MSc thesis in Embedded Systems

Evaluation of lane change predictions for schedule-driven intersection control

Lourens Pool
2021

Evaluation of lane change predictions for schedule-driven intersection control

by

Lourens A. Pool

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday May 3, 2021 at 13:00.

Student number:	4084861	
Project duration:	September 3, 2018 – May 3, 2021	
Thesis committee:	Dr. F.A. Oliehoek,	TU Delft, thesis advisor
	Dr. A. Czechowski,	TU Delft, daily supervisor
	Prof. Dr. C.M. Jonker,	TU Delft
	Dr. ir. M. Wang,	TU Delft

TU Delft

Preface

In front of you lies my thesis, which marks the long-awaited end point of my studies. After my Bachelor's in Industrial Design Engineering, I enjoyed a Master's programme at the faculty of Computer Science. During the end of my Master's programme, I developed an interest in the topic of this thesis, i.e. applying Machine Learning to solve practical problems. I still find it fascinating that just by observing past human behaviour, Machine Learning models are able to learn and predict future human behaviour. I hope to see more applications of this general topic, which could benefit society in a sound manner.

Although this thesis is just a small step into the direction of that topic, I hope that my work can contribute to research on traffic light control, by building upon the demonstrated insights. And by having re-implemented Schedule-driven Intersection Control [1] (SCHIC) I hope to have lowered the threshold for someone else to improve this traffic light controller.

I am grateful for the guidance of Frans and Aleksander, who have been my supervisors throughout this project. Frans showed me the approach to scientific research and objectively evaluated my work which led to some neat improvements. Last but certainly not least, Aleksander has always been of great support to me. Both during the practical parts of the thesis, as well as by keeping an eye on the bigger picture of the project. I would like to thank Frans and in particular Aleksander for thoroughly proofreading earlier versions of this thesis. Both Frans and Aleksander have also been able to put the results of this work into perspective, which boosted my motivation.

Finally, I would like to thank my parents, family and friends for their everlasting support during my studies.

Lourens Pool
Delft, April 2021

Abstract

Optimization of traffic signal control has been widely investigated by means of model-based strategies. In 2012 a new model-based controller was published, named Schedule-driven Intersection Control (SCHIC). This controller uses a job-scheduling algorithm to minimize the cumulative delay for all observed vehicles. The algorithms of SCHIC are at the basis of the commercially deployed controller SURTRAC. This means that SCHIC is partly used in real-world traffic situations and its algorithms are well documented. Therefore, it was possible to reproduce this controller in this thesis.

Previous literature often evaluated SCHIC in simplified scenarios with (1) unrealistically large look-ahead times for vehicle detection. Secondly, (2) in most scenarios all approaching vehicles were restricted to request a single turning-movement at the intersection thereby removing inter-phase uncertainty (as explained directly below). Inter-phase uncertainty occurs when the chosen entry-lane of a vehicle determines the green phase that it requires from the traffic light controller. So, inter-phase uncertainty occurs under the following two conditions. *Condition 1* – an approaching vehicle must eventually choose between different entry-lanes, with each entry-lane being associated to a different turning-movement at the intersection. *Condition 2* – each of these individual entry-lanes is serviced during a different and competing green phase. For example, inter-phase uncertainty occurs when two competing green phases service *either* the straight-ahead or the left-turning entry-lane.

The aim of this thesis is to evaluate SCHIC with both (1) realistic look-ahead times and (2) inter-phase uncertainty, since these two uncertainties are often not found in previous literature, however, they are commonly found in reality. Thus, it is not directly clear how the reported results would translate to more realistic settings, which include these two common uncertainties. In this study SCHIC is re-implemented and evaluated on a typical Dutch urban intersection located in The Hague. The lane dimensions of this intersection determine the look-ahead times as used in this thesis.

For this study, a new method is introduced to reduce inter-phase uncertainty which is inspired by lane change prediction modules for automated driving. This method, Lane Change Prediction (LCP), makes use of e.g. radar-equipment in order to observe: (1) the driving style of each approaching vehicle and (2) the upcoming traffic state on each reachable entry-lane. Based on these two observations a classification model, i.e. a Random Forest, predicts the desired entry-lane before the vehicle actually enters it. This thesis researches whether the potentially higher accuracy of LCP will improve SCHICs performance as compared to its simpler statistical baselines, *fixed-label* and *split-vehicles*, which could have a lower accuracy.

All traffic scenarios are simulated with Simulation of Urban Mobility (SUMO). Two popular driver-models are evaluated which determine the dynamic behaviour of the vehicles in simulation, i.e. the Intelligent Driver Model (IDM) and Krauss driver-model.

The results showed that despite its higher accuracy, LCP does not significantly improve SCHICs performance over its baseline methods *fixed-label* and *split-vehicles*. Furthermore, the maximum theoretical performance gain of using LCP is evaluated with perfect predictions (*Oracle*). When applying perfect predictions, the average speed of all vehicles increased by 0.65% and 0.51% for the Intelligent Driver Model (IDM) and Krauss driver-model, respectively. The average number of stops per vehicle was reduced by 1.53% and 1.12% for the IDM and Krauss driver-model, respectively.

Overall, it is concluded that the performance potential of all methods, i.e. the {*Oracle, split-vehicles, fixed-label and LCP*}, is quite comparable on SCHICs traffic flow. These methods show some performance potential, however, their confidence intervals are relatively wide which draws them closer to each other. In our scenario, the intersection in The Hague, another method might increase SCHICs traffic flow more significantly.

Contents

Chapter 1

Introduction

Urban traffic congestion is a well known problem and affects road users across the world, therefore, in this thesis a focus is placed on adaptive traffic light control. The number of motorized road vehicles in The Netherlands increased in early 2019 to a grand total of 12.7 million vehicles. Out of this grand total, passenger cars comprise two thirds by reaching 8.5 million registered passenger cars [2]. Congested traffic in general leads to several cost-factors, for the European Union these costs are estimated to be 1% of its GDP [3]. Fortunately, automated traffic light control has the potential to reduce congestion, thereby saving time and money of road users. With urban traffic on the rise, improvements to traffic light optimization increase in value [4].

The actual performance and weaknesses of commercial traffic light controllers are not always clear. A large number of traffic control methods have been proposed in the past, and some of these methods made it into commercial products. Since these commercial controllers often require a hefty financial investment, researchers do not always have access to them. As a result it is not always clear how well these methods perform and which weaknesses exist. However, in order to reduce everyday travel-times it is important to evaluate the weak-points of these commercial controllers.

This thesis takes a practical approach by evaluating the performance of a controller named Schedule-driven Intersection Control [1] (SCHIC). In short, SCHIC uses online job-scheduling to minimize the cumulative delay for all observed vehicles. Xie et al. [1] demonstrated that SCHIC outperformed the approximate dynamic programming procedure Combined Optimization of Phases [5] (COP). Given the fact that SCHIC outperformed COP, Xie et al. [1] concluded that SCHIC is a state-of-the-art or at least promising novel method.

Furthermore, SCHICs algorithms are at the basis of the recently commercially deployed controller named SURTRAC. This means that SCHIC is used in real world traffic situations and therefore interesting to evaluate. Its algorithms are well-documented, however, an implementation is unavailable. Therefore, a re-implementation of SCHIC is realized for this thesis.

Inter-phase uncertainty

Before describing the aim of this thesis, the problem of phase based uncertainties is explained in this section. Dhamija [6] states that turn-induced uncertainty can result in two forms of uncertainty for a traffic light controller, namely intra -and inter-phase uncertainty. Both forms of uncertainty and the conditions to remove the uncertainty all together are visualized and explained directly below.

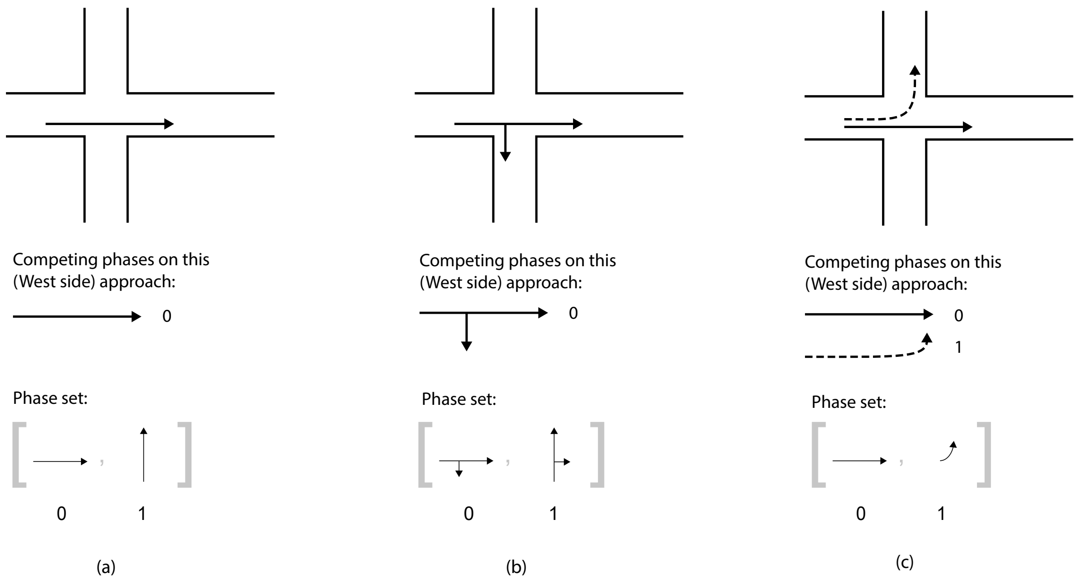


Figure 1.1: Phase based uncertainties for vehicles which approach on the South -or West side approach. These two approaches are enough to define the uncertainties. The phase set contains all possible green phases which the controller can select. (a) No uncertainty; (b) Intra-phase uncertainty; (c) Inter-phase uncertainty. Image based on Dhamija [6].

See Figure 1.1
A vehicle arrives at the intersection originating from a single approach (e.g. the West or South approach), whilst travelling on just a single *entry-lane*. An *entry-lane* is defined as a portion of the pavement which is allocated to a single line of traffic which originates from the same direction, as often indicated by painted longitudinal lines or markers. Please note that each approach towards the intersection can contain multiple entry-lanes, but can also contain just a *single* entry-lane.

The phase set contains all possible green phases which the controller can select. During a green phase, an entry-lane (or multiple entry-lanes) receive a green signal so vehicles on this entry-lane can leave the intersection on the *exit-lane(s)* as indicated by the arrow(s). The characteristics of a phase set, combined with, the amount of entry-lanes per approach towards the intersection determine which uncertainties are present for a traffic light controller.

- ◇ **No phase based uncertainty.** Figure 1.1a visualizes two conditions for a phase set in order to eliminate turn-induced uncertainty. *Condition 1* – on each approach of the intersection, there is only a single entry-lane leading towards the intersection, this single entry-lane is serviced by a single green phase. *Condition 2* – during each green phase vehicles are restricted to leave the intersection on only one exit-lane. In our example, each approaching vehicle can only request the green phase for straight-ahead traffic. In conclusion, each approaching vehicle can be detected far away from the intersection and its green phase and exit-lane are known in advance.

- ◇ **Intra-phase uncertainty.** Figure 1.1b visualizes intra-phase uncertainty, which can influence a communication mechanism among intersections – if a communication mechanism is actually in place.

  Intra-phase uncertainty is an uncertainty within a single green phase and defined as follows: during a green phase vehicles on a single entry-lane can choose to leave the intersection on multiple exit-lanes. For example, this occurs when vehicles can choose to either continue straight-ahead or turn right during the same green phase (Figure 1.1b). When one of the green phases is active, intra-phase uncertainty only affects the uncertainty on the number of vehicles per individual exit-lane.

- ◇ **Inter-phase uncertainty.** Figure 1.1c visualizes inter-phase uncertainty. Which occurs under the following two conditions. *Condition 1* – an approaching vehicle must choose between different entry-lanes. *Condition 2* – each of these different entry-lanes is serviced during a different green phase. For example, inter-phase uncertainty occurs when two entry-lanes are present on a single approach; one for straight-ahead traffic and one for left-turning traffic, when each *(single)* entry-lane is serviced during two different green phases. See Figure 1.1c.

Aim of this study

Knowledge gap SCHIC

It is not completely clear how the reported results of SCHIC translate when including (1) realistic look-ahead times and (2) inter-phase uncertainty. These two uncertainties are often not considered in previous evaluations of SCHIC, whereas they are commonly found in reality. For example, one scenario of Xie et al. [1] considered a single isolated intersection which uses a look-ahead time of 70 *seconds*. On this isolated intersection, the phase set only contained straight-ahead traffic (see Figure 1.1a), thereby removing inter-phase uncertainty. Secondly, Xie et al. [1] omitted inter-phase uncertainty on another series of adjacent intersections (Arterial network), where they reported results of SCHIC with vehicle detection at 10 *seconds* away from each of the intersections. Specifically, with the proposed look-ahead time of 10 *seconds* the complete phase set of Xie et al. [1] only contained two green phases for straight-ahead and right-turning traffic, as shown in Figure 1.1b.

When SCHIC needs to deal with inter-phase uncertainty, the proposed look-ahead time of 10 *seconds* by Xie et al. [1] is quite unrealistic. To clarify, at 10 *seconds* from the intersection most vehicles will still change lanes, depending on their desired turning-movement and thereby requesting a different green phase. Therefore, the aim of this thesis is to evaluate SCHIC with both (1) realistic look-ahead times and (2) inter-phase uncertainty.

Up until now three baseline methods have been described that can increase SCHICs look-ahead time whilst handling inter-phase uncertainty, namely split-vehicles [1], [7], fixed-label [8] and Vehicle to Intersection Communication (V2I) [7]. However, their performance on SCHICs traffic flow is *not* well reported. These baseline methods are described directly below. Afterwards, SCHIC is described in more detail and a new method is introduced: Lane Change Prediction (LCP). LCP could potentially reach higher predictive accuracies than the existing statistical baseline methods (i.e. split-vehicles and fixed-label). Thereby, LCP could also bring more traffic flow performance to SCHIC than its existing statistical baseline methods.

Baseline approaches for modelling inter-phase uncertainty

Several methods have been proposed in past literature to efficiently cope with inter-phase uncertainty. These methods detect vehicles quite far from the intersection and consider the possibility that each detected vehicle will eventually request a different green phase. For SCHIC, two methods have been proposed which use historic statistical properties that are observed in the distribution of vehicles per turning-movement. Both statistical methods increase the look-ahead time by estimating or predicting a vehicle's desired turning-movement. For example, the methods can be applied on a detected vehicle before it will actually enter one of the (short) entry-lanes associated to a single turning-movement. The two statistical methods and a third one, are listed below:

- ◇ *split-vehicles [1], [7]* - as proposed by Xie et al. [1], this method splits each approaching vehicle into a fraction for each turning-movement weighted by the probability per turning-movement.
- ◇ *fixed-label [8]* - as based on a field-test of SCHIC in 2015 and as proposed by Smith [8]. Smith observed a dominant green phase on his intersections, i.e. most vehicles chose this single dominant green phase for continuing straight-ahead over any other green phase. In this dominant scenario a *fixed-label* is used for each new vehicle, i.e. upon first detection, each vehicle is assumed to request the dominant (straight-ahead) green phase at the intersection.
- ◇ *Vehicle to Intersection Communication (V2I) [7]* - inter-phase uncertainty can be removed for vehicles which use Vehicle to Intersection Communication (V2I). Since a vehicle can now communicate his desired green phase directly to the traffic light controller.

Recall that the three methods above are well described in previous literature, however, their traffic flow performance on SCHIC is *not* well reported.

SCHIC

In this thesis, the focus is limited to the influence of inter-phase uncertainty on SCHICs local planning, as explained below. As input, SCHIC requires the following information per individual vehicle: (1) its arrival time at the intersection and (2) the green phase that this vehicle will request. When inter-phase uncertainty is present, each allowed entry-lane (of a single approach) is serviced during a different green phase. This means that approaching vehicles can only be associated to a single green phase when they are close to the intersection. Detecting vehicles closer to the intersection reduces the look-ahead time, which is defined as: $look\_ahead\_time_{detector}[s] = distance_{detector}[m]\ /\ speed\ limit[m/s]$.

In general, reducing the look-ahead time of SCHIC reduces its performance [1]. So, it is desired to detect vehicles as far away from the intersection as possible. However, increasing this distance also increases the uncertainty on the desired green phase of each detected vehicle, when inter-phase uncertainty is present. Therefore, inter-phase uncertainty can affect the local planning of SCHIC, when vehicles are detected so far away from the intersection that they are likely to change to a different entry-lane and thereby requesting a different green phase.

In general, one can identify three main components of SCHIC – the traffic input information, the job-scheduler and the coordination mechanism among intersections. While all components could be improved, the focus in this study is limited to local intersection scheduling without a coordination mechanism between neighbouring intersections. Several weaknesses of SCHIC have been identified by those who re-implemented SCHIC. Most of these studies outperformed SCHIC with various commercial and/or simpler controllers, whereas some improved SCHICs logic [6], [9]–[17]. In this study an effort is made to improve SCHIC on the traffic input level.

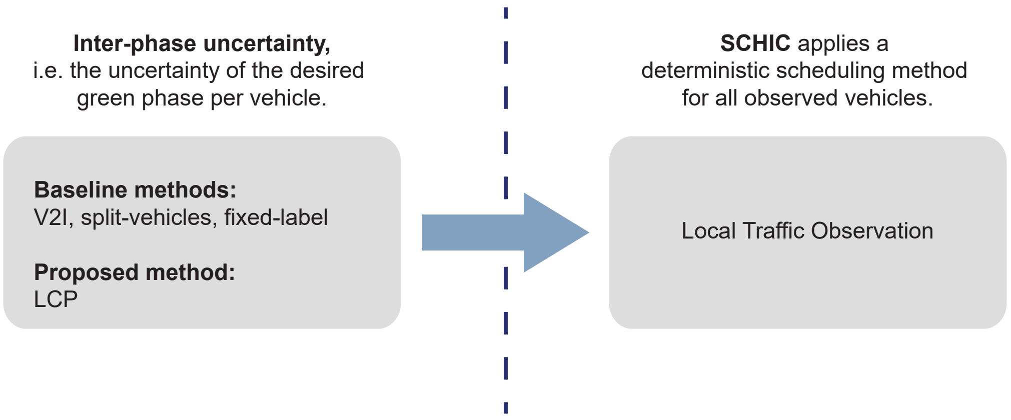

Inter-phase uncertainty,
i.e. the uncertainty of the desired green phase per vehicle.
SCHIC applies a deterministic scheduling method for all observed vehicles.
Baseline methods:
V2I, split-vehicles, fixed-label
Proposed method:
LCP
Local Traffic Observation

Figure 1.2: Coupling of inter-phase uncertainty methods to SCHICs local traffic observation.

Proposed method LCP

In this study, a novel approach with respect to SCHIC is introduced to extend the look-ahead time for vehicle detection while handling inter-phase uncertainty, namely: Lane Change Prediction (LCP). LCP is designed and applied here on a lane-split scenario. A lane-split leads the approaching vehicle (ego-vehicle) to multiple short entry-lanes as shown in Figure 1.3b. LCP makes use of e.g. radar-equipment in order to observe: (1) the driving style of an approaching ego-vehicle and (2) the upcoming traffic state on each of its reachable entry-lanes (see Figure 1.3b).

