POSITIVE);

    /** Parameter of time after lane change command when the system will
        start cooperation (indicator). */
    public static final ParameterTypeDuration COOPTIME = new
        ParameterTypeDuration("tCoop",
           "Time after which cooperation starts (indicator).",
               Duration.createSI(2.0), NumericConstraint.POSITIVE);

    /** GTU. */
    private final LaneBasedGTU gtu;

    /** Car-following model for gap-acceptance. */
    private final CarFollowingModel carFollowingModel;

    /** Parameters containing the system settings. */
    private final Parameters settings;

    /** Initiated lane change direction. */
    private LateralDirectionality direction = LateralDirectionality.NONE;

    /** Time when the lane change was initiated. */
    private Time initiationTime;

    /**
     * Constructor.
     * @param gtu LaneBasedGTU; GTU
     * @param carFollowingModel CarFollowingModel; car-following model
```

```
 * @param settings Parameters; system settings
 */
SyncAndAccept(final LaneBasedGTU gtu, final CarFollowingModel
    carFollowingModel, final Parameters settings)
{
    this.gtu = gtu;
    this.carFollowingModel = carFollowingModel;
    this.settings = settings;
}

/** {@inheritDoc} */
@Override
public SimpleOperationalPlan operate(final SimpleOperationalPlan
    simplePlan, final Parameters parameters)
       throws OperationalPlanException, ParameterException
{
    // active?
    if (this.direction.isNone())
    {
        return simplePlan;
    }

    // check gap
    InfrastructurePerception infra =
           this.gtu.getTacticalPlanner().getPerception()
        .getPerceptionCategory(InfrastructurePerception.class);
    SpeedLimitInfo sli =
        infra.getSpeedLimitProspect(RelativeLane.CURRENT)
    .getSpeedLimitInfo(Length.ZERO);
    NeighborsPerception neighbors =
           this.gtu.getTacticalPlanner().getPerception()
        .getPerceptionCategory(NeighborsPerception.class);
    if (infra.getLegalLaneChangePossibility(RelativeLane.CURRENT,
        this.direction).gt0()
           && !neighbors.isGtuAlongside(this.direction)
           && acceptGap(neighbors.getFirstFollowers(this.direction),
               sli, false)
           && acceptGap(neighbors.getFirstLeaders(this.direction),
               sli, true))
    {
        // gaps accepted, start lane change
        SimpleOperationalPlan plan =
               new
                   SimpleOperationalPlan(simplePlan.getAcceleration(),
                   simplePlan.getDuration(), this.direction);
        this.direction = LateralDirectionality.NONE;
        this.initiationTime = null;
        return plan;
    }
```

```
        // synchronization
        Duration since =
            this.gtu.getSimulator().getSimulatorTime().minus(this.initiationTime);
        if (since.gt(this.settings.getParameter(SYNCTIME))
                || this.gtu.getSpeed()
            .lt(this.settings.getParameter(ParameterTypes.VCONG)))
        {
            PerceptionCollectable<HeadwayGTU, LaneBasedGTU> leaders =
                    neighbors.getLeaders(new RelativeLane(this.direction,
                        1));
            if (!leaders.isEmpty())
            {
                HeadwayGTU leader = leaders.first();
                Acceleration a =
                    CarFollowingUtil.followSingleLeader(this.carFollowingModel,
                    this.settings,
                       this.gtu.getSpeed(), sli, leader);
                a = Acceleration.max(a,
                    this.settings.getParameter(ParameterTypes.B).neg());
                simplePlan.minimizeAcceleration(a);
            }
        }

        // cooperation
        if (since.gt(this.settings.getParameter(COOPTIME))
                || this.gtu.getSpeed()
            .lt(this.settings.getParameter(ParameterTypes.VCONG)))
        {
            if (this.direction.isLeft())
            {
                simplePlan.setIndicatorIntentLeft();
            }
            else
            {
                simplePlan.setIndicatorIntentRight();
            }
        }

        // return
        return simplePlan;
    }

    /**
     * Checks whether a gap can be accepted.
     * @param neighbors Set&lt;HeadwayGTU&gt;; neighbors
     * @param sli SpeedLimitInfo; speed limit info
     * @param leaders boolean; whether we are dealing with leaders, or
         followers
```