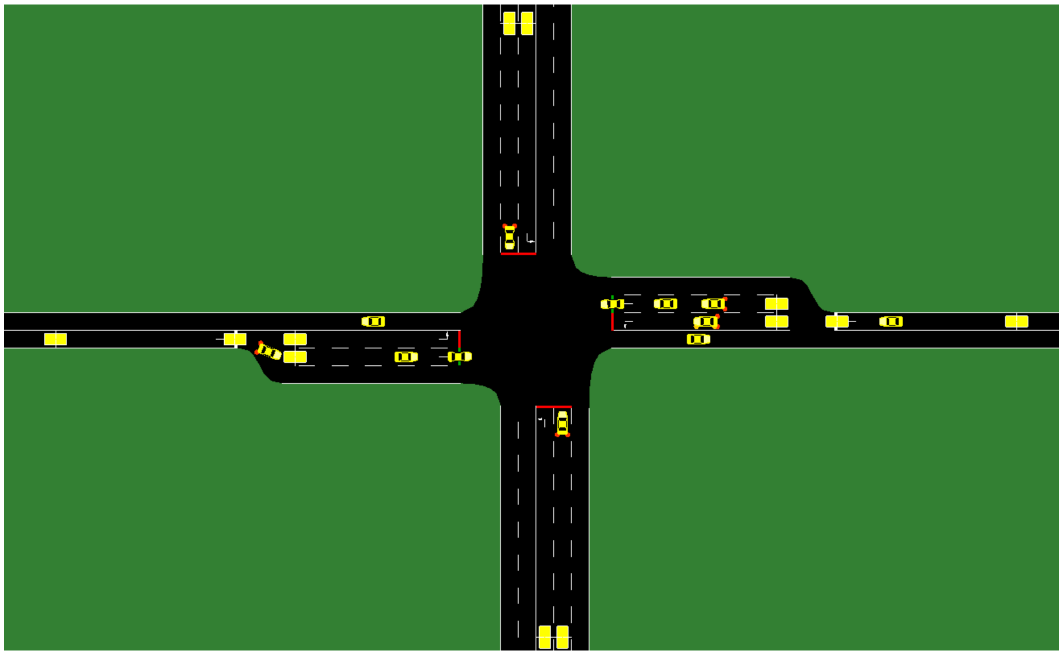
This method is based on the following assumption: each vehicle which is driving towards but has not yet entered one of the short entry-lanes; will show anticipative clues in its driving style to indicate its desired short entry-lane. For example, different anticipative driving behaviour is expected for entering either the left-turning or straight-ahead entry-lane. Furthermore, this is assumed when only one of these entry-lanes is receiving a green signal and the other a red signal, see Figure 1.3b.

LCP is a classification model that predicts mandatory lane changes as based on the observed driving style of each vehicle which approaches a lane-split. LCP extends its data input compared to the statistical baseline methods of *split-vehicles* and *fixed-label*, since LCP also considers the anticipative driving style. Therefore, LCP could achieve higher predictive accuracies than the statistical baselines. This thesis researches whether the potentially higher accuracies of LCP increase the traffic flow performance of SCHIC as compared to its statistical baseline methods (i.e. *split-vehicles* and *fixed-label*).

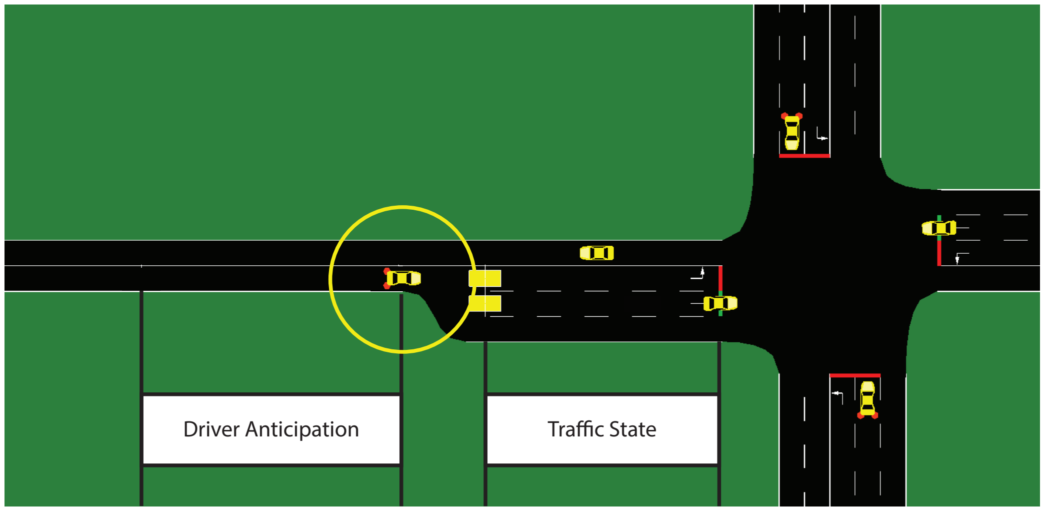
An intrinsic limitation of LCP is the fact that each prediction can only be made for the near future, since it is limited by the future time-frame on which each driver anticipates. I.e. human drivers only show anticipative driving behaviour for a traffic state which they will reach in a short period of time. Ortiz et al. [18] based and verified their predictions on real-world vehicle dynamics in an inner-city scenario, where vehicles approach an inner-city traffic light. The predictive model is limited to output the following future behaviour: [”keep speed”, ”braking”, ”stopped”]. The system of Ortiz et al. can warn drivers when they do not brake in time, for example to warn a driver which needs to *brake* since he/she is approaching other (still standing) vehicles with a velocity which is too high.

Although this model does not predict an actual lane change, it is expected to fit well in the design of LCP. When LCP is applied with our phase set, one of the two reachable entry-lanes is always receiving a red signal, whereas the other entry-lane is currently serviced with a green signal. The predictive output of ”stopped” is expected to correlate highly with the entry-lane which is receiving the red signal as vehicles will need to perform a stop on that entry-lane.

Ortiz et al. propose to apply their predictive model for a future time-frame of up to 3.0 *seconds*, since the model is highly accurate up until this range. So, based on the results of Ortiz et al. [18], Lane Change Prediction (LCP) can theoretically achieve high accuracies up to 3.0 *seconds* before a vehicle reaches the stop-line of the intersection. Although LCP can only increase the look-ahead time of SCHIC with a limited amount, the relative increase from 2.2 *seconds* to 3.0 *seconds* is still large.



(a) Overview of intersection with two lane-splits, located in The Hague.



(b) The final entry-lane of the vehicle inside the yellow circle is still unknown. However, its final entry-lane can be obtained now by SCHICs baseline methods, or by making a Lane Change Prediction.

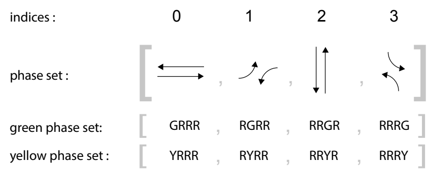
Figure 1.3: Inter-phase uncertainty in SCHIC.

Experimental setup

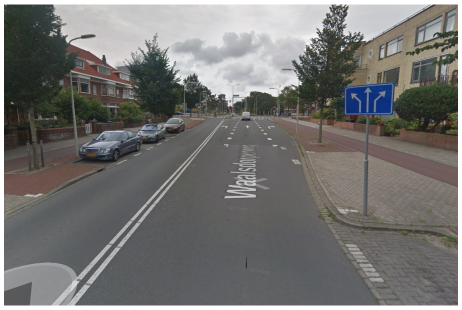
The practical approach of this thesis is pursued by evaluating SCHIC on a simplified Dutch intersection which is located in The Hague. First of all, inter-phase uncertainty is considered since straight-ahead and left-turning traffic (each having a separate entry-lane) are serviced in different and contending green phases as shown in Figure 1.4a. Secondly, the dimensions of the entry-lanes as used in this study match with the dimensions of this typical real-world intersection. The intersection contains two lane-splits, which lead vehicles to short ($30m$) entry-lanes for both allowed turning-movements at the intersection, one of these lane-splits is shown in Figure 1.4b.

Lane-based vehicle detection as required for SCHIC, can't be performed beyond the length of these shorter $30m$ entry-lanes. This leads to a more realistic look-ahead time of 2.2 *seconds* on these shorter entry-lanes. As for the longer lanes at this intersection which do not contain a lane-split, vehicle detection can be performed further away from the intersection. However, in reality, on these longer lanes vehicle detection is only performed at $40m$ from the intersection.

Especially for intersections located in urban areas, limited lane-dimensions are quite often a fact of life. Hence, the look-ahead time per entry-lane will remain limited as well given the following trade-off. This trade-off mainly balances on three opposing values: (1) the expected traffic volumes per individual entry-lane (2) the availability of additional ground space (3) the cost of extending individual entry-lanes and their associated infrastructure. Given the complexity of extending entry-lanes in the real world, it is quite interesting to evaluate in which manner the logic of traffic light controllers can be improved to handle limited lane-dimensions more efficiently.



(a) An often encountered phase set in The Netherlands, also the default phase set of Simulation of Urban Mobility [19] (SUMO).



(b) A drivers perspective on one of the lane-splits which is found on the intersection which is evaluated in this thesis. Source: Google Street View, accessed on 07-01-2020.

Figure 1.4: Scenario design.

Focus and scope

Given the focus on inter-phase uncertainty, two main simplifications are applied in this study. Firstly, pedestrian and bicycle traffic are omitted, i.e. only car traffic is considered. Secondly, we omit right-turning traffic (which can introduce intra-phase uncertainty), since we only focus on inter-phase uncertainty. SCHICs local intersection optimization is evaluated on this simplified version of the intersection. The above two simplifications are in-line with the previously mentioned knowledge gap and scope of this study.

The scope of this study on LCP is limited to simulated driving behaviour since there is no access to real-world vehicle dynamics. Simulation of Urban Mobility [19] (SUMO) is used to obtain the simulated driving behaviour. The choice for this simulator is based on two reasons. First and foremost, traffic simulators can be classified as macroscopic or microscopic depending on the manner in which they model vehicle dynamics. Since SUMO is a microscopic traffic simulator, it is able to to model highly detailed dynamic behaviour of each driver which can lead to valuable clues for informed predictions. Secondly, SUMO is a popular software package among researchers which is beneficial both during and after the performed study.

Research Questions

The following research questions are addressed, which are derived from the knowledge gap and proposed method of LCP. As stated before, the following knowledge gap is present for SCHIC: SCHICs performance is *not* well reported for (1) limited and realistic look-ahead times combined with (2) inter-phase uncertainty.

1. Given a typical Dutch intersection with limited lane-dimensions and hence limited look-ahead times, how does SCHICs local performance relate to a simpler but commonly applied Time-Gap Controller?
2. What is the performance potential for SCHIC when perfect Lane Change Predictions are applied on this typical Dutch intersection with two lane-splits?
3. How accurate are Lane Change Predictions (LCPs) in SUMO?
4. How does SCHIC in combination with LCP perform in comparison to the existing statistical baseline methods of *split-vehicles* and *fixed-label*?

Outline of this thesis

This thesis is structured as follows. Chapter 2 starts by discussing general related work on responsive traffic light controllers, before shifting the focus to related work on SCHIC. Lastly, that chapter also further explains the baseline methods to handle inter-phase uncertainty. Chapter 3 gives a theoretical framework of SCHIC by explaining its algorithms in a high-level overview. This explanation of SCHIC is helpful, since a clear understanding of SCHIC enables a clear understanding of the coupling of LCP to SCHIC. Chapter 4 introduces our design of the proposed method of LCP. The classification accuracy of Lane Change Prediction (LCP) is given in Chapter 5. Chapter 6 shows the traffic flow improvements of SCHIC in combination with its existing baseline methods and the novel method of LCP. Lastly, Chapter 7 brings this work to a close by answering the research questions and by discussing suggested improvements and extensions to this work.

Chapter 2

Related work

This work is rooted in the field of adaptive traffic light control. Specifically, we focus on a controller named – Schedule-driven Intersection Control [1] (SCHIC) – when it is applied on an urban intersection with limited lane-dimensions and inter-phase uncertainty (as explained in the next paragraph). Although limited lane-dimensions and inter-phase uncertainty are commonly found on Dutch urban intersections, they are often omitted in previous research.

Most previous work on SCHIC does not consider inter-phase uncertainty, recall that inter-phase uncertainty is explained prior in subsection 1 on page 7. Without inter-phase uncertainty, each approaching vehicle is unable to change its entry-lane and each entry-lane is associated to just a single green phase. Therefore, the green phase of each approaching vehicle is fixed beforehand, this enables SCHIC to detect vehicles for each green phase already long in advance. Using these large look-ahead times increases the performance of SCHIC [1].

In this chapter, we first give a general background on adaptive traffic light control which sketches a context around SCHIC. Afterwards the related work on SCHIC starts with general improvements to this controller. But these improvements are still presented under the restriction that each vehicle is unable to change its entry-lane, thereby only able to request a single green phase (i.e. the removal of inter-phase uncertainty). Subsequently, three existing baseline methods are discussed which do consider inter-phase uncertainty. These three baseline methods increase the look-ahead time for vehicle detection, although each detected vehicle is still able to perform a lane change towards a different entry-lane and thereby able to request a different green phase from SCHIC.

Section 2.1 covers a brief history of some important (commercial) traffic light controllers. Secondly, Section 2.2 shifts the focus towards SCHIC by summarizing several works which demonstrate a weakness of this controller, whilst omitting inter-phase uncertainty. Section 2.3 explains three existing baseline methods which can perform vehicle detection given the fact that inter-phase uncertainty is present. These baseline methods detect vehicles while they are still able to change to a different entry-lane and thereby able to request a different green phase. Lastly, Section 2.4 briefly concludes this chapter and based on the identified knowledge gaps, a motivation is presented for the novel proposed method of Lane Change Prediction (LCP).

Relevant terminology
Most terms which are used in this chapter are also explained and listed in the glossary on page 96.

2.1 General background on adaptive traffic light control

Firstly, this section covers a brief history of some important (commercial) traffic light controllers and sketches the general context to support the next section which covers related work on SCHIC.

In 1964 Dunne [20] introduced a linear actuated control algorithm which directly responds to observed traffic by applying simple linear control rules. This method of Dunne [20] can be classified as a Vehicle Actuated Control (VA) method. A theoretical analysis, under the assumption of constant vehicular arrival and departure rates in unsaturated conditions, demonstrates that this algorithm delivers a stable performance. With the right choice of control parameters, the algorithm minimizes the average delay of all vehicles. Even to this date simple VA is often optimal for isolated intersections [1].

Current versions of VA are often based on time-gaps. This Time-Gap Controller works by extending each current green phase as long as a continuous flow of traffic is detected for it. More precisely, the next green phase is selected when a sufficient time-gap is observed after the last vehicle has left in the current green phase. As a result, this method clears the queues which are found for each active green phase.

Although the former methods are effective, their use of fixed parameter settings – like time-gap or maximum allowed time per green phase – might be sub-optimal at times. Based on insights found in historic traffic routes information, one can apply predictive modelling for traffic congestion in order to make more informed decisions. In general, these predictions allow to tune parameters of traffic light controllers in real-time in order to increase their performance.

The value of real-time parameter tuning based on predictive modelling of traffic is particularly seen during a special event like a concert, since this event will cause high traffic volumes. The set of routes which are affected are quite predictable given the event location and road network. A method as proposed by Pescaru in 2013 [21] demonstrates the value. With Pescaru's method [21], each controller will still react directly to its observed vehicles but is also able to predict two important variables. First of all a prediction can be made on the total queue length. Secondly, it is important to predict the set of roads which will influence traffic congestion most in the next time-frame [21]. Both predictions can be used to tune parameters of common traffic light controllers. For example by allowing a longer maximum green time to selected lanes, in order to fully clear a queue of vehicles on these selected lanes.

Intersection controllers which only take local observations into account are prone to make sub-optimal decisions which only look good locally but not globally. Ideally, the different controllers ensure smooth travel of vehicles throughout large urban regions. Coordination mechanisms between traffic lights are of utmost importance, some popular mechanisms will be given here.

The simplest coordination method might be a *fixed time controller* where each intersection uses an *offset* time to each neighbouring intersection to compensate for the travel time between them. Cycle and split are two important terms, which are now explained. Cycle is defined as, the total time it takes for all phases of one intersection to receive their allocated green -and inter-green times. Split is related to cycle and defined as, the amount of time within a phase where a traffic signal is green. With a *fixed time controller* each green phase has a fixed duration, i.e. a fixed split of the total cycle time. It is only possible to set a single *offset* time, and therefore required to have an equal travel time (distance / free-flow speed) between all direct neighbouring intersections. Some real-world grid networks found in the United States comply to this single travel-time requirement, since all buildings are placed in square blocks. And these square blocks are connected by roads that run parallel to them. Doing so, the travel-times between all direct neighbouring intersections is identical [22].

Since traffic demands vary throughout the day, the use of fixed times for *split, cycle and offset* are not ideal. In 1981 a popular commercial system has been introduced which adjusts these parameters in real-time, hence the name SCOOT (Split-Cycle Offset Optimization Technique [23]). The system obtains real-time information on traffic volumes by means of sensors which are embedded in each road. This information is used to make changes to the split and cycle times. The *offsets* are adjusted with respect to intersections that belong to a region, i.e. a fixed set of connected intersections.

An auction-based system as presented by Raphael [24] forms ”coalitions” of intersections dynamically instead of relying on pre-determined coalitions. In each coalition, traffic light controllers work together to improve the observed traffic flow. A market-based mechanism observes the changes in traffic flow over time and updates the coalitions of traffic lights for a smoother flow of traffic throughout the network. This method has been demonstrated to outperform SCOOT which uses pre-defined coalitions.

2.2 Related work on Schedule-driven Intersection Control (SCHIC)

While SCHIC is at the basis of the commercial controller SURTRAC, it is also precisely documented in a publication [1] which allows others to reproduce and study it. A literature study on SCHIC has been performed which resulted in three main papers which identify a weakness of SCHIC. Some authors improve SCHIC in order to reduce the identified weakness, whereas others demonstrate a second method which outperforms SCHIC.

The following three papers are further explained in Appendix A, and briefly summarized in this section.

- In 2019, Dang [14] introduced lane-based job-scheduling for SCHIC which improves the accuracy of SCHICs cost-function.
- In 2018, Goldstein and Smith [7] introduced ERIS. Which uses a job-scheduling algorithm like SCHIC, however it applies a more flexible phase set than the one originally proposed for SCHIC [1].
- In 2019, Hu and Smith [25] demonstrated SCHICs poor performance during saturated traffic conditions. They proposed a method which gradually shifts from SCHICs job-scheduling control to queue-based control when the traffic demand grows to a saturated demand.

Dang [14] improved the accuracy of SCHICs cost-function for a common real-world occurrence, by considering that vehicles can approach an intersection on opposing entry-lanes whilst sharing the same green phase. Dang [14] has shown considerable improvements to SCHICs performance, but omitted inter-phase uncertainty. To clarify, inter-phase uncertainty is omitted since each vehicle is restricted to approach the intersection on a single entry-lane (for straight-ahead traffic) and this single entry-lane is associated to a single green phase for straight-ahead traffic. With inter-phase uncertainty removed vehicle detection can be performed at the beginning of the lane, since each vehicle on this lane will request the same green phase from the controller. Therefore, the length of each lane is now the only limiting factor to maximize the look-ahead time for vehicle detection. In Dangs [14] 5x5 grid network, vehicles on 20 out of the 25 lanes are are detected at at least 7.5 seconds from the intersection. This look-ahead time is reasonable given that inter-phase uncertainty is eliminated. However, when vehicles are able to request different and competing green-phases by switching lanes, inter-phase uncertainty is quite an important factor to consider at 7.5 seconds from the intersection.

Goldstein and Smith [7] include both straight-ahead and left-turning traffic in their proposed job-scheduler: ERIS. However, inter-phase uncertainty is removed since all vehicles communicate their desired entry-lane and hence green phase directly to the traffic light controller. Furthermore, all vehicles also communicate their speed and position to the the controller. The assumed 100% adoption rate of communicating vehicles is quite unrealistic, so it is unclear how the results will translate for more realistic adoption rates of communicating vehicles. ERIS does improve over SCHIC in terms of extending the search space for new potential schedules, since it allows a larger set of green phases than SCHIC. To clarify, SCHIC restricts each lane to be present in just one single green phase, whereas with ERIS each lane is now allowed to be present in multiple green phases. Since ERIS and SCHIC are both based on job-scheduling, the communicating vehicle scenario indicates a need for long look-ahead horizons for job-scheduling algorithms.

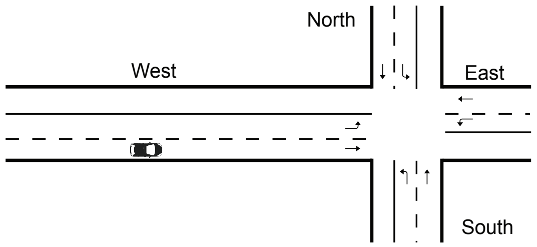
Hu and Smith [25] demonstrated that queue-based control outperforms SCHIC during saturated traffic demands. However, it can be concluded that SCHIC does perform well during low to unsaturated traffic demands given long look-ahead times for vehicle detection.

Although these three papers critically evaluate SCHIC, they all rely on quite large look-ahead times for vehicle detection which is only possible due to an important simplification. This simplification implies that all approaching vehicles are either, restricted to approach the intersection on a single entry-lane for just straight-ahead traffic [14] [25], or each vehicle communicates his upcoming lane change well in advance to the controller [7]. Doing so, inter-phase uncertainty is omitted, i.e. the uncertainty that a vehicle can actually request a different green phase by changing lanes is omitted.

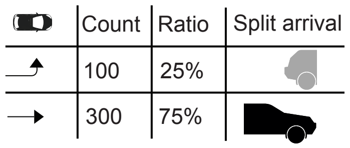
2.3 Existing baseline methods for inter-phase uncertainty

Instead of assuming that vehicles cannot change lanes whilst approaching the intersection [14], [25], three methods have been proposed for SCHIC which increase the look-ahead time for approaching vehicles while each detected vehicle is still able to change lanes. These three baseline methods are: *split-vehicles [1], [7], fixed-label [8] and Vehicle to Intersection Communication (V2I) [7].* These three baseline methods can be divided based on their working principle, into two categories:

- **Category A, containing both split-vehicles and fixed-label.** Both methods make use of statistical properties found in historic data on turning-movement proportions.
- **Category B, contains just Vehicle to Intersection Communication (V2I). V2I is also referred to as Oracle in this thesis.** In this category, a vehicle directly communicates his desired lane and hence green phase to the controller, thereby removing the uncertainty for this vehicle.

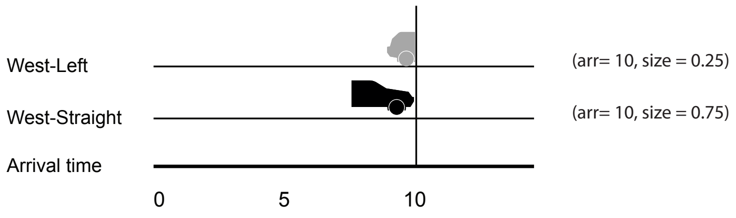


(a) The vehicle approaching from the West can still change lanes to request different competing green-phases.



	Count	Ratio	Split arrival
↱	100	25%	
→	300	75%	

(b) Traffic counts from the West origin.



(c) Arrival time at the intersection for the vehicle which is split into two.

Figure 2.1: Splitting vehicles into multiple phases based on historic traffic counts, images reproduced from Goldstein et al. [7].

Category A: split-vehicles based on historical turning movement proportions

A historical method to handle inter-phase uncertainty is to apply turning movement proportions to each vehicle count, i.e. upon first detection each vehicle is split based on the historically observed distribution of traffic per green-phase. The method is briefly mentioned in the first publication on SCHIC by Xie et al. [1] in 2012 and further explained by Goldstein et al. [7] in 2018.

A visual example is given in Figure 2.1a, on the West side of the intersection one vehicle has just entered the lane for straight-ahead traffic. At this point it is unclear if this vehicle will change lanes to end up in the left-turning lane, each lane-change will change which green-phase is requested from the controller. Therefore its vehicle count of 1 is split based on the historically observed turning movement proportions, resulting in two vehicle counts of size $\{0.25, 0.75\}$, as shown in Figure 2.1b. Figure 2.1c shows the the local phase based traffic observation, which uses a single arrival -and departure-time for both fractions of the vehicle count.

When individual vehicles are expected to arrive at the intersection within a threshold time, a single cluster is formed which contains these vehicles. When clustering split vehicles, the vehicle count of a cluster is based on the weighted fractions. A cluster of vehicles can for example contain a vehicle count of 1.5. The vehicle count is used to compute the weighted cumulative delay by SCHICs scheduler. More details on SCHICs scheduling algorithm will be supplied in section 3.1.3.

Category A: fixed-label corresponding to the dominant turning movement
Smith [8] proposes to use a fixed label for all vehicles which can still change lanes and hence change their desired turning movement at the intersection. This fixed label corresponds to the historically observed major turning movement. This method is proposed for the historic turning movement proportions as obtained during a field-test of SCHIC in 2015. Smith distinguishes between major -and minor traffic flows in his road network, i.e. major flows contain much more traffic per day than minor flows. In this road-network Smith identifies straight-ahead traffic as a major flow and all other turning-movements as a minor flow.

More specifically, Smith [8] introduces a *hybrid cycle* which is a combination of SCHICs scheduling algorithm for just two green-phases which only service the major flows. However in order to allocate green-time for the minor flows, this two phase schedule of SCHIC is overruled by another control algorithm – namely Vehicle Actuated Control (VA) – which services the minor flows. Additionally, the green-phases which service minor flows are allowed to be skipped when no vehicles are available for this phase. Performance metrics on the *hybrid cycle* are limited. Since it is only reported how often each minor phase is executed and how often a queue of vehicles of a minor traffic flow is unable to fully clear during its green time.

In this thesis, the *hybrid cycle* is not completely identical to Smiths [8] proposal since green-phases which service minor flows are scheduled by SCHIC and therefore not allowed to be skipped. Xie et al. motivate that green phases are not skipped by SCHIC based on an argument of safety and fairness [1]. Smiths proposed method is implemented as follows in this thesis. The fixed label classification which corresponds to the historically observed major turning movement is identical to Smith [8]. Given the fixed label prediction, SCHIC creates a schedule in which all vehicles will join the major traffic flow for straight-ahead traffic. If a miss-prediction occurs, so when a vehicle is actually detected on a left-turning lane, its predicted lane is corrected. After detecting a miss-prediction, the left-turning vehicles are now also scheduled by SCHIC. The comparison is therefore limited to the baseline method of a fixed label for each new vehicle.

Category B: Vehicle to Intersection Communication (V2I) or Oracle
Inter-phase uncertainty can be eliminated when a vehicle is able to directly communicate his desired turning movement to the traffic light controller. Vehicles equipped with this technology can broadcast precise information about their position, velocity and desired turning movement to the intersection. These messages are known as Cooperative Awareness Messages (CAM). The drivers lane choice is known in advance rather than predicted, the performance gain can thus be seen as an upper limit of any predictive approach. On an isolated real-world intersection controlled by a commercial controller named ImFlow, the reduction in travel time can be up to 9% as demonstrated by Czechowski et al. [26].

These three baseline methods are well documented, however, the actual traffic flow performance of SCHIC when either of these methods is applied is ill-reported. So, it is unclear how much each of the three methods actually increase SCHICs traffic flow performance.

2.4 Conclusion

In this chapter, general literature on (commercially applied) traffic light controllers has been discussed. In succession, the focus shifted towards strong -and weak points of SCHICs traffic control algorithm. In this thesis we focus on two real-world constraints, since they have been overlooked in previous work on SCHIC. These two constraints are (1) a realistically short length of an entry-lane and (2) the uncertainty than an approaching vehicle can change to a different entry-lane which is serviced in a different and competing green phase.

Literature on SCHIC has introduced several baseline methods to handle inter-phase uncertainty, i.e. {fixed-label, split-vehicles and Vehicle to Intersection Communication (V2I)}. Yet, the traffic flow improvement of SCHIC when either of these methods is applied is often not reported or ill-reported. Therefore, the traffic flow improvement of SCHIC with each of its baseline methods is quantitatively evaluated in this thesis. Furthermore, a new method to increase SCHICs traffic flow performance under inter-phase uncertainty is presented, namely Lane Change Prediction (LCP). LCP can potentially reach higher accuracies than the two existing statistical baseline methods, i.e. {fixed-label, split-vehicles}, so it can potentially outperform these two former baseline methods.

Chapter 3

Theoretical framework of SCHIC

In this chapter SCHIC is explained in detail, in order to clarify in which manner the proposed extension of Lane Change Prediction (LCP) connects to SCHICs local traffic observation. SCHICs main components are briefly explained by stating both the abilities and inabilities of this traffic light controller. Additionally, this non-biased and detailed overview allows for a clear comparison to other traffic light controllers.

Section 3.1 explains SCHICs local planning method by giving a high-level overview of the supported features. In that section, SCHIC is decomposed into its three main components: **(1)** the traffic input information, **(2)** the scheduling algorithm and **(3)** the coordination mechanism among intersections. These main components are further explained by means of a flowchart which shows the main logic of SCHIC in Section 3.2. Afterwards, Section 3.3 compares the originally published results of SCHIC with those as obtained with the re-implementation of SCHIC which is realized for this study. The most important information of the entire chapter is summarized in Section 3.4.

Relevant terminology
Most terms which are used in this chapter are also explained and listed in the glossary on page 96.

3.1 Detailed overview of supported features

In this section a high-level overview of SCHICs local scheduling method is given. Both the capabilities and inabilities of SCHIC will be explained which allows for a comparison to other traffic light controllers.

3.1.1 Restrictions imposed on the phase set

This section is illustrated by Figure 3.1.
Xie et al. [1] define the intersection as a set of entry -and exit roads which intersect each other in a single point. The entry-roads lead traffic towards the intersection whereas exit-roads move traffic away from the intersection. Each entry-road can contain multiple entry-lanes, where each entry-lane is dedicated to a single traffic movement. Examples of traffic movements are straight-ahead -and left-turning traffic. A movement is formally defined as a path from one entry-lane to one exit-lane. Each movement is shown as an arrow in Figure 3.1.

Phases and movements
A phase is an assignment of light signals – i.e. red, yellow or green – to each and every movement on the intersection. Movements which can safely pass the intersection at the same time are grouped into a single green phase. In other words, a green phase services a set of non-conflicting movements. When one green phase is active it shows a green signal (i.e. a green light) to its assigned movements, while all other movements observe a red signal. The total set of green phases is referred to as I. In some images, a green phase is referred to with two integers: (i, g) or $g^{(i)}$, where $i \in I$ is the phase index and g the variable phase duration in seconds.

To conclude, a simplifying restriction is imposed by Xie et al. [1], i.e. for SCHIC each movement is only allowed to be present in one green phase. To clarify, there is no overlap of movements between all green phases. Including more green phases and allowing movements to be present in multiple green phases will increase the flexibility and performance of a single intersection. However it also increases the search space for the scheduler.

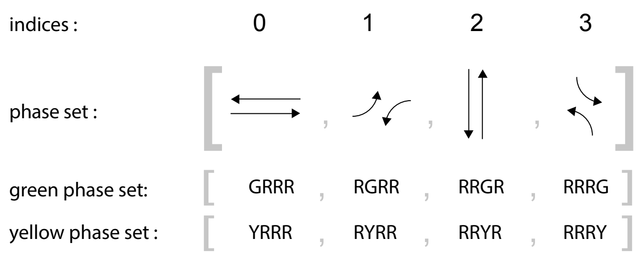


Figure 3.1: Example of a phase set which does not contain overlapping movements. Including the individual green -and yellow light signals in "green -and yellow phase set". This is an often encountered phase set in The Netherlands, and also the default phase set of Simulation of Urban Mobility [19] (SUMO).

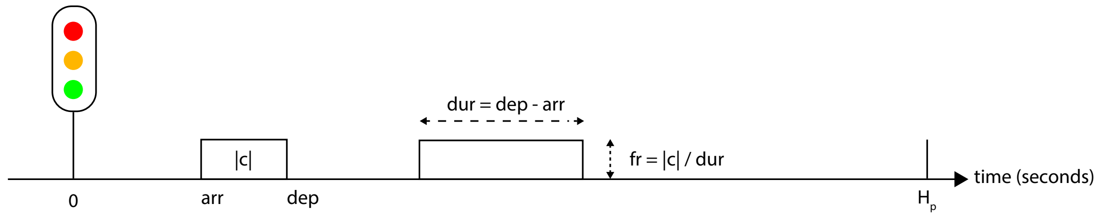


Figure 3.2: Visual example of clusters on an entry-lane. Each cluster represents at least one car and is defined with variables: arr, dep, $|c|$. Image reproduced from Xie et al. [1].

3.1.2 Local traffic observation

Vehicle detection relies on induction loop sensors in SCHIC [1], these induction loop sensors are commonly used by real-world intersections. For SCHIC these induction loops are placed at the start -and end of each entry-lane, respectively called stop -and advance detectors. Each loop can detect vehicles which pass or stay over the loop sensor. The stop -and advance detectors are respectively placed at points 0 and H_p in Figure 3.2.

Working principle of induction loop sensors
Induction loop sensors are placed inside the pavement of e.g. an entry-lane. Each loop is charged with an alternating electric current at a certain frequency. When a vehicle passes or remains above the loop, the ferrous material of this vehicle will change the inductance of the induction loop sensor. This change of inductance allows the sensor to detect vehicles [1].

Traffic observation on an entry-lane
Upon first detection of a new vehicle at point H_p in Figure 3.2, the arrival time of this new vehicle at the intersection is computed by assuming it will continue with a fixed speed. This travel-time of a newly detected vehicle towards the intersection will now be referred to as: prediction horizon (H_p).

H_p is computed as follows: H_p = distance between advance detector and the stop-line of the intersection / speed limit [1]. Upon first detection of each new vehicle a timestamp is saved which is referred to as the current decision time (cdt). This *cdt* is used to compute the expected arrival time of each new vehicle. The expected arrival time (arr) is computed as arr = cdt + H_p.

[1]`https://en.wikipedia.org/wiki/Induction_loop`

Clusters (in a Road Flow List (RF))
A cluster (c) is a data structure in use by SCHIC, which represents the detected vehicles. For each newly detected vehicle a new cluster is made. The data fields of a cluster are: $|c|$ – the number of vehicles in this cluster, $arr(c)$ – the expected arrival time of the first vehicle, $dep(c)$ – the expected departure time of the last vehicle. For a new cluster, the number of vehicles ($|c|$) is set to 1. The arrival time is computed as explained in the previous paragraph. The departure time $dep(c) = arr(c) + samp$, where $samp$ is the sampling interval in seconds of the vehicle detection method which is set to $samp = 0.5$ *sec.*

All clusters (which represent arriving vehicles) approach the intersection on a single entry-lane and each entry-lane is associated to only a single movement m. Recall that movements are shown as individual arrows in Figure 3.1. All clusters are first saved in a list corresponding to their desired movement m. Specifically, for each individual movement m at the intersection; a separate *Road Flow List* is maintained: $C_{RF,m} = (c^1, .., c^{|C|})$. Here, Road Flow List (RF) stands for *Road Flow List*, where m is the index of the desired movement, and $|C_{RF,m}|$ is the total number of clusters for movement m.

Phase based traffic observation (Inflows (IF))
A green phase can service multiple entry-lanes (each for a single movement m) at the same time. E.g. two opposing entry-lanes for straight-ahead movements can be serviced by the same green phase (as seen in Figure 3.1). All individual RFs with an individual movement m ($C_{RF,m}$) which are serviced by the same green phase are simply merged into a separate *Inflow List*: $C_{IF,i}$. So, for each individual green phase with index i, a separate list is maintained: $C_{IF,i}$. Here, the Inflows (IF) is used to describe all approaching traffic, i.e. IF is a set containing all individual *Inflow Lists*: $C_{IF,i}$. So, the following set is collected $IF = \{C_{IF,i} : \forall i \in |I|\}$. Where i is an individual green phase index and $|I|$ is the total number of green phases in the phase set I. IF is used by the scheduler in order to determine a schedule with minimal cumulative delay.

Threshold based clustering
Clusters which arrive within a certain threshold time, $th_c = 3.0$ *seconds*, are merged into a single cluster. Threshold based clustering is not performed for each individual entry-lane which is associated to a single movement m. Instead, it is only performed for the movements which are already merged in $C_{IF,i}$ based on their green phase with index i. The merging of multiple clusters into one reduces the problem size of the scheduler. To clarify, a merge of two clusters, c_1 and c_2, into one cluster c_0 is performed as follows. This new cluster has $arr(c_0) = min(arr(c_1), arr(c_2))$, $dep(c_0) = max(dep(c_1), dep(c_2))$, $|c_0| = |c_1| + |c_2|$.

Parameter	Full name	Setting
th_c	threshold clustering	3 seconds
$sult$	start-up lost time	3.5 seconds
shw	saturation headway	2.5 seconds

Table 3.1: Fixed parameter settings of SCHIC

Queue modelling
While the previously explained departure times work for SCHICs scheduling algorithm, they only work when these vehicles will not become a queue at the intersection. If these vehicles will need to stop at the intersection (since they observe a red signal) and become a queue, their departure times are updated since queued vehicles are assumed to have:

1. A fixed saturation headway ($shw = 2.5s$) towards the next queued vehicle (see Table 3.1.
2. A start-up lost time since they need to accelerate from a stand-still (see Table 3.1).