```
         * @return boolean; whether the gap is accepted
         * @throws ParameterException if a parameter is not defined
         */
        private boolean acceptGap(final Set<HeadwayGTU> neighbors, final
            SpeedLimitInfo sli, final boolean leaders)
                throws ParameterException
        {
            for (HeadwayGTU neighbor : neighbors)
            {
                Acceleration a =
                    CarFollowingUtil.followSingleLeader(this.carFollowingModel,
                    this.settings,
                        leaders ? this.gtu.getSpeed() : neighbor.getSpeed(),
                            sli, neighbor.getDistance(),
                        leaders ? neighbor.getSpeed() : this.gtu.getSpeed());
                if (a.lt(this.settings.getParameter(ParameterTypes.B).neg()))
                {
                    return false;
                }
            }
            return true;
        }

        /** {@inheritDoc} */
        @Override
        public LateralDirectionality initiatedLaneChange()
        {
            return this.direction;
        }

        /** {@inheritDoc} */
        @Override
        public void initiateLaneChange(final LateralDirectionality dir)
        {
            this.direction = dir;
            if (!dir.isNone())
            {
                this.initiationTime =
                    this.gtu.getSimulator().getSimulatorTime();
            }
            else
            {
                this.initiationTime = null;
            }
        }
    }

}
```

- Ramp Metering following RWS algorithms


```
package org.opentrafficsim.road.network.control.rampmetering;

import java.util.List;

import org.djunits.unit.FrequencyUnit;
import org.djunits.unit.SpeedUnit;
import org.djunits.value.vdouble.scalar.Duration;
import org.djunits.value.vdouble.scalar.Frequency;
import org.djunits.value.vdouble.scalar.Speed;
import org.djutils.exceptions.Throw;
import org.opentrafficsim.road.network.lane.object.sensor.Detector;

import nl.tudelft.simulation.dsol.logger.SimLogger;

/**
 * Switch implementing the RWS algorithm.
 * <p>
 * Copyright (c) 2013-2019 Delft University of Technology, PO Box 5, 2600
     AA, Delft, the Netherlands. All rights reserved. <br>
 * BSD-style license. See <a
     href="http://opentrafficsim.org/node/13">OpenTrafficSim License</a>.
 * <p>
 * @version $Revision$, $LastChangedDate$, by $Author$, initial version 12
     jun. 2019 <br>
 * @author <a href="http://www.tbm.tudelft.nl/averbraeck">Alexander
     Verbraeck</a>
 * @author <a href="http://www.tudelft.nl/pknoppers">Peter Knoppers</a>
 * @author <a href="http://www.transport.citg.tudelft.nl">Wouter Schakel</a>
 */
public class RwsSwitch extends SingleCrossSectionSwitch
{

    /** Maximum cycle time. */
    private static final Duration MAX_CYCLE_TIME = Duration.createSI(15);

    /** Capacity. */
    private final Frequency capacity;

    /** Flow threshold. */
    private final Frequency flowThreshold;

    /** Speed threshold. */
    private final Speed speedThreshold = new Speed(70,
        SpeedUnit.KM_PER_HOUR);

    /** Red time. */
```

```
    private Duration cycleTime;

    /** Flow in previous time step. */
    private Frequency lastFlow;

    /**
     * @param detectors List&lt;Detector&gt;; detectors
     */
    public RwsSwitch(final List<Detector> detectors)
    {
        super(Duration.createSI(60.0), detectors);
        this.capacity = new Frequency(2000,
            FrequencyUnit.PER_HOUR).multiplyBy(detectors.size());
        this.flowThreshold = new Frequency(1500,
            FrequencyUnit.PER_HOUR).multiplyBy(detectors.size());
    }

    /** {@inheritDoc} */
    @Override
    public boolean isEnabled()
    {
        Frequency flow = totalFlow();
        SimLogger.always().info("Flow is " +
            flow.getInUnit(FrequencyUnit.PER_HOUR));
        if (meanSpeed().le(this.speedThreshold)
                || (this.lastFlow != null && flow.gt(this.lastFlow) &&
                    flow.gt(this.flowThreshold)))
        {
            if (flow.lt(this.capacity))
            {
                this.cycleTime = Duration.createSI(1.0 /
                    this.capacity.minus(flow).si);
                this.cycleTime = Duration.min(this.cycleTime, MAX_CYCLE_TIME);
            }
            else
            {
                this.cycleTime = MAX_CYCLE_TIME;
            }
            this.lastFlow = flow;
            return true;
        }
        this.lastFlow = flow;
        return false;
    }

    /** {@inheritDoc} */
    @Override
    public Duration getCycleTime()
    {
```

```
        Throw.whenNull(this.cycleTime, "The method isEnabled() in a
            RwsSwitch should set a cycle time.");
        return this.cycleTime;
    }