Queues on each entry-road are associated to a single movement m, these queues are modelled in a linear manner. The arrival time of a queued vehicle remains the same as before, however, its departure time is updated. The increase in departure time is based on the saturation headway ($shw = 2.5s$), which is the assumed average headway in seconds between vehicles in saturated flow. To clarify, when the queue has accelerated from a standstill then each vehicle is assumed to have an headway of 2.5 seconds to the vehicle in front of it.

An estimate of the queue size ($q^{(m)}$) is made for each entry-lane, where m is the index of the entry-lane which services movement m. $q^{(m)}$ is defined as the difference between the number of expected vehicles at the current decision time (cdt) and the actually departed vehicles from that entry-lane.

The scheduler considers a start-up lost time ($sult$) of 3.5 seconds to account for the acceleration time of a queue cluster from a standstill. So, the $sult$ is not added by the clustering method. But added by the scheduler since adding the $sult$ depends on the chosen schedule. To clarify, when two queues from opposing entry-lanes are serviced during the same green phase the start-up lost time of 3.5 seconds should only be added once.

Pseudocode of queue modelling
The pseudocode of the queue clustering is shown in Algorithm 1 together with the following explanation.

Line 7: the queue count $q^{(m)}$ is updated here. $q^{(m)}$ is a count of the vehicles which are still at the intersection while they were assumed to have left already based on their expected arrival time.

Line 12: a queue is expected if the queue count is larger than 0, i.e. $q^{(m)} > 0$. The queued vehicles (i.e. the first $q^{(m)}$ vehicles in the Road Flow List $C_{RF,m}$) are now saved in a single queue cluster c_q. The departure time of the vehicles in c_q is updated by using the Saturation Flow Rate (sfr). Sfr is defined as $sfr = N_{lanes}/shw$. Where N_{lanes} is the number of entry-lanes which are available for movement m and shw is the average saturation headway. $N_{lanes} =$ 1 in this thesis.

Algorithm 1: Arriving vehicles on the entry-lane for movement m which have become a queue; require more time to be serviced. So, they are now saved in the Road Flow List ($C_{RF,m}$) as a single queue cluster c_q with an increased departure time.

```
1  create_queue_cluster_on_roadflow($cdt, m$)
   Input  : cdt is current decision time, m is the entry-lane index associated to movement
            m.
   Output : The queue count estimate ($q^{(m)}$) is updated. Which is used to update the
            queued clusters with a fixed flow rate in $C_{RF,m}$.
2
3  $q^{(m)} = 0$
4  old_c = first_cluster($C_{RF,m}$)   // Save the earliest to arrive cluster of $C_{RF,m}$, to reference
     it later.
5
   // For each cluster c in $C_{RF,m}$, check if its arrival time (arr(c)) is earlier than the
      current decision time (cdt).
6  for $\forall c \in C_{RF,m}$ do
7      if $arr(c) < cdt$ then
           // The arrival time of cluster c (arr(c)) is below the current decision time (cdt), so
              the vehicles of cluster c are labelled as a queue.
8          $q^{(m)}$ += count(c)                    // Update the queue count estimate ($q^{(m)}$).
9          delete c from $C_{RF,m}$              // Cluster c is later converted to a queue cluster.
10     end
11 end
12 if $q^{(m)} > 0$ then
       // The first $q^{(m)}$ cars will become a single queue cluster $c_q$ in $C_{RF,m}$.
       // Create a single queue cluster $c_q$ and update its variables accordingly, e.g. update
          its departure time.
13     count($c_q$) = $q^{(m)}$
14     arr($c_q$) = arr(old_c)
15     fr($c_q$) = sfr
16     dep($c_q$) = arr($c_q$) + (count($c_q$) / fr($c_q$))
17 end
18
19 return insert $c_q$ into $C_{RF,m}$      // All queued vehicles for movement m are now saved in one
     queue cluster $c_q$.
```

Anticipated queues
For each movement, the *anticipated* queue contains the number of vehicles that are either presently or in the future expected to join into the existing queue – before this existing queue clears at the intersection. The anticipative queue is modelled by extending the existing queue, c_q, with new anticipated vehicles such that the arrival time and flow-rate of c_q remain unchanged. To conclude, the number of vehicles in c_q, is increased and only its departure -and duration-time is updated by scaling it linearly with the increased vehicle count, $|c_q|$.

To fully explain the algorithm, the clusters in $C_{RF,m}$ are iterated over one by one. Consider the non-queue cluster, c^j, for which conditions 0, 1 and 2 are performed.

Condition 0: if $arr(c^j) \leq dep(c_q)$, then condition 1 is also evaluated.

Condition 1: If $dep(c^j) \leq dep(c_q)$ or $fr(c^j) \geq fr(c_q)$, a full join of c^j into c_q is performed: $|c_q| = |c_q| + |c^j|$, and c^j is removed from $C_{RF,m}$.

Condition 2: if conditions 0 and 1 do not hold, a part or even all of c^j might still join into c_q. This Condition 2 is further shown in the pseudo-code given in the Appendix in Algorithm 4.

3.1.3 Scheduler

As explained before (Section 3.1.2 on page 26), the scheduler uses the Inflows (IF) which contains all approaching vehicles as ordered for each individual green phase. A single green phase can be referred to by its index: i.

The Inflows (IF) are used to create and evaluate different Phase Switching Sequences (PSSs). Each PSS is a sequence of green phases with appropriate durations to service the set of currently observed vehicles found in Inflows (IF). A complete PSS services all currently observed vehicles, so quite often different complete phase switching sequences are possible. For each PSS the cumulative delay of all serviced vehicles is computed based on the entire process of approaching and passing through the intersection. Lastly, a complete PSS with minimal cumulative delay is selected from the available set of complete PSSs. Figure 3.3 shows an example with traffic for three phases within $C_{IF,i}$, which results in two different complete phase switching sequences (PSS), i.e. Schedule $S_{(A)}$ and $S_{(B)}$.

Theorem 1. *(Elimination Criterion)* of Xie et al. [1]
As shown in Figure 3.3, schedule $S_{(A)}$ and $S_{(B)}$ service the exact same clusters but schedule $S_{(A)}$ does so with a lower cumulative delay. So schedule $S_{(A)}$ is selected. Schedule $S_{(B)}$ can already be eliminated at stage $k = 3$ (i.e. when the first $k = 3$ clusters of schedule $S_{(A)}$ and $S_{(B)}$ are known). At stage $k = 3$ the following theorem can be applied: Theorem 1. *(Elimination Criterion)* of Xie et al. [1], as further explained in the caption of Figure 3.3.

Safety and fairness
The scheduler applies the following safety and fairness rules since the traffic light interacts with human drivers.

1. Each green phase has a variable duration between a minimum and a maximum value: $[G^{(i)}_{min},\ G^{(i)}_{max}] = [5\ seconds,\ 55\ seconds]$.

2. The yellow phase will run for a fixed inter-green time between two subsequent green phases: $Y^{(i)} = 5\ seconds$. This yellow phase takes the reaction time of human drivers into account, as during this yellow phase vehicles are requested to stop at the intersection if it is still safe to do so.

3. Each green phase will run for a fixed minimum duration, so skipping a green phase is not allowed. Instead green phases switch in a fixed cycle, i.e. the next green phase index is determined by: next(i) = (i+1) modulo $|I|$. Where i is the current green phase index and $|I|$ is the total number of green phases in the phase set I.

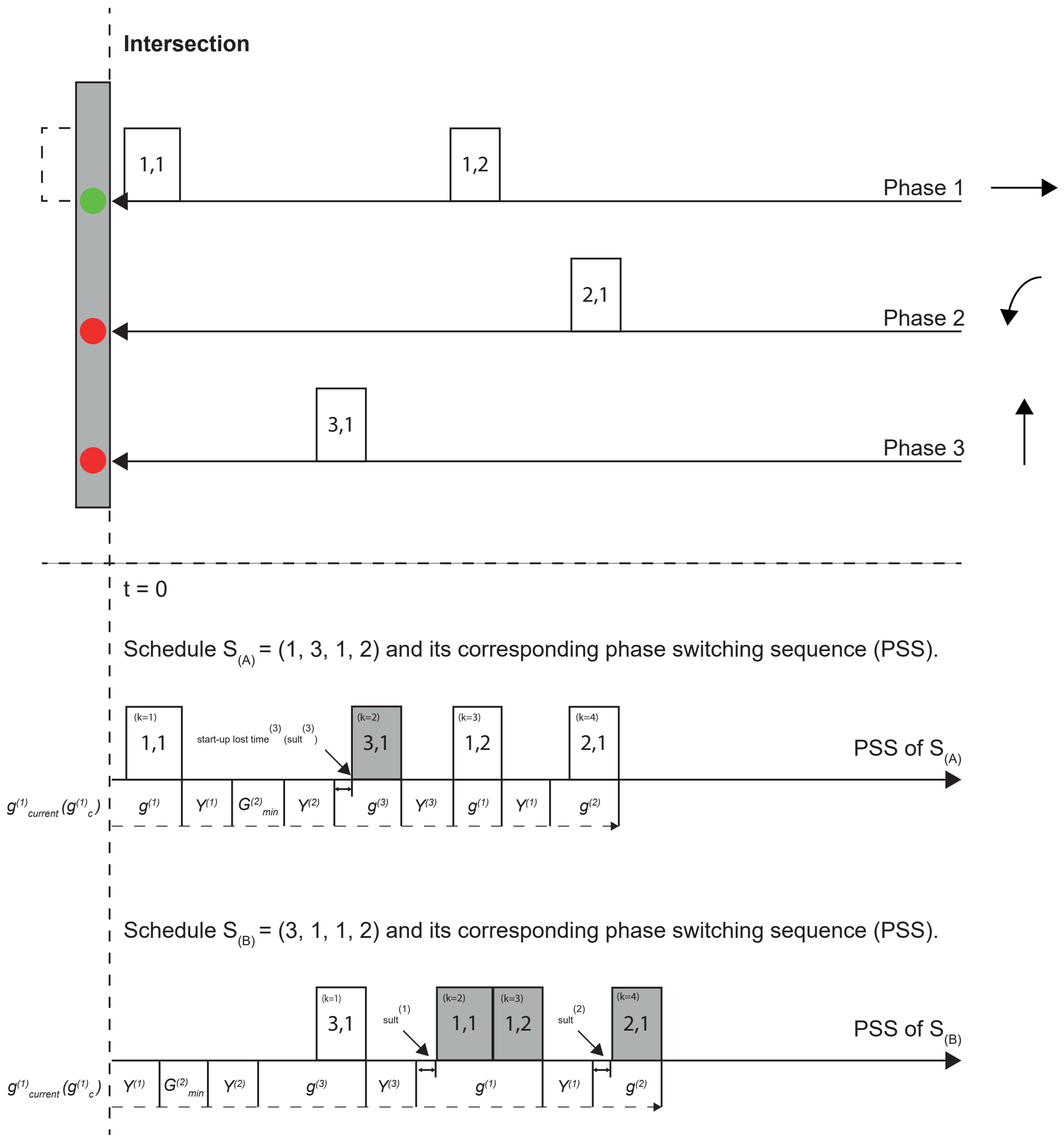

Intersection
1,1
1,2
Phase 1
2,1
Phase 2
3,1
Phase 3
t = 0
Schedule S(A) = (1, 3, 1, 2) and its corresponding phase switching sequence (PSS).
(k=1) 1,1
start-up lost time (3) (sult (3))
(k=2) 3,1
(k=3) 1,2
(k=4) 2,1
PSS of S(A)
g(1)current (g(1)c)
g(1) Y(1) G(2)min Y(2) g(3) Y(3) g(1) Y(1) g(2)
Schedule S(B) = (3, 1, 1, 2) and its corresponding phase switching sequence (PSS).
(k=1) 3,1
sult (1)
(k=2) 1,1
(k=3) 1,2
sult (2)
(k=4) 2,1
PSS of S(B)
g(1)current (g(1)c)
Y(1) G(2)min Y(2) g(3) Y(3) g(1) Y(1) g(2)

Figure 3.3: *Top half of image:* observed vehicles per individual phase, known as clusters (e.g. for the movement straight-ahead or left-turn). *Lower half of image:* different possible (complete) schedules $S_{(A)}$ and $S_{(B)}$ which (completely) service all observed clusters. Grey rectangles indicate queued vehicles. Among all different schedules, the one with minimal cumulative delay is selected. At stage $k = 3$, both $S_{(A)}$ and $S_{(B)}$ have serviced the same clusters *and* the same phase is active. However, the cumulative delay of $S_{(B)}$ is higher than that of $S_{(A)}$, so schedule $S_{(B)}$ can already be eliminated at stage $k = 3$ – see **Theorem 1.** *(Elimination Criterion)* of [1]. Image reproduced from: Xie et al. [1].

3.1.4 Coordination mechanism between intersections

Although the coordination mechanism of SCHIC is not implemented in this thesis, it is mentioned here as a reference. Xie et al. [27] extend SCHIC by adding a scalable, decentralized coordination mechanism which is limited to interaction among directly neighbouring intersections. As stated before, each SCHIC agent develops a Phase Switching Sequence (PSS) which optimizes for its own locally observable traffic movements through the intersection. These locally observable traffic movements are within the local look-ahead horizon of each controller.

The basic coordination protocol allows each agent to extend his locally observable traffic movements by sending requests to each of its direct upstream neighbours. I.e. after the requests it simply receives a projection of newly planned traffic outflows from its upstream neighbours. Depending on the horizon length of each request, the reply may contain influences of non-direct neighbours as well. Lastly, Xie et al. [27] mention that this coordination mechanism can suffer from two problems: (1) "nervousness" and (2) dynamic instability in the network. Therefore, the coordination mechanism is applied in such a manner that these two problems are reduced thereby increasing SCHICs performance.

3.2 Flowchart of SCHICs algorithms

Figure 3.4 shows a flowchart of SCHICs algorithms where the existing Phase Switching Sequence (PSS) is updated in six steps. Each of these steps are further explained in the paragraphs directly below. A PSS is a sequence of phases and it is built on the initial phase condition at its start time. This initial phase condition is (i_c, g_c) – where i_c is the current green phase index and g_c is the time that the current phase with index i_c has been green [1].

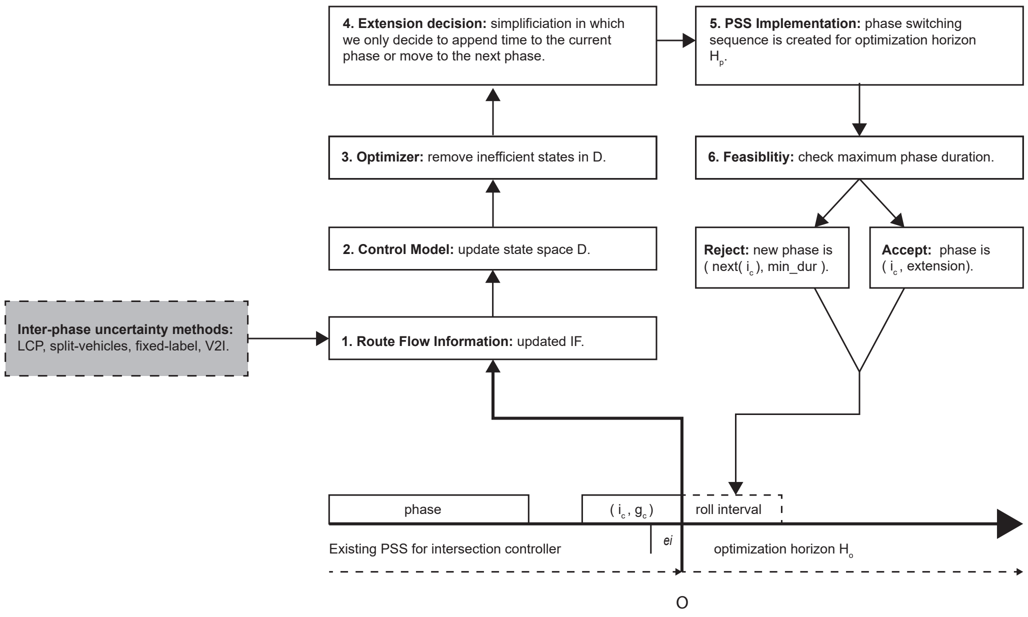

4. Extension decision: simplificiation in which we only decide to append time to the current phase or move to the next phase.
5. PSS Implementation: phase switching sequence is created for optimization horizon Hp.
3. Optimizer: remove inefficient states in D.
6. Feasiblitiy: check maximum phase duration.
2. Control Model: update state space D.
Reject: new phase is (next(ic), min_dur).
Accept: phase is (ic , extension).
Inter-phase uncertainty methods: LCP, split-vehicles, fixed-label, V2I.
1. Route Flow Information: updated IF.
phase
(ic , gc)
roll interval
ei
Existing PSS for intersection controller
optimization horizon Ho
O

Figure 3.4: Flowchart of SCHICs components, in a high level abstraction. The methods for inter-phase uncertainty connect to step: *1. Route Flow Information.* The image is re-created without any intended changes from SCHICs original publication by Xie et al. [1].

1. Route Flow Information (i.e. local traffic observation)
At each time-step of 0.5 *seconds*, the scheduler obtains the updated current traffic observation, namely the Inflows (IF). These Inflows (IF) are defined as a separate list per green phase and each list contains the arrival times of all approaching vehicles for each single green phase. However, vehicles which are expected to arrive briefly after each other are aggregated into a single arriving cluster as explained in section 3.1.2.

2. Control Model
The scheduler uses a forward recursive algorithm to re-order the observed clusters found in the Inflows (IF) in all allowed manners. A simple example is shown in Figure 3.5, which shows a tree structure where each path is a different re-ordering of the observed clusters. This tree structure is referred to as the statespace D and each state is shown as a node of the tree. Each node is defined by three variables: $\{X, s, s_{prev}\}$.

X is an array which counts the number of serviced clusters per phase, i.e. $X = [1, 1]$ means that a single cluster is serviced for each of the two phases. The other two parameters are as follows: s is the current phase index of this state and s_{prev} is the phase index of the previous state.

The example in Figure 3.5 considers two green phases and a single cluster for each of the two green phases. In this particular case two paths can lead to the full state $X = [1, 1]$, the path with minimal cumulative delay is used to create a Phase Switching Sequence (PSS). However, during the creation of statespace D, the optimizer limits the search space as explained in the next step directly below.

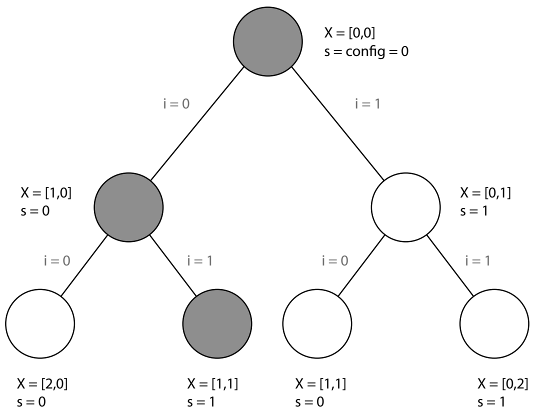


Figure 3.5: Visualisation of the statespace D. Which is built by a forward recursive algorithm to re-order the observed clusters for all phases. In this example there is one cluster for each phase. Each state (shown as a node of the tree) is defined by three variables: $\{X, s, s_{prev}\}$.

3. Optimizer – (Theorem 1. *(Elimination Criterion)* of Xie et al. [1])
An elimination criterion is in place in order to reduce the search space without losing optimality. The state-space D can contain multiple paths which lead to different states, which have however serviced the exact same clusters of vehicles.

Let's call these two states: $\{A, B\}$. The paths towards $\{A, B\}$ will each lead to a different Phase Switching Sequence (PSS). If both states $\{A, B\}$ have the same current phase index s, and an identical X array, then these states have previously and currently serviced the exact same clusters. If one of these states is dominated by the other, i.e. the cumulative delay $d_A \leq d_B$ and the end-time $t_A \leq t_B$, then one can eliminate state B without loss of optimality [1].

This optimization step is visualized by schedules $S_{(A)}$ and $S_{(B)}$ in Figure 3.3.

4. Extension decision
As mentioned by Xie et al. [1], SCHIC finishes each cluster which is currently being serviced. Therefore, the extension decision computes the amount of time (i.e. extension time) in order to finish the currently serviced cluster. This extension time is based on the expected departure time of the currently serviced cluster. Pseudo-code is given in the Appendix (section 10).

5. PSS Implementation
The state space D is fully built if it contains at least one full state, i.e. a full-state has serviced all observed clusters in IF. SCHIC retrieves the optimal schedule S by selecting the the full state with minimal cumulative delay d. From this full state SCHIC tracks back to the root node, and the phase durations, pd, are saved which have been stored in each state. The schedule S and corresponding phase durations $PD = (pd_1, ..., pd_{|S|})$ are saved until they are replaced in the next scheduling iteration.

Thereby, the schedule S which services all observed clusters with minimal delay is converted into a Phase Switching Sequence (PSS). This PSS contains the green phase durations and inter-green times (i.e. yellow light times).

More details are supplied in the Appendix (section 19, section 22).

6. Feasibilty check
During the creation of state-space D there is no check performed on the maximum duration of a green phase (i.e. 55 *sec*). This check is only performed on the final schedule. If a violation of the maximum green time occurs, the scheduler simply selects the next allowed green phase. This is possible since SCHIC cycles through green phases in a fixed order, where each phase has a minimum green time. Pseudo-code is given in the Appendix (section 10).

3.3 Quantitative comparison between SCHICs re-implementation and SCHICs original published results

In this research SCHIC is re-implemented based on the pseudo-code supplied by Xie et al. [1]. Appendix C concludes with a quantitative comparison between the published results on SCHIC by Xie et al. [1] and the results as obtained with the re-implementation of SCHIC. In order to make these comparisons, all intersections and vehicle demands are recreated based on specifications found in [1]. Unfortunately, some specifications and parameter settings are omitted in the publication of Xie et al. [1]. Therefore, the used parameter settings in this research are clearly reported in Appendix C. While each parameter described in Appendix C might change the results of the experiments, one can reasonably conclude that the results of the re-implementation are close to the original published results.

3.4 Summary

To summarize, a vital part of SCHIC is its clustering method and queue modelling which both aggregate vehicles together and thereby reduce the input for the job-scheduler. Secondly, its elimination criterion also reduces the computational load by removing inefficient schedules at an early stage. Additionally, it has been demonstrated that a scalable coordination mechanism can be layered on top of the a-synchronously operating single intersections. However, the time duration of each green phase varies between a minimum and maximum duration. Since green phases are not skipped this is expected to lead to sub-optimal behaviour when unbalanced traffic scenarios are applied.

Chapter 4

Proposed method: Lane Change Prediction

While SCHIC has been demonstrated to be successful, this was most often realised with large look-ahead times which required an important simplification. This simplification is as follows, each approaching vehicle is not allowed to change lanes when approaching the intersection and is therefore only able to request a single green phase from the traffic light controller. For example, Xie et al. [1] and Dang et al. [14] restrict each vehicle to approach the intersection on just a single entry-lane and this entry-lane is serviced in a single green phase for straight-ahead traffic. This simplification removes the uncertainty that vehicles can request different green phases, for example by entering the entry lane for left turning traffic which is often serviced in a different and competing green phase. By applying this restriction, vehicles can now be detected long in advance, since their final entry-lane and hence desired green phase is known a priori. This enables unrealistically long look-ahead times which improve SCHICs performance [1].

In this chapter we take a different approach, rather than assuming that lane changes are not present, we investigate how far we might be able to use machine learning methods to predict these lane changes, thereby reducing the uncertainty for SCHIC. Specifically, we intend to predict a lane change of a vehicle based on its observed driving style towards multiple short entry-lanes of an intersection. Chapter 2 already explained three baseline methods which are aimed at a similar scenario, since they aim to increase the look-ahead time for vehicle detection while each detected vehicle is still able to perform a lane change. However, these three baseline methods do not use the anticipative driving style of each approaching vehicle as is intended with the proposed method of Lane Change Prediction (LCP).

Section 4.1 discusses related work on Lane Change Prediction (LCP), and is used to motivate our approach to LCP. Section 4.2 partly covers the realisation of our ideas by presenting our *feature vector*. This feature vector captures the observed driving style of an approaching vehicle and its upcoming traffic state. It is used by a classification model to predict the future lane change of a vehicle towards one of the shorter entry-lanes. Section 4.3 motivates based on some characteristics of SCHICs logic, how the method of Lane Change Prediction (LCP) can be applied best to SCHIC in order to reduce the uncertainty of lane changes. This chapter is concluded in Section 4.4.

4.1 Related work on Lane Change Prediction (LCP)

Although Chapter 2 contains related work, it does not contain work on Lane Change Prediction (LCP). So, this section contains related work *only* on LCP and is thereafter used to motivate the design of our classification model. I.e. our approach uses a comparable *feature vector* and classification model as those described below.

Hou et al. [28] proposed a lane changing assistance system to advise drivers in 2014. The system detects if a safe gap is found to execute a mandatory lane change on lane drops at Highway and Inter-state roads, see Figure 4.1a.

A decision-tree and Bayes classifier are used to predict if the lane change can be performed safely. Open-Source real-world vehicle trajectory data is used to evaluate the performance of the model on Highway and Inter-state lane drops. Important input data for the model are the vehicle's speed and distance relative to the vehicles on the target lane. The best performance is found when both a Bayes -and Decision Tree classifier are combined as a single classifier which uses a majority voting principle. This results in a prediction accuracy for a merge event of 79.3% [28].

Studies on lane change predictions are mainly focussed on highway traffic. However, in 2011 Ortiz et al. [18] introduce a predictive approach for an inner-city traffic light which closely resembles the intersection which is evaluated in this thesis, see Figure 4.1b.

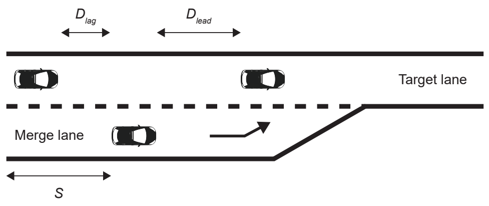
The input of the predictive model is limited to the speed of the vehicle and the distance and status of its approaching traffic light. Future behaviour of each vehicle is predicted with a multi-layer perceptron model (MLP). This application of Ortiz et al. [18] is aimed at Advanced Driving Assistant Systems (ADAS). The system can warn a driver when he/she does not brake in time. The predictive output is one of three behaviour primitives: "keep speed", "braking" or "stopped".

While the system does not predict a lane-*change* it is expected to fit well into our scenario, when one of the two final entry-lanes is serviced with a green signal and the other entry-lane is receiving a red signal. To clarify, a high correlation can be expected between "keep speed" and merging into the entry-lane which is currently serviced with a green signal. A similar high correlation can be expected between "braking" or "stopped" when the driver wants to enter the entry-lane which is currently observing a red signal.

The predictive accuracy of the model is shown in a single ROC curve for several future time-frames, i.e. several increasingly large future time-frames are compared for which future behaviour is predicted, see Fig. 7 in [18]. In conclusion, although the model can perform behaviour prediction for a time-frame of up to 6 *sec* in the future, Ortiz et al. conclude that the model is highly accurate for a future time-frame of up to 3 *sec*.

Motivations for an interpretable predictive model
Given these learning models of different complexity, one could consider the trade-off between the accuracy and interpretability of the predictive model. An argument can be made for a relatively simple and explainable machine-learning model, since in this thesis all predictions are based on

simulated driving behaviour. A Decision Tree as proposed by Hou et al. [28] is evaluated further in this thesis, since the learned model is proven on real-world data and therefore likely to be accurate on our simulated data as well. Therefore, a more complex multi-layer perceptron Model (MLP) as proposed by Ortiz et al. [18] might not be needed for our simulated data. Lastly, for the Traffic Light Control industry it can be common practice to validate classification models in simulation by means of statistical tests before these models are applied to real-world road users. In conclusion, we focus on a Decision Tree since the results of this open-box model are in general easy to explain and interpret by means of statistical tests.



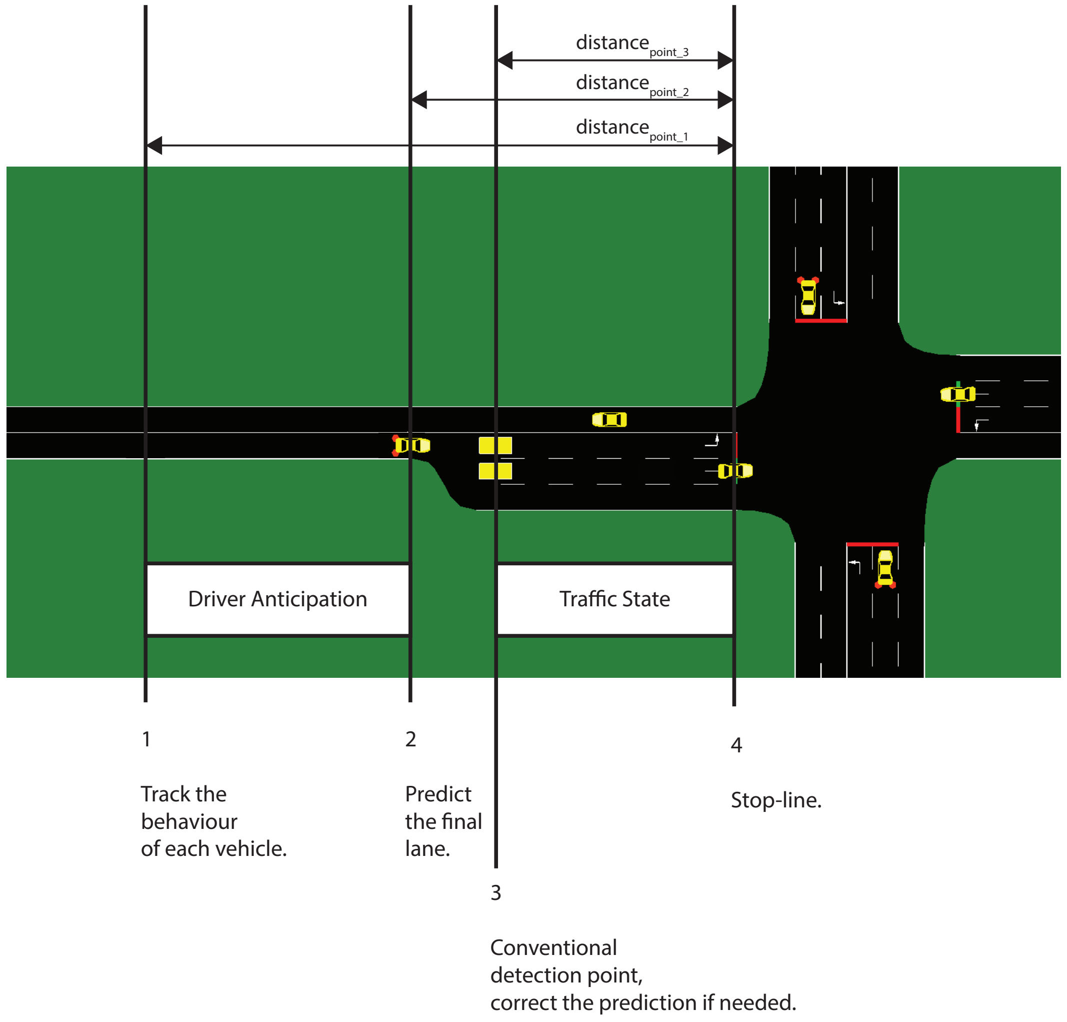
(a) Scenario of an Inter-State or Highway lane-drop, in use by Hou et al. [28]. Image based on Hou et al. [28]

(b) The approach scene towards the Waalsdorperweg traffic light in The Hague (own picture). Which closely resembles the traffic light approach scene as evaluated by Ortiz et al. [18]. See Fig. 3 of Ortiz et al. [18].

Figure 4.1: Scenario of Hou et al. [28] and a comparable scenario to that of Ortiz et al. [18]

4.2 Feature vector for Lane Change Prediction (LCP)

In this section, we begin to present our proposed approach to Lane Change Prediction (LCP). Specifically, this section introduces the feature vector which will be used by a classification model.



$\text{distance}_{\text{point\_3}}$
$\text{distance}_{\text{point\_2}}$
$\text{distance}_{\text{point\_1}}$
Driver Anticipation
Traffic State
1
Track the behaviour of each vehicle.
2
Predict the final lane.
3
Conventional detection point, correct the prediction if needed.
4
Stop-line.

Figure 4.2: Locations used by Lane Change Prediction (LCP).

In order to obtain the feature vector, two important regions are defined within Figure 4.2:

1. **Driver Anticipation Region**, located between points 1 and 2.
2. **Traffic State Region**, located between points 3 and 4.

Each lane change prediction is made for an *ego-vehicle*, this *ego-vehicle* is defined as a single vehicle which is first observed and tracked in the Driver Anticipation Region. Its future lane change is predicted when it reaches the end of the Driver Anticipation Region. The first set of features are based on the driving behaviour of the *ego-vehicle* when approaching the lane-split in the Driver Anticipation Region. The second set of features consider the observed traffic state on the two individual lanes within the Traffic State Region.

The feature vector in this study is based on related work of Hou et al. [28] and Ortiz et al. [18], as previously described in Section 4.1. Hou et al. [28] present a predictive model for a scenario which does not consider a traffic light, but a lane-drop which is used to merge into a single target lane of a Highway. They compose their features based on the ego-vehicle's speed and distance relative to the vehicles on the target lane. Ortiz et al. [18] introduce a predictive approach for an inner-city traffic light. They limit the input of their predictive model to the speed of the ego-vehicle and the distance and status of its approaching traffic light. Our proposed feature vector is loosely connected to that of Hou et al. [28], since we initially based it on the feature vector of Ortiz et al. [18].

Obtaining the feature vector
Below we consider an *ego-vehicle* which approaches the lane-split, and we describe how its feature vector is obtained.

1. Driver Anticipation features
When the ego-vehicle enters the *Driver Anticipation Region*, it is being tracked by radar equipment in order to observe its velocity. In total, two features are obtained in this region to describe the *Driver Anticipation (D.A.)*, namely *Speed Profile* and *Speed Change*.

- *Speed Profile* contains four logged speeds (m/s) of the ego-vehicle, these speeds are saved at an interval of 0.5 seconds. *Speed Profile* is expressed as:

  $Speed\ Profile = [speed_1, speed_2, ..., speed_4]$

- *Speed Change* computes the division of all subsequent speed values which have been saved prior in *Speed Profile*, these are used to detect braking -or acceleration of the ego-vehicle. *Speed Change* is expressed as:

  $speed\_change_i = speed_{i+1}/speed_i$
  $Speed\ Change = [speed\_change_1, ..., speed\_change_3]$

2. Traffic State features
The second set of features are obtained by observing the *Traffic State Region*, therefore, these features are categorized as *Traffic State (T.S.)* features.

The T.S. features are saved once, to be specific, they are saved just before a Lane Change Prediction is made for the ego-vehicle. This prediction is made when the ego-vehicle reaches point 2, which is located just before the lane-split see Figure 4.2. At this single moment, the T.S. features are added once to feature-vector of the ego-vehicle. Each Lane Change Prediction is thus based on the previously observed *Driver Anticipation (D.A.)* features and the currently observed *Traffic State (T.S.)* features.

The *Traffic State* features are composed of 7 individual features, as listed in Table 4.1 and as explained below.

- *Current Phase Index (cpi)*, an integer value describing the Current Phase Index. Indicating which entry-lane is currently being serviced with a green-phase and which is currently observing a red-phase.
- *Occupancy* considers the occupancy-ratio of each split lane, i.e. it is the total area occupied by vehicles divided by the length of this split lane. The *Occupancy* value implicitly contains the distance between the ego-vehicle and the vehicles which the ego-vehicle approaches after the lane-split.
- *Average Speed (m/s)* considers each of the two shorter lanes after the lane-split, thus within the *Traffic State Region*, for each lane the average speed of its vehicles is saved.

The subsequent features consider the *leading-vehicles* with respect to the ego-vehicle. To clarify, the *last* vehicle on each of the shorter split lanes is the closest vehicle on that lane with respect to the the ego-vehicle. Therefore, these *last* vehicles are also referred to as a *leading-vehicle* with respect to the ego-vehicle. These *leading-vehicles* are expected to influence the behaviour of the ego-vehicle considerably. When a *leading-vehicle* is present on the shorter split lanes, the following features are saved based on the current speed and acceleration of both the *leading-vehicle* and *ego-vehicle*:

The speed of the ego-vehicle might be influenced by both the acceleration and speed of a leading-vehicle. For example, the ego-vehicle might approach a leading-vehicle with a higher velocity, when this leading-vehicles is accelerating. Therefore, the following four features are also computed.

- $speed_{leading-vehicle}$ (m/s)
- $acceleration_{leading-vehicle}$ (m/s^2)
- $Relative\ Speed\ to\ a\ leading\ vehicle = speed_{ego-vehicle}/speed_{leading-vehicle}$
- $Relative\ Acceleration\ to\ a\ leading\ vehicle = acceleration_{ego-vehicle}/accelaration_{leading-vehicle}$

Practical considerations
A vehicle's speed can be accurately measured with radar equipment. This radar equipment has an error range below 1.5 *mph* given a speed of 35 *mph*, or 2.4 *km/h* and 56 *km/h* respectively [29]. Thus resulting in an error percentage of $\frac{\pm\, 1.5\ mph}{35\ mph} * 100\% = \pm\ 4.3\%$.

Although a vehicle's speed can be accurately measured, we also propose to use some acceleration based features. We obtain our acceleration measurements directly from our simulator SUMO. In reality, the acceleration of a vehicle can be computed by measuring its velocity (v_{t_0}, v_{t_1}) at two moments in time, t_0, t_1 respectively. The acceleration can now be computed with: $accelaration = \frac{\Delta v}{\Delta t} = \frac{v_{t_1} - v_{t_0}}{t_1 - t_0}$.

On just the Krauss drivers, noise is applied by our traffic simulator SUMO. However, the acceleration measurements can be even more noisy when they are obtained from real world drivers. Therefore, we also determine the accuracy of our predictive models without these acceleration features in Section 5.3.6.

Feature type	**Name**	**Description**	**Type**	**Value range**
D.A.[1]	Speed Profile	Four logged speeds before the lane-split.	Real	[0, 13.3]
D.A.	Speed Change	Division of subsequent speeds in own speed profile.	Real	$[0, \infty)$
T.S.[2]	cpi	Current phase index of the traffic light.	Integer	{0,2}
T.S.	Occupancy	Occupancy of each split lane.	Real	[0, 1]
T.S.	Average Speed	Average speed of all vehicles on each split lane.	Real	[0, 13.3]
T.S.	Leading Vehicle Speed	Speed of the last vehicle on each split lane.	Real	[0, 13.3]
T.S.	Leading Vehicle Acceleration	Acceleration of the last vehicle on each split lane.	Real	[-7.5, 2.9]
T.S.	Relative Speed to a leading vehicle	Division of ego vehicle speed and speed of last vehicle on each split lane.	Real	$[0, \infty)$
T.S.	Relative Acceleration to a leading vehicle.	Division of ego vehicle acceleration and acceleration of last vehicle on each split lane.	Real	$(-\infty, \infty)$

[1] Driver Anticipation (D.A.)
[2] Traffic State (T.S.)

Table 4.1: Features which compose the feature-vector for predictive models

4.3 Application of LCP and baseline methods to SCHIC

Some details specific to SCHIC are addressed, to clarify in which manner Lane Change Prediction (LCP) and the previously mentioned baseline methods are applied to SCHIC. The baseline methods and LCP are only applied to SCHIC after evaluating two conditions, as explained below and shown in the pseudo-code of Algorithm 2.

Condition 1 – see line 5 of Algorithm 2
Is the lane-split currently blocked by stationary or slowly moving vehicles?
The lane-split can be *blocked* by stationary or slowly moving vehicles, which prevents the *ego-vehicle* from passing the lane-split and reaching its desired entry-lane within a reasonable time. SCHIC uses an expected arrival time for each approaching vehicle. However, when the lane-split is in a *blocked* state it will wait for this vehicle even when it is unable to reach its desired entry-lane. Therefore, when the lane-split is *blocked*, no new predictions are made and current predictions are removed.

The lane-split is considered to be in a *blocked* state when the vehicles of an entry-lane queue up to the lane-split, i.e. between points 2 and 3 of Figure 4.2. This state is also known as *spillback*. In order to detect this *blocked* state, the average speed is observed for the vehicles which are on the lane-split, thus between points 2 and 3 of Figure 4.2. If their speed drops below 1 m/s, the lane-split is considered to be in a *blocked* state.

Condition 2 – see line 7 of Algorithm 2
Is a green phase currently active on one of the reachable short entry-lanes?
SCHIC does not skip green-phases but cycles through all green-phases in a fixed order, where each green-phase has a minimum duration of 5 *seconds*. It is possible that the *ego-vehicle* approaches the intersection, and only observes red signals for all reachable entry-lanes. In this scenario, the *ego-vehicle* will need to wait for its fixedly ordered green-phase while the other phases are active for at least a minimum duration of 5 *seconds*. In this scenario, the performance gain of LCP is assumed to be minimal, therefore, predictions are only applied if one of the short entry-lanes is currently serviced with a green phase.

Correcting mis-predictions – see line 12 of Algorithm 2
When the *ego-vehicle* reaches its final short entry-lane (by reaching point 3 in Figure 4.2) it is confirmed if the *ego-vehicle* has reached its predicted lane. False predictions are now corrected and updated in the local traffic observation of SCHIC.

Algorithm 2: Pseudo-code of Lane Change Prediction (LCP) in use by SCHIC.

```
1  update_lane_change_predictions(lane-split Y)
   Input  : Lane-split Y.
   Output : Lane change prediction for ego-vehicle X which is currently at the end of the
            Driver Anticipation (D.A.) region, see point 2 of Figure 4.2. Previous lane
            change predictions are corrected if needed, incorrect predictions are detected
            when each vehicle has reached its final (short) entry-lane.
2
   // 1. Save and observe the vehicles which are currently: (1) approaching, (2) on or (3)
      have passed the lane-split. 2. Check if the lane-split is blocked by slowly moving or
      stationary vehicles.
3  vehicle_list, lane_split_blocked = get_and_observe_vehicles_around(lane-split Y)
4
5  if lane_split_blocked == True then
       // Remove the predicted vehicles which are currently approaching or on the lane-split,
          since they are unable to move to their desired entry-lane.
6      remove_current_predictions(lane-split Y)
7  else if current_phase_index ∈ {0,1} then
       // Ego-vehicle X needs to be at the end of the Driver Anticipation (D.A.) region, see
          point 2 of Figure 4.2. It approaches two entry-lanes, and one of these is receiving a
          green signal.
8      predicted_lane = get_lane_change_prediction(feature-vector of ego-vehicle X)
9      insert_into_local_traffic_observation(ego-vehicle X, predicted_lane)
10 end
11
   // For each vehicle that reaches its short entry-lane, its predicted lane is updated in the
      correct Road Flow List ($C_{RF,m}$) if it was incorrectly predicted.
12 update_mis_predictions(lane-split Y)
13
14 return    // Road Flow Lists ($C_{RF,m}$) for both possible movements at the intersection (i.e.
     left-turn and straight-ahead) have been updated by inserting a Lane Change Prediction (LCP)
     and the removal of mispredictions.
```

4.4 Conclusion

In this chapter the fundamental structure of the LCP approach is established and it is connected to related work. It established three fundamental ideas that concern **(1)** the belief that anticipative driving styles can be captured in a simple feature vector **(2)** a reasoning that lane change predictions can be used best when one of the reachable entry-lanes is currently serviced with a green signal **(3)** the natural advantage of predictions based on anticipative driving styles. Recall that the existing baseline methods of *fixed-label and split-vehicles* solely rely on historic lane occupancies. The natural advantage of LCP can occur in situations in which other statistical measures only contain a small information gain, e.g. when historic lane occupancies are balanced.

A simplification is made in this study as the predictive output is limited to only two lanes, for either *left-turning or straight-ahead* traffic. In reality, there is a third lane for vehicles which will make a *right-turn* at the intersection. Since *right-turning* traffic can be serviced during the same green-phase as the one for straight-ahead traffic, it does not increase inter-phase uncertainty, However, this study only considers inter-phase uncertainty and therefore *right-turning* traffic is omitted from this study all together. Lastly, reducing the predictive output to either *left-turning or straight-ahead* traffic is likely to increase the predictive accuracy while it does not affect the influence on inter-phase uncertainty.

What was explained in this chapter will be extended in the next, where the predictive models are trained on simulated driver behaviour.

Chapter 5

Accuracy of Lane Change Prediction

The proposed method of Lane Change Prediction (LCP) is finalized in this chapter by covering the training procedure of several learning models and by comparing their predictive accuracies. Additionally, the accuracies of these trained classification models for LCP are compared to the only baseline method of SCHIC which also performs classification namely *fixed label.* This baseline method applies a *fixed label* prediction to each new vehicle, where the label corresponds to the historically observed dominant turning movement.

Section 5.1 fully defines the design of the evaluated intersection. In succession, Section 5.2 informs on the several available driver-models which describe the behaviour of drivers in simulation. This is concluded with a selection of two driver models which are used in our data-sets: the Krauss -and Intelligent Driver Model (IDM). Section 5.3 begins to explain how we model Lane Change Prediction (LCP) as a classification task. Perhaps most important, the accuracies of the trained classification models are compared to each other in Section 5.3.4. This chapter is concluded in Section 5.4.

5.1 Intersection design

This section defines the evaluated intersection of this thesis in such detail that it can be reproduced by others. The dimensions of the intersection are based on a technical drawing as obtained from the municipality of The Hague. However, matters which differ from the real-world intersection are touched upon as well. We give a clear overview of lane-dimensioning, detector placement and the chosen phase set.

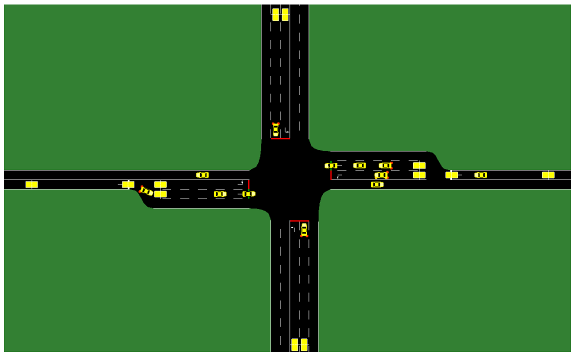


Figure 5.1: Overview of the Waalsdorperweg intersection which contains two lane-splits.

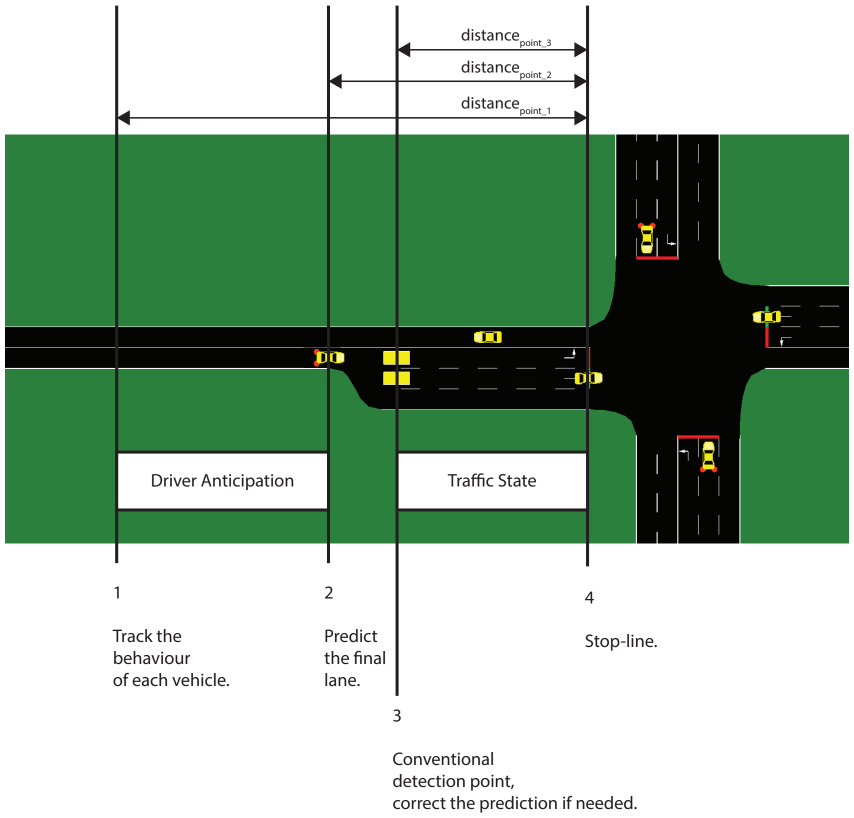


Figure 5.2: This image is a duplicate image, and shows the locations in use by Lane Change Prediction (LCP).

Lane dimensions

The total length of each lane which is leading traffic towards -or away from our intersection is set to $200m$. In reality there is a neighbouring intersection on the right-hand side of our intersection located at roughly $70m$, however the effect of neighbouring intersections is not considered in this work.

As specified by the technical drawing of the municipality, all short entry-lanes are $40m$ in length, i.e. these short entry-lanes are within the "Traffic State" region of Figure 5.2.

Detector placement

The placement of each detector is specified by its distance to the stop-line of the intersection, as shown for two detectors with $distance_{point\_3}$ at the top of Figure 5.2. The travel-time between each detector and the stop-line is defined as follows:
$travel\_time_{detector}[s] = distance_{detector}[m]/speed\ limit[m/s]$.

Detectors on entry-lanes after a lane-split
Due to restrictions in the traffic simulator SUMO, the detector placements are slightly different than the real-world positions. Please refer to Table 5.1 for a clear overview of detector placements.

Here, we consider the entry-lanes after a lane-split. In reality the detectors on these entry-lanes are placed at $distance_{detector} = 35m$ for the lane on the West-side and at $distance_{detector} = 30m$ for the lane on the East-side. Unfortunately SUMO does not support a distance of more than $30m$, since this places the detector on the actual lane-split which is not supported. Therefore, in order to not cover the lane-split, within our SUMO scenario these detectors are all set to $30m$.

As derived from Ortiz et al. [18] (Section 4.1), it is assumed here that an accurate Lane Change Prediction (LCP) is possible at 3.0 seconds before the stop-line of an intersection. At point 2 of Figure 5.2, we obtain the Lane Change Prediction. In order to determine the actual distance at which we make this Lane Change Prediction, the value of 3.0 s is multiplied by the speed-limit of 50 km/h or 13.9 m/s. Thus, $distance_{point_2} = 3.0\ s\ *\ 13.9\ m/s = 42m$.

Detectors on entry-lanes without a lane-split
Here, we consider the entry-lanes without a lane-split, i.e. the vertical lanes of Figure 5.1. In reality, the detectors on these entry-lanes are placed at $40m$. We slightly increased this to $42m$ in order to equalize the detector distances on all four approaches (North, East, South and West) towards the intersection.

Lane type	**Real world** $distance_{detector}$	**SUMO scenario** $distance_{detector}$	**SUMO scenario** $travel\_time_{detector}$
Entry lanes without a lane-split. (vertically oriented lanes in Figure 5.1)	40m	42m	3.0s
Entry lanes after a lane-split (see point 3 in Figure 5.2).	30m (West-side) and 35m (East-side)	30m (West-side) and 30m (East-side)	2.2s

Table 5.1: Detector placements and travel-times are equalized for all entry-lanes.

Phase set

The phase set is shown in Figure 5.3. Given our focus on inter-phase uncertainty, we removed right-turning vehicles, pedestrians and cyclist since they do not affect inter-phase uncertainty (see Section 4.4).

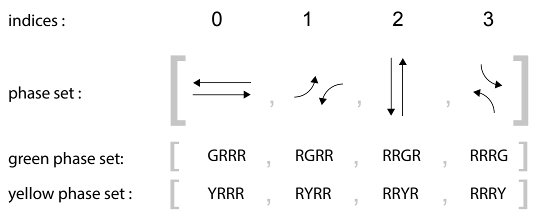


Figure 5.3: An often encountered phase set in The Netherlands, also the default phase set of Simulation of Urban Mobility [19] (SUMO). It is a non-overlapping phase set, since each movement (indicated with an arrow) is only found in one of the green phases.

5.2 Simulated driving behaviour

We have no access to real-world data of vehicles which perform a lane change manoeuvre in our scenario. Therefore, we simulate all vehicle behaviour in SUMO. This simulated vehicle behaviour is used as training -and testing-data for the machine learning models. Within the simulation environment, one can select different car-following models which compute the dynamic behaviour of a vehicle at all times. I.e, the model is applied when a vehicle is closely following another vehicle. But also when a vehicle is approaching an empty lane without someone to follow. The former empty lane-scenario is shown for different car-following models in Figure 5.4.

The level of detail in traffic simulations can vary from macroscopic to microscopic. SUMO is a microscopic traffic simulator, which enables it to model highly detailed dynamic behaviour of each driver. These high levels of detail can lead to valuable clues which enable informed Lane Change Predictions. This section is brought to a close by describing two popular car-following models in more detail, namely the Krauss-model and the IDM.

General categories of car-following models
Based on their logic, car-following models are often divided into the following categories [30]:

- **Gazis-Herman-Rothery models (GHR)** compute a vehicle's acceleration based on the velocity of the leading vehicle and the distance to this leading vehicle. The Intelligent Driver Model (IDM) fits into this category.
- **Safety-distance models** directly adjust the velocity of each vehicle in order to maintain a safe distance to the leading vehicle. This category applies to the Krauss-model [31].
- **Psycho-physical models** take the limited human perception of each driver into account. For example, thresholds are set for the minimal observable speed difference between a follower -and leading vehicle. The Wiedemann-model fits into this category.

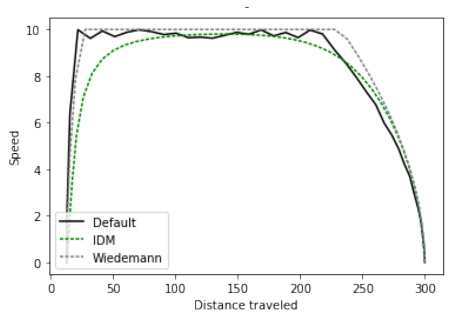
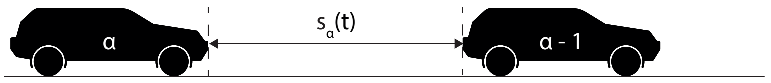


Figure 5.4: Different car-following models determine the dynamic behaviour of each vehicle which approaches an empty lane with a red signal. ("Default" in the legenda refers to the Krauss car-following model)



(a) Car following behaviour of vehicle: α, when following a leading vehicle: $\alpha - 1$.

Variable	**Description**
a_α	Proposed acceleration for the α vehicle
a_{max}	Maximum desired acceleration
b_{max}	Maximum desired deceleration
v_α	Current velocity of the α vehicle
v_{des}	Desired velocity of the α vehicle
δ	Tuning exponent
Δv_α	Velocity difference between the α -and leading vehicle
s_0	Minimum desired net distance to the leading vehicle
T	Desired headway in seconds
τ_k	Reaction time of the driver
Δt	Time step

(b) Explanation of variables used by car following models [32]

Table 5.2: Car following behaviour of vehicle α and explanation of its variables.

Krauss Car-Following Model
Stefan Krauss introduced the Krauss car-following model [31] in 1998, although slight modifications are applied to this model, it is currently the default car-following model of SUMO [1]. In this model the velocity of a vehicle is directly adjusted, without deriving it from the acceleration. The main goal is to maintain a safe distance to the leader vehicle.

Figure 5.5 contains the main equations of this model. First of all a safe velocity: v_{safe}, is computed by Equation 5.1. Two important variables in order to compute v_{safe} are the reaction time of the driver: τ_k and the maximum desired deceleration in non-emergency situations: b_{max}.

The desired velocity is simply limited in Equation 5.2, either by the just computed safe velocity: v_{safe}, the speed-limit: v_{max} or by the current velocity times the maximum desired acceleration. This limitation should ensure a safe gap to the next vehicle.

In the final step a random perturbation: $\eta > 0$ is subtracted from the desired velocity which results in the velocity for the next time step: $v_\alpha(t + \Delta t)$, see Equation 5.3.

[1]A link to SUMOs documentation on car-following models and default paramater settings

$$v_{safe}(t) = v_{\alpha-1}(t) + \frac{s_\alpha(t) - v_\alpha(t)\tau_k}{\frac{v_{\alpha-1}(t)+v_\alpha(t)}{2b_{max}} + \tau_k} \tag{5.1}$$

$$v_{des}(t) = min(v_{safe}(t), v_{max}, v_\alpha(t) + a_{max}\Delta t) \tag{5.2}$$

$$v_\alpha(t + \Delta t) = max(0, v_{des}(t) - \eta) \tag{5.3}$$

Figure 5.5: Formulas to compute the velocity of a vehicle by the Krauß car following model [32].