}
```


- Cycle Time Controller

```
package org.opentrafficsim.road.network.control.rampmetering;

import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;

import org.djunits.value.vdouble.scalar.Duration;
import org.djunits.value.vdouble.scalar.Time;
import org.djutils.exceptions.Try;
import org.opentrafficsim.core.compatibility.Compatible;
import org.opentrafficsim.core.dsol.OTSSimulatorInterface;
import org.opentrafficsim.core.gtu.RelativePosition;
import org.opentrafficsim.core.network.NetworkException;
import org.opentrafficsim.road.gtu.lane.LaneBasedGTU;
import org.opentrafficsim.road.network.lane.CrossSectionElement;
import org.opentrafficsim.road.network.lane.object.sensor.AbstractSensor;
import
    org.opentrafficsim.road.network.lane.object.trafficlight.TrafficLight;
import
    org.opentrafficsim.road.network.lane.object.trafficlight.TrafficLightColor;

import nl.tudelft.simulation.dsol.SimRuntimeException;
import
    nl.tudelft.simulation.dsol.formalisms.eventscheduling.SimEventInterface;
import nl.tudelft.simulation.dsol.logger.SimLogger;
import nl.tudelft.simulation.dsol.simtime.SimTimeDoubleUnit;
import
    nl.tudelft.simulation.dsol.simulators.DEVSSimulatorInterface.TimeDoubleUnit;

/**
 * Controller using a cycle time.
 * <p>
 * Copyright (c) 2013-2019 Delft University of Technology, PO Box 5, 2600
     AA, Delft, the Netherlands. All rights reserved. <br>
 * BSD-style license. See <a
     href="http://opentrafficsim.org/node/13">OpenTrafficSim License</a>.
 * <p>
```

```
 * @version $Revision$, $LastChangedDate$, by $Author$, initial version 12
     jun. 2019 <br>
 * @author <a href="http://www.tbm.tudelft.nl/averbraeck">Alexander
     Verbraeck</a>
 * @author <a href="http://www.tudelft.nl/pknoppers">Peter Knoppers</a>
 * @author <a href="http://www.transport.citg.tudelft.nl">Wouter Schakel</a>
 */
public class CycleTimeLightController implements RampMeteringLightController
{

    /** Minimum red duration. */
    private static final Duration MIN_RED_TIME = Duration.createSI(2.0);

    /** Whether the controller is enabled. */
    private boolean enabled = false;

    /** Time when red phase was started. */
    private Map<TrafficLight, Time> greenStarts = new LinkedHashMap<>();

    /** Cycle time. */
    private Duration cTime;

    /** Simulator. */
    private final OTSSimulatorInterface simulator;

    /** Traffic lights. */
    private final List<TrafficLight> trafficLights;

    /** Scheduled red event. */
    private Map<TrafficLight, SimEventInterface<SimTimeDoubleUnit>>
        redEvents = new LinkedHashMap<>();

    /** Scheduled green event. */
    private Map<TrafficLight, SimEventInterface<SimTimeDoubleUnit>>
        greenEvents = new LinkedHashMap<>();

    /**
     * @param simulator OTSSimulatorInterface; simulator
     * @param trafficLights List&lt;TrafficLight&gt;; traffic lights
     * @param compatible Compatible; GTU types that trigger the detector,
         and hence the light to red
     */
    public CycleTimeLightController(final OTSSimulatorInterface simulator,
        final List<TrafficLight> trafficLights,
           final Compatible compatible)
    {
        this.simulator = simulator;
        this.trafficLights = trafficLights;
        for (TrafficLight trafficLight : trafficLights)
```

```
    {
        Try.execute(() -> new RampMeteringSensor(trafficLight,
            simulator, compatible),
                "Unexpected exception while creating a detector with a
                    ramp metering traffic light.");
        this.greenStarts.put(trafficLight,
            Time.createSI(Double.NEGATIVE_INFINITY));
    }
}

/** {@inheritDoc} */
@Override
public void disable()
{
    Iterator<TrafficLight> it = this.redEvents.keySet().iterator();
    while (it.hasNext())
    {
        TrafficLight trafficLight = it.next();
        this.simulator.cancelEvent(this.redEvents.get(trafficLight));
        it.remove();
    }
    it = this.greenEvents.keySet().iterator();
    while (it.hasNext())
    {
        TrafficLight trafficLight = it.next();
        this.simulator.cancelEvent(this.greenEvents.get(trafficLight));
        it.remove();
    }
    this.enabled = false;
    for (TrafficLight trafficLight : this.trafficLights)
    {
        trafficLight.setTrafficLightColor(TrafficLightColor.GREEN);
    }
}