Intelligent Driver Model (IDM)
The Intelligent Driver Model (IDM) is introduced by Treiber [33] in 2000 as a deterministic model for highway traffic. In 2017 Treiber incorporated stochastic properties in order to model a common phenomena, i.e. traffic flow oscillations of congested traffic [34]. Treiber identifies three possible causes of traffic flow oscillations: instabilities in traffic flow, finite human perception thresholds (i.e. indifference regions) and external acceleration noise. These three possible causes are added to IDM in isolation by Treiber [34], in order to investigate the relative importance of each of them.

Simulations of this stochastic version of IDM are compared to real-world data which is acquired from experimental car-platoons in China. It is demonstrated that within simulation, similar results are obtained by either adding external acceleration noise or by applying finite human perception thresholds with action points. For inner city traffic with relatively low speeds of 30 km/h, action points or external noise can both accurately reproduce the oscillations as observed in the real-world data.

However, this stochastic variant of IDM is not implemented in SUMO. Instead, the original deterministic IDM is implemented in SUMO. This was concluded based on the source code of SUMO. To confirm our findings on the source code, a forum post was made, where the main developer of SUMO confirmed our findings.

A brief description of the formulas used in deterministic IDM is given below.

See Equation 5.4. The acceleration of the following vehicle: $a_\alpha(t)$, is computed based on the current velocity of the following vehicle: v_α, the velocity difference with the leading vehicle: Δv_α and the distance to the leading vehicle: s_α.

See Equation 5.5. In order to compute the previously explained acceleration of the following vehicle: $a_\alpha(t)$, one first needs to compute: $s^*(.)$. This $s^*(.)$ is a summation of the following distances: the minimum desired distance to the leading vehicle: s_0, the desired headway distance given the current velocity of the α vehicle: $v_\alpha(t)T$, and a distance which compensates for the braking of the leader.

$$a_\alpha(t) = a_{max}(1 - (\frac{v_\alpha(t)}{v_{des}(t)})^\delta - (\frac{s^*(.)}{s_\alpha(t)})^2) \tag{5.4}$$

$$s^*(v_\alpha(t), \Delta v_\alpha(t)) = s_0 + v_\alpha(t)T + \frac{v_\alpha(t)\Delta v_\alpha(t)}{2\sqrt{a_{max}b_{max}}} \tag{5.5}$$

Figure 5.6: Formulas to compute the acceleration of a vehicle by the IDM car following model, Pourabdollah [32].

Parameter settings of car-following models

In order to reproduce the experiments as conducted in this thesis, the parameter settings of each car-following model are listed in Table 5.3. These settings are the default parameter settings of each model [2] .

Driver Model	**Variable**	**Description**	**Value**
Krauss, IDM	accel	Normal (non-emergency) acceleration ability	2.9 m/s^2
Krauss, IDM	decel	Normal (non-emergency) deceleration ability	7.5 m/s^2
Krauss, IDM	maxSpeed	Maximum speed	13.8 m/s
Krauss, IDM	minGap	Minimum gap when standing to leader vehicle	2.5 m
Krauss, IDM	length	Vehicle length	4.3 m
Krauss	sigma	Driver imperfection with range [0,1] (0 denotes perfect driving)	0.5

Table 5.3: Explanation of default parameter settings of car-following models.

Conclusion and selection of car-following models

In order to make meaningful predictions on lane changes, the car-following model should resemble real-world behaviour as much as possible. IDM is a deterministic model while the Krauss car-following model incorporates stochastic properties. In reality nobody drives deterministic, however some drivers do tend to anticipate smoothly on upcoming traffic as occurs with the Intelligent Driver Model. Other drivers might use a more aggressive approach in traffic, but always maintain a safe distance to leading vehicles, which matches the objective of the Krauss driver model.

Both models are commonly used in research on traffic control, and in this thesis each model is evaluated in isolation for Lane Change Prediction (LCP).

[2]A link to SUMOs documentation on default paramater settings for car-following models.

5.3 LCP as a classification task

Given the preceding description of the intersection and how we can simulate traffic in it, we can now formalize our approach to Lane Change Prediction (LCP) as a classification task.

First of all, Section 5.3.1 stipulates the formation of the training -and testing-sets. The baseline methods of SCHIC are explained in Section 5.3.2. Thereafter, Section 5.3.3, explains several Machine Learning models and their training procedures. The accuracies of the trained classification models are compared in Section 5.3.4. Additionally, some promising classification models are further evaluated by means of Learning Curves in Section 5.3.5. The proposed acceleration based features (see Section 4.2) could be quite noisy when obtained in reality. Therefore, we obtain the accuracy scores of our models without these acceleration features in Section 5.3.6.

5.3.1 Generation of training -and testing sets

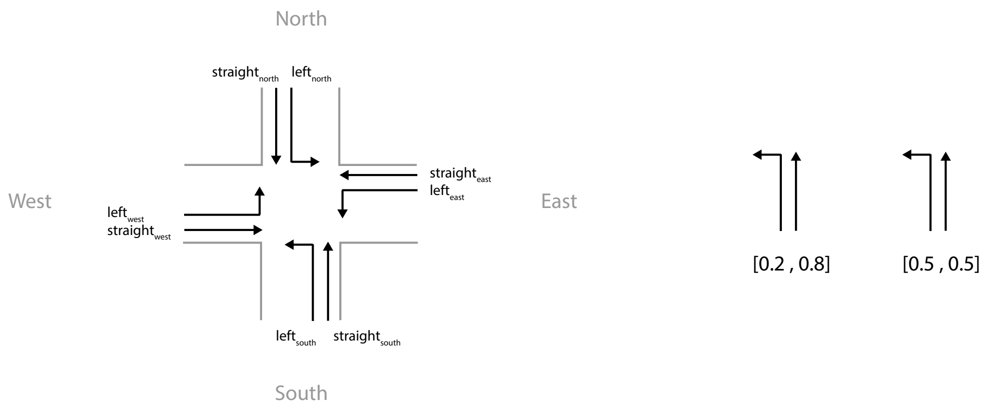


Figure 5.7: Visual representation of the intersection. Each arrow in this figure is an Origin -and Destination pair, and defines the movement of a vehicle through the intersection. Each Origin -and Destination pair is composed by selecting an *Origin* and an allowed *Destination* from the set $Directions = \{North, East, South, West\}$. For example, an allowed Origin-Destination pair is South-West which is a left-turn at the intersection.

Distribution of turning-movements
The predictive models are evaluated on two commonly found traffic scenarios which differ in the distribution of left-turning and straight-ahead traffic – respectively [left-turning, straight-ahead] = [0.2, 0.8] or [0.5, 0.5] as illustrated in Figure 5.7.

The [20,80] scenario considers a distinction between major -and minor traffic flows as is often made in research and practice, for example as mentioned in a field study by Smith in 2015 [8]. In this thesis the major flow (straight-ahead traffic) is assumed to contain 80% of all traffic which leaves 20% for the minor flow (left turning traffic). The second [50, 50] scenario, is a balanced one in which both traffic flows receive an equal number of vehicles. The baseline methods of SCHIC use the observed distribution of these traffic flows, i.e. either [20,80] or [50,50]. In order to realize a fair comparison between these baseline methods and the learning models, both need access to the same information, therefore the learning models are trained on these different distributions in isolation.

Traffic generation method

All traffic scenarios are obtained by using a single driver model, i.e. either the Intelligent Driver Model or Krauss. Origin-Destination pairs are used to specify the number of vehicles from each origin-lane to each destination-lane, as shown in Figure 5.7. Although the distribution of turning-movements differ, the vehicle demand from all four directions (North, East, South, West) are equal. Four hourly vehicle demands are used and each demand is composed of 15 individual hourly simulations. Each hourly simulation contains randomized arrival times of vehicles, since a different seed is used in SUMOs traffic generator: OD2TRIPS.[3]

The hourly vehicle demands used for our single intersection are based on SCHICs original paper, where Xie et al. [1] evaluate the performance of SCHIC on a single intersection which contains some differences compared to our single intersection. In their scenario, traffic is found on only two lanes, i.e. only straight-ahead traffic originating from the North and West is considered see Figure 5.7. Therefore, two green phases for straight-ahead traffic suffice in their scenario. Their vehicle demand varies from [600, 800, ..., 1200] (vehicles / hour).

However, in this thesis left-turning traffic is scheduled as well and four green phases are in use instead of just two. Using four green phases instead of two means that each vehicle needs to wait longer before his green phase is reached, since SCHIC cycles through each phase in a fixed order where each green phase runs for at least a minimum duration. Therefore, the hourly demands on the intersection of this thesis are slightly lower and vary between [200, 400, 600, 800] (vehicles / hour). These hourly demands are considered to be low to moderate, however related to the original publication on SCHIC since traffic is now approaching on eight different lanes and serviced in four green phases instead of just two.

Sample counts in training -and testing-sets

For both the [20,80] and [50,50] data-set, 300 independent hourly simulations are combined in total and contains the four different hourly vehicle demands. Therefore, a single predictive model is trained for all four hourly vehicle demands. Each data-set is split into two parts, i.e. 80% (240h) is used for training and 20% (60h) for testing.

As stated prior in Section 4.3, the Lane Change Prediction (LCP) is only applied under the following two conditions: 1) an approaching vehicle needs to observe a green phase on one of its reachable entry-lanes 2) an approaching vehicle needs to observe a ”non-blocking” lane-split, so it can swiftly move to its desired entry-lane. Since the model is only applied under these two conditions it is also only trained under these two conditions. Therefore these two conditions limit the number of samples on which a model is trained, the sample counts of the training and testing-sets are listed in Table 5.4.

[3]A link to SUMOs documentation on OD2TRIPS.

Turn distribution	**Training-set** Krauss	**Testing-set** Krauss	**Training-set** IDM	**Testing-set** IDM
[0.5, 0.5]	7981	1962	7838	1957
[0.2, 0.8]	7804	1954	7902	2014

Table 5.4: Sample counts in training -and testing-sets.

Traffic light controller
The traffic light controller is not considered to influence the driving behaviour towards the intersection significantly. Therefore, for computational speed-up the training -and testing-sets are derived from a VA controller instead of a SCHIC controller. Later on in Chapter 6, Lane Change Prediction (LCP) is applied on a SCHIC controller. However, the predictive accuracies of LCP with the SCHIC controller match closely with the predictive accuracies which will be given in this chapter which are obtained with a VA controller instead.

Notes on bias
A considerable bias is found in the [20, 80] training-set, here 80% of all traffic continues straight ahead, so a model which predicts all vehicles to continue straight-ahead results in an high accuracy of 80%. Luckily, this bias is removed in the balanced [50,50] training-set.

A second form of bias considers the historic lane occupancies. The predictive model is aimed to be trained on just the drivers behaviour, although the feature vector also contains historic data on lane occupancy. Recall that lane occupancy is defined as the total length of all vehicles on a lane divided by the length of that lane. Therefore, the lane occupancy implicitly contains the distance from the ego-vehicle (for which a prediction is made) to the vehicles which it is approaching on subsequent lanes. Some measures have been taken to prevent the model from being biased towards historic lane occupancies.

Most importantly, we aim to prevent a shift towards Time Series Classification (TSC). To clarify, Time Series Classification (TSC) would use a large history of observations of this lane occupancy feature over time in order to predict the lane occupancy for the near future. For example, with TSC the feature vector can contain the lane occupancies of the last hour which is used to predict the lane occupancy for the next minute. Whereas our predictive model only has access to a single observation of the lane occupancy feature in its feature vector. Therefore, the lane occupancy feature is not given in series to the predictive model, but instead, the lane-occupancy feature is only found once in our feature vector. Doing so we aim to prevent a shift towards Time Series Classification (TSC).

Lastly, a single hourly vehicle demand could correlate highly with a mean and maximum value for each lane occupancy value. However, the model is trained on four different hourly demands. Both measures combined, aim to reduce the influence of time series information as much as possible. However these arguments are only valid for the balanced [50,50] training-set. In case of a [20,80] distribution the lane occupancies are able to bias the model towards these historic values.

5.3.2 Baseline methods of SCHIC

In this subsection we explain classification models which are based on SCHICs baseline methods.

Fixed label
This baseline method of SCHIC uses a fixed label prediction which corresponds to the observable major turning-movement. The accuracy of this model depends on the turning-movement distribution in the used test-set, i.e. [50, 50] or [20, 80] in our case, thus the accuracy of a *fixed-label* is respectively 0.5 or 0.8.

Probability classifier
This method is loosely based on SCHICs baseline method *split vehicles*, which splits vehicles based on the observed turning movement distribution. However, in order to compare *split vehicles* to a classification methods it is first converted into a classification method as described below. SCHICs baseline method splits each vehicle into two fractions, for example $\{0.2, 0.8\}$. However, the probability classifier generates a random number between [0,1] for each vehicle that will receive a Lane Change Prediction. Afterwards, the turning movement distribution is used to predict a single lane. In case of a [0.2, 0.8] distribution, a random number below 0.2 is labelled as a left-turn (label = 0) and values above 0.2 as straight-ahead traffic (label = 1).

5.3.3 Selected Machine Learning methods

Here, we supply implementation details of our proposed machine learning models, i.e. Support Vector Machine (SVM), Decision Tree and a Random Forest. These machine learning models are based on the literature study on Lane Change Prediction (LCP) (Section 4.1).

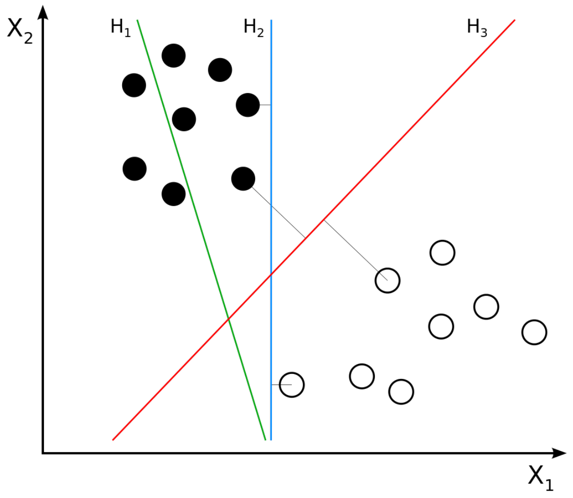
Support Vector Machine (SVM)
This model did not follow directly from the performed literature study, but is included as a reference. In this study the following implementation is used: *sklearn.SVM.*

Recall that with the Krauss driver model a safe velocity is computed for each vehicle in order to maintain a safe distance to its leading vehicle (Section 5.2). Afterwards a random perturbation is either added or subtracted from this deterministically computed safe velocity. Therefore, the velocity will always deviate from the deterministically computed safe velocity within the range of the random perturbation.

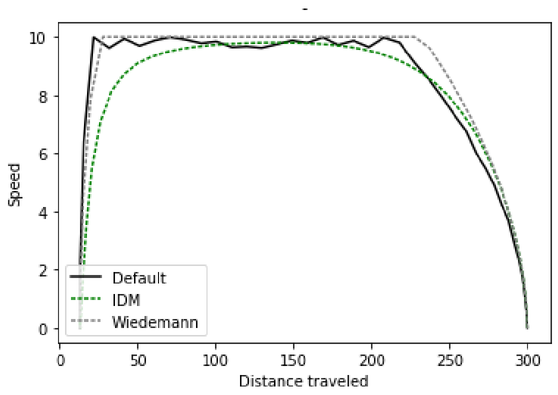
In our classification problem, each feature vector belongs to either of the following two classes: {*left-turn, straight-ahead*}. With a SVM, each data point (feature vector) is seen as a p-dimensional vector. The aim is to separate these data points as well as possible with a (p-1) dimensional hyperplane. Classification for a new data point is simply performed by checking on which side of the hyperplane this new point lies, with each side corresponding to a single class. An example is given for $p = 2$, where the filled and unfilled circles represent two different classes in Figure 5.8a.

Multiple hyperplanes might be able to fully separate both of the two classes, e.g. H_2 and H_3 both separate the two classes. However, H_3 has a wider margin towards the different data points than H_2. This margin can be measured by the perpendicular ($\perp$) distance from a hyperplane towards the nearest point of each different class. Maximizing this margin-distance is beneficial, since small variations in the data points will be less likely to change the classification output of the model [35].

The randomization found in speed-profiles for Krauss vehicles might be separated quite well by linear vectors as occurs with an SVM model, see Figure 5.8b.



(a) Two classes, as indicated with a filled black circle and unfilled circle respectively. Different hyperplanes (H_1, H_2, H_3) aim to separate the two classes. H_2 and H_3 both fully separate the two classes. Source: `https://en.wikipedia.org/wiki/Support_vector_machine`, accessed on 17-05-2020.



(b) Different car-following models determine the dynamic behaviour of each vehicle which approaches an empty entry-lane with a red signal. ("Default" in the legend refers to the Krauss car-following model)

Figure 5.8: Visual explanation of the hyperplanes of a Support Vector Machine (SVM) and a speed profile for several driver-models in the simulation package SUMO.

Decision Tree

In this section we explain the working principle of a Decision Tree and the manner in which each tree learns from the training-data.

A decision tree as proposed by Hou et al. [28] followed directly from the literature study (section 4.1). In this thesis, the following implementation is used: *sklearn.tree.DecisionTreeClassifier* and its hyper parameters are listed below in Table 5.5.

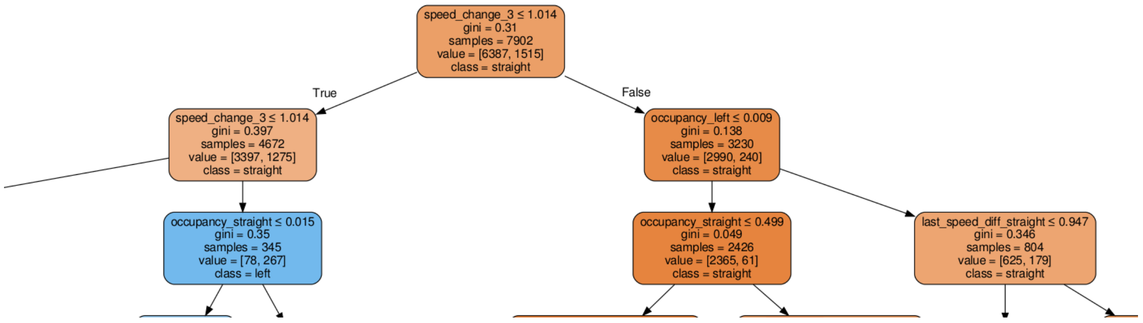


Figure 5.9: Top part of a Decision Tree for the IDM driver-model, showing the root node and some branches.

Working Principle A decision tree uses a feature vector as input and performs a sequence of tests on it along a path of nodes, which results in a classification decision [36]. Each internal node of the tree performs a test, with the following general structure. "Is feature $x_i \leq a$?", with a being a threshold value. The outcome of each test is thus a binary answer, which corresponds to a descendant node in which the next test is performed on this feature vector. At the end, this feature vector traversed a path throughout the tree by reaching a terminal node which returns a class label.

Training procedure To assemble a decision tree, the tests performed by each node should be determined, as well as the ordering of these nodes. By performing each test, a tree iteratively splits its training set into two branches. Where each branch is a sub-tree, which receives a subset of the training-set.

Each node t receives a subset of the the training set X_t. The root node (or starting node) is assigned with the entire training-set: X_t. The goal of each binary split is to create two subsets that are more homogeneous than the parent subset. Our implementation, i.e. *sklearn.tree.DecisionTreeClassifier*, uses the Gini impurity to measure the quality of a split [37].

The Gini impurity $I_G(t)$ [38], can be computed for each subset X_t of each node t. The Gini impurity measures how often a randomly chosen element from the entire training set would be incorrectly labelled based on the distribution of labels in the subset X_t. Therefore, Gini impurity is computed by summing the probability that an item p_i is chosen from the entire training set, times the probability of incorrectly labeling that item in subset X_t, i.e. $\sum_{k \neq i} p_k = 1 - p_i$.

The Gini impurity $I_G(t)$ can thus be defined as shown below. J defines the number of classes, in our scenario we label each feature vector as {left-turn, straight-ahead}, i.e. $i \in \{1, 2, .., J\} = \{1, 2\}$ *with* $J = 2$. And p_i is the fraction of items labelled with class i in subset X_t.

$$I_G(t) = \sum_{i=1}^{J} (p_i \sum_{k \neq i} p_k) = 1 - \sum_{i=1}^{J} p_i^2$$

The optimal value of $I_G(t) = 0$ which is achieved when all items of subset X_t are assigned to a single label i. Therefore, the lowest impurity at a node t, i.e. $I_G(t)$, indicates the best possible split.

The quality of a split, can be referred to as the decrease in node impurity of the parent node $\Delta I_G(t)$. After the binary split at node t its subset X_t is split and assigned to two nodes, the next node with answer "yes" t_Y is assigned with subset X_{t_Y}, and the next node with answer "no" t_N is assigned with subset X_{t_N}. The quality of a split is computed weighted by the number of vectors N_{t_Y} and N_{t_N} in each subset X_{t_Y} and X_{t_N}.

$$\Delta I_G(t) = I_G(t) - \frac{N_{t_Y}}{N_t} I_G(t_Y) - \frac{N_{t_N}}{N_t} I_G(t_N)$$

After an exhaustive search for all candidate questions, the candidate question with the maximum impurity decrease is chosen at node t.

Stop-Splitting Criteria and Class assignment A tree can be expanded until all its terminal nodes are pure, i.e. when all items of its subset X_t correspond to a single class label. This can lead to a large tree, which is likely to over-fit on the training set.

Therefore, we set a maximum depth of 25 to limit the growth of our decision tree. Secondly, we only allow a node to be split if it contains at least 2% of the entire training set. The tree is not further expanded when either of the these two conditions are violated, see Table 5.5.

Let y_1 and y_2 describe our two classes: *left-turn* and *straight-ahead*. When a terminal node is reached, the class label j is given by computing the dominant class label in its subset X_t:

$$j = \arg\max_i P(y_i|t)$$

	Decision Tree	**Random Forest**
Maximum depth of each tree.	25	25
Minimum number of samples per node in each tree, defined as a fraction of the entire training set X_t.	0.02	0.02
Number of features during node-splitting.	$n_{features}$	$\sqrt{n_{features}}$

Table 5.5: Parameter settings of the Decision Tree and Random Forest.

Random Forest (RF)
A Random Forest is an ensemble method, since it combines several individually trained Decision Trees [39]. The aim of ensemble methods is to combine the predictions of several classifiers, in order to improve the robustness over just a single classifier. With a Random Forest the individual predictions of all Decision Trees are averaged in order to reach a single prediction, this often improves the variance when compared to each of its individual Decision Trees. The predicted class label of an input feature-vector is a vote by the Trees in the Forest, this vote is weighted by the probability estimate of each Tree. Therefore, the predicted class label is the one with the highest mean probability estimate across all Trees.

Each Decision Tree is trained individually on a different sub-samples X_{tree_n} of the original training-set X. The sub-sample size of X_{tree_n} is identical to the size of the original training-set X. However, the feature-vectors of X_{tree_n} are drawn with replacement from X (i.e. bootstrap sampling). Drawing a feature vector with replacement means that each feature vector is allowed to occur multiple times in a sub-sample X_{tree_n}. For example, when the original training-set contains five feature-vectors (denoted as $X = [1, 2, 3, 4, 5]$), drawing with replacement can lead to the following sub-samples X_{tree_n} which contain duplicates of the same feature-vector.

- $X = [1, 2, 3, 4, 5]$
- $X_{tree_1} = [1, 2, 3, 3, 5]$
- $X_{tree_2} = [1, 1, 2, 3, 5]$

Secondly, when determining the split at each node during the tree construction, the best split can be found by considering all features ($n_{features}$) or by considering a subset ($\sqrt{n_{features}}$). A single Decision Tree uses all features ($n_{features}$), whereas a Decision Tree of a Random Forest uses the smaller subset ($\sqrt{n_{features}}$).

In conclusion, each individual Decision Tree is now trained on a different training-set X_{tree_n}. Secondly, each Decision Tree of the forest receives a small set of features to determine the best split at each node. The above two sources of randomness aim to decrease the variance of a Random Forest.

In this study the following implementation is used: *sklearn.ensemble.RandomForestClassifier, version 0.19.1.* The Random Forest is composed of 100 individually trained decision trees and its parameters are shown in Table 5.5.

5.3.4 Accuracies of trained classifiers

The accuracy scores of these trained predictive models are evaluated on the previously described test-sets. A separate test-set is used for each driver model and distribution of turns. (Section 5.3.1)

Scenario A: Intelligent Driver Model (IDM)
In general, the Random Forest and Decision Tree classifiers show a high accuracy on the deterministic IDM model, as shown in Figure 5.10.

In the [50,50] case, all models are trained on a balanced training-set and an accuracy of roughly 0.8 is reached by the Decision Tree and Random Forest. Given this balanced turning movement distribution, the fixed label method obviously results in an accuracy of 0.5. The SVM model does not perform properly given its accuracy of 0.59, which is close to the performance of a fixed label (acc = 0.5). The higher accuracies of 0.8 indicate that the Tree and RF model have learned to predict lane changes quite well based on the supplied training-data, i.e. the driving style of each vehicle.

In the [20,80] case, the models might be strongly biased since they are trained on this unbalanced turning movement distribution. Again, fixed label and SVM perform similar by reaching an accuracy of, respectively 0.80 and 0.81. However, an accuracy of 0.89 is reached for both the Decision Tree and Random Forest. This accuracy indicates that these models do not only use the bias of the unbalanced turning movement distribution, but additionally use the actual features as observed in the driving style.

Scenario B: Krauss driver model
In general, all classifiers show a weak accuracy for the Krauss driver model. Based on the confusion matrix of the Random Forest classifier, we observed a fixed label prediction in the unbalanced [20,80] scenario. Potential reasons for this weak performance are listed below.

The Krauss car following model aims to just keep a safe distance to the next vehicle. Doing so, it anticipates less on the upcoming traffic state than an IDM vehicle. Lastly, this model contains stochastic properties, which increases the difficulty of the learning process compared to the deterministic IDM vehicles.

Conclusion
The Random Forest performs most accurate on both test-sets of Scenario A. The Random Forest and Decision Tree models are further evaluated by means of learning curves in the next section, in part to judge the size of the training-set.

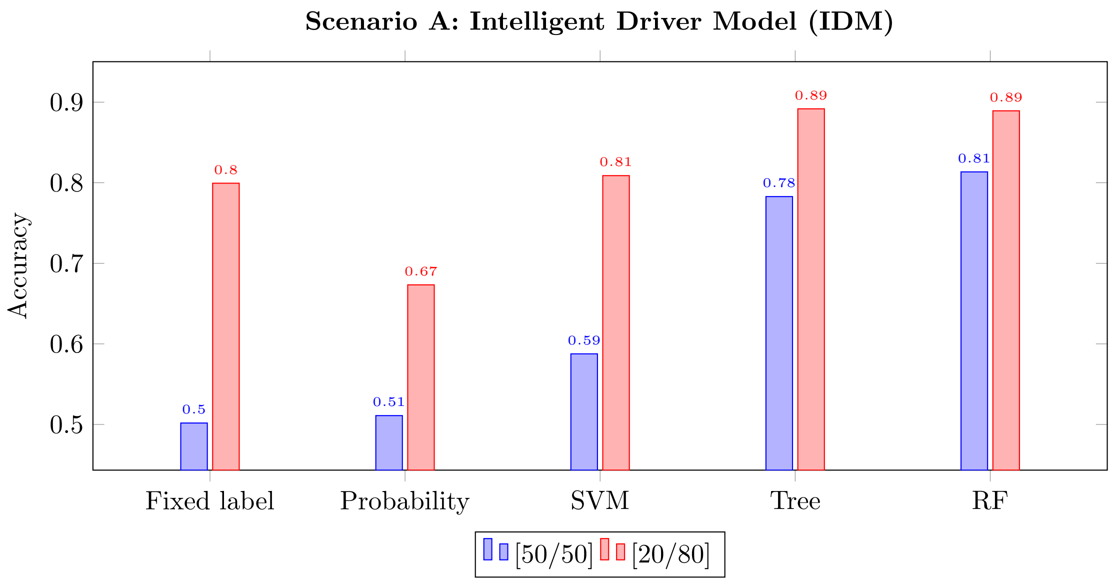


Figure 5.10: Scenario A: Intelligent Driver Model (IDM). Accuracies of different predictive models on two test-sets with different turning-movement distributions, i.e. [0.5, 0.5] and [0.2,0.8].

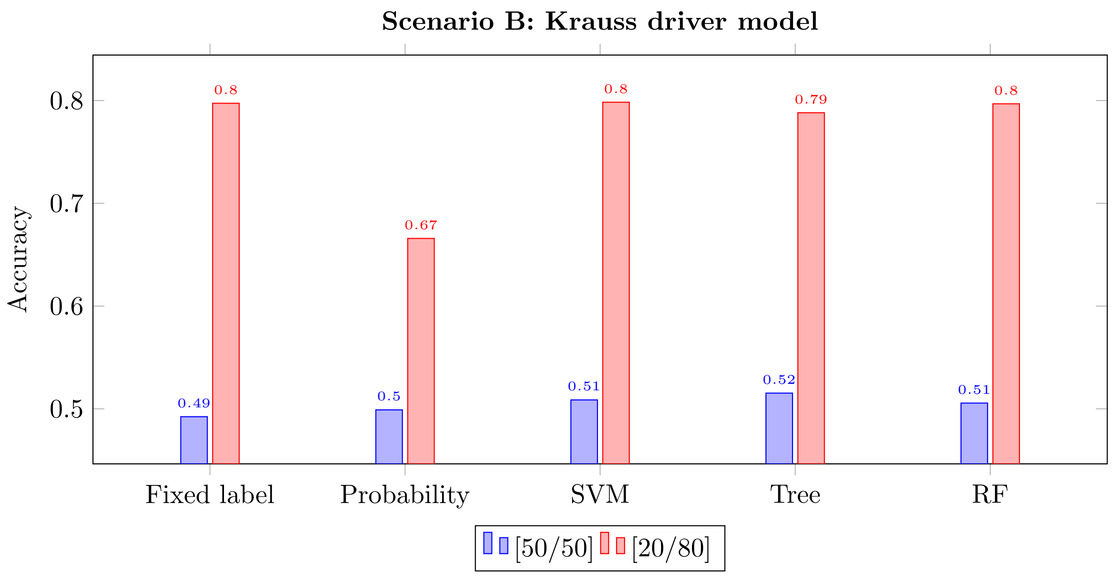


Figure 5.11: Scenario B: Krauss driver model. Accuracies of different predictive models on two test-sets with different turning-movement distributions, i.e. [0.5, 0.5] and [0.2,0.8].

5.3.5 Model validation with learning curves

After choosing a set of machine learning models and their hyper parameters, it is important to evaluate how well each model performs both after and during its training procedure. During training each model learns from the training-data which is given to it, and adjusts itself to perform well on this training-data. Ideally, the models performance will be more accurate and robust when it is fed with more training data. When this occurs the model learns and generalizes well. However, it is important to diagnose if a model is actually learning properly from his training data, we diagnose this with learning curves, as explained below.

Generating a learning curve
A learning curve plots the average performance of several models on the Y-axis, where each model is trained on an incrementally increased training-set as shown on the X-axis. The learning curves are generated as follows. A cross-validation generator splits the entire dataset k times (with k=7) in training and validation data. Subsets of the training set, with increasing sizes, are used to train the model and an accuracy score is obtained for each training subset and the validation set, respectively called training score and validation score. Lastly, these scores are averaged over all k=7 runs for each training subset size.

The learning process of each model is diagnosed with two performance metrics, training score and validation score, as explained below:

- ◇ **Training score:** an accuracy-score based on data on which the model has been trained before.
- ◇ **Validation score:** an accuracy-score based on novel data, i.e. the model performs a prediction for novel data which it has not seen during training.

Interpreting a learning curve
The training-score will thus always be higher than the validation-score, since the model made a prediction for data it has seen before. To clarify, it has seen this feature vector and actual label before in its training phase. The training-score is expected to decrease when the training size increases, since it is more difficult to fit a model well on a larger set of training data. Contrary to the training-score, the validation-score is based on unseen validation-sets generated by K-fold cross validation (k=7). The validation-score is expected to increase with a larger training size, since the model will learn to generalize better. When the training -and validation scores converge it indicates that the model has reached its optimal performance. Since it now performs similar for data that it has seen before, as for data which it did not see before.

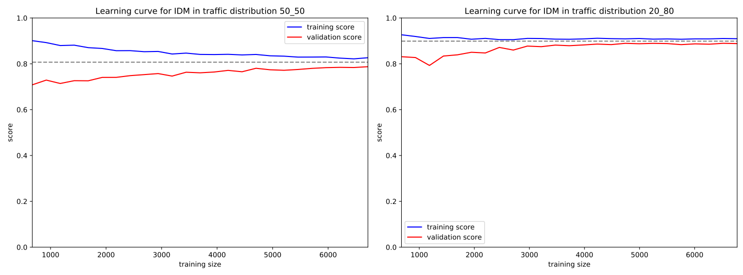
Learning curves are also used to identify under-fitting and over-fitting of a model.

- ◇ **Under-fit (biased model)** Some models are said to under-fit, since they are unable to learn and adjust themselves properly based on an increase in training-data. Their predictive accuracy remains low and constant despite an increase in training-data.
- ◇ **Over-fit (high variance model)** It is possible that a model adjusts itself erroneously on his training-data, i.e. instead of generalizing well on the training-data it over-fits on noise which is only found in this specific set of training-data.

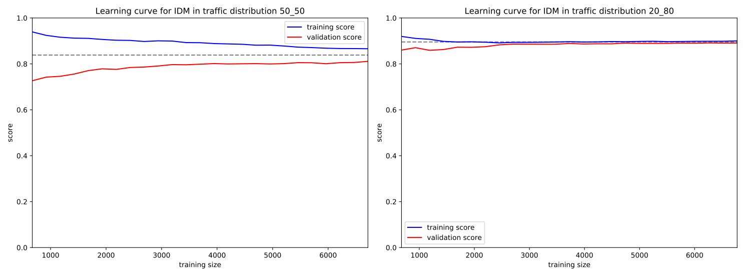
In conclusion, learning curves are a tool to find out if the model will benefit from adding more training-data and if the model suffers from under-fitting or over-fitting.

As concluded in Section 5.2, two driver models are used to represent the general problem domain in which the learning process occurs. On the next pages, learning curves are evaluated for Scenario A: IDM and Scenario B: Krauss driver model.

Scenario A: Intelligent Driver Model (IDM)



(a) Learning-curves for a Decision-Tree classifier.



(b) Learning-curves for a Random Forest classifier (composed of 100 Decision-Trees).

Figure 5.12: Learning curves on the IDM driver-model for both training-sets: [50,50] and [20,80]. K-Fold cross validation (k=7) is used to derive the validation score.

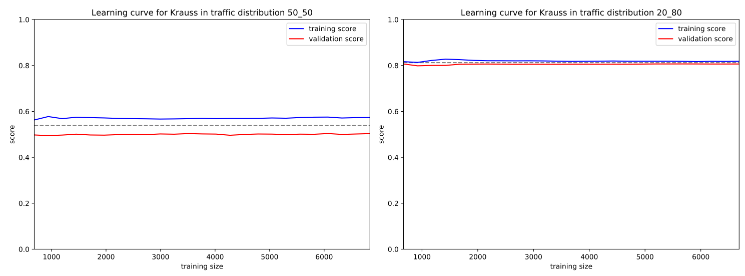
Here, we focus on the deterministic, Intelligent Driver Model (IDM). Figure 5.12a shows the learning curves for the Decision Tree classifier and Figure 5.12b shows these for the Random Forest. The two graphs on the left consider the balanced [50,50] turning movement distribution, whereas the two graphs on the right consider the unbalanced [20,80] turning movement distribution.

In the [50,50] case, i.e. the graphs on the left, the Decision Tree and Random Forest show a similar performance and both converge after roughly 6000 training-samples.

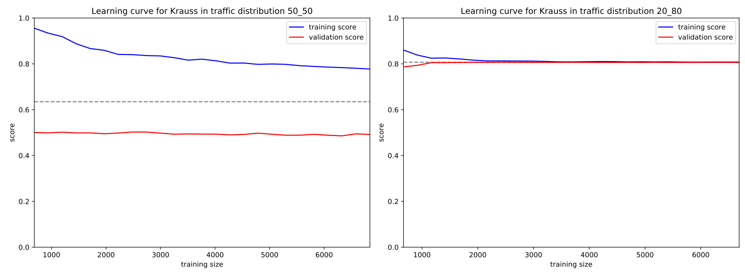
In the [20,80] case, i.e. the graphs on the right, the convergence point is reached with less training data when compared to the [50,50] case, this holds for both models. The rapid convergence in the [20,80] case is most likely a result of the bias in the training-set. The models are now trained on a [20,80] distribution of turning-movements instead of on a balanced, [50,50] distribution. In the [20,80] case, the Decision Tree converges at a later point than the Random Forest. The Random Forest contains 100 individually trained Decision Trees and uses a majority voting principle, it is therefore more likely to converge faster than a single Decision Tree.

Both models learn to predict the behaviour of the deterministic IDM driver model quite well. Since IDM is deterministic, there is little noise and variation expected in the training-data. It is trivial, that low noise in the training-date reduces the risk that a model is able to over-fit on noise. A fixed label prediction will only apply the bias which is found in the training-set and will yield an accuracy of 0.5 or 0.8, respectively for the [50,50] and [20,80] training-sets. However, the convergence of the validation -and training-score towards respectively 0.8 and 0.9 for both training sets is well above the bias found in these training sets. Therefore, the learning curves indicate that the models have learned to predict lane changes for the IDM model properly, and will not benefit from adding more training data.

Scenario B: Krauss driver-model



(a) Learning-curves for a Decision-Tree classifier.



(b) Learning-curves for a Random Forest classifier (composed of 100 Decision-Trees).

Figure 5.13: Learning curves on the Krauss driver-model for both training-sets: [50,50] and [20,80]. K-Fold cross validation (k=7) is used to derive the validation score.

Here, we focus on the stochastic Krauss driver-model. Figure 5.13a shows the learning curves for the Decision