/**
 * Starts the cycle.
 * @param cycleTime Duration; cycle time
 */
@Override
public void enable(final Duration cycleTime)
{
    SimLogger.always().info("Traffic light uses " + cycleTime);
    this.cTime = cycleTime;
    if (!this.enabled)
    {
        this.enabled = true;
        for (TrafficLight trafficLight : this.trafficLights)
        {
```

```
                setGreen(trafficLight);
            }
        }
    }

    /**
     * Sets the traffic light to red. Can be scheduled.
     * @param trafficLight TrafficLight; traffic light
     */
    protected void setRed(final TrafficLight trafficLight)
    {
        this.redEvents.remove(trafficLight);
        SimLogger.always().info("Traffic light set to RED");
        trafficLight.setTrafficLightColor(TrafficLightColor.RED);
    }

    /**
     * Sets the traffic light to green. Can be scheduled and remembers the
         green time.
     * @param trafficLight TrafficLight; traffic light
     */
    protected void setGreen(final TrafficLight trafficLight)
    {
        this.greenEvents.remove(trafficLight);
        this.greenStarts.put(trafficLight,
            this.simulator.getSimulatorTime());
        SimLogger.always().info("Traffic light set to GREEN");
        trafficLight.setTrafficLightColor(TrafficLightColor.GREEN);
    }

    /** Ramp metering sensor. */
    private class RampMeteringSensor extends AbstractSensor
    {

        /** */
        private static final long serialVersionUID = 20190618L;

        /** The traffic light. */
        private final TrafficLight trafficLight;

        /**
         * @param trafficLight TrafficLight; traffic light
         * @param simulator TimeDoubleUnit; simulator
         * @param detectedGTUTypes Compatible; GTU types
         * @throws NetworkException when the position on the lane is out of
             bounds
         */
        RampMeteringSensor(final TrafficLight trafficLight, final
            TimeDoubleUnit simulator, final Compatible detectedGTUTypes)
```

```
        throws NetworkException
{
    super(trafficLight.getId() + "_sensor", trafficLight.getLane(),
        trafficLight.getLongitudinalPosition(),
            RelativePosition.FRONT, simulator, detectedGTUTypes);
    this.trafficLight = trafficLight;
}

/** {@inheritDoc} */
@SuppressWarnings("synthetic-access")
@Override
protected void triggerResponse(final LaneBasedGTU gtu)
{
    if (CycleTimeLightController.this.enabled &&
        this.trafficLight.getTrafficLightColor().isGreen())
    {
        try
        {
            // schedule green
            Time minRedTime = CycleTimeLightController.this.simulator
            .getSimulatorTime().plus(MIN_RED_TIME);
            Time cycleRedTime =
                CycleTimeLightController.this.greenStarts.get(this.trafficLight)
                    .plus(CycleTimeLightController.this.cTime);
            Time green;
            if (minRedTime.ge(cycleRedTime))
            {
                SimLogger.always().info("Traffic light set to RED");
                this.trafficLight.setTrafficLightColor(TrafficLightColor.RED);
                green = minRedTime;
            }
            else
            {
                SimLogger.always().info("Traffic light set to YELLOW
                    (RED over 'MIN_RED_TIME')");
                this.trafficLight.setTrafficLightColor(TrafficLightColor.YELLOW);
                CycleTimeLightController.this.redEvents.put(this.trafficLight,
                        CycleTimeLightController.this.simulator
                        .scheduleEventRel(MIN_RED_TIME, this,
                                CycleTimeLightController.this,
                                    "setRed", new Object[] {
                                    this.trafficLight }));
                green = cycleRedTime;
            }
            CycleTimeLightController.this.greenEvents.put(this.trafficLight,
                    CycleTimeLightController.this.simulator.scheduleEventAbs(green,
                        this, CycleTimeLightController.this,
                            "setGreen", new Object[] {
                                this.trafficLight }));
```

```
                }
                catch (SimRuntimeException exception)
                {
                    throw new RuntimeException(exception);
                }
            }
        }

        /** {@inheritDoc} */
        @Override
        public AbstractSensor clone(final CrossSectionElement newCSE,
                final nl.tudelft.simulation.dsol.simulators.SimulatorInterface
                .TimeDoubleUnit newSimulator)
                throws NetworkException
        {
            return null; // TODO: should be cloned as part of the ramp
                metering
        }

    }

}
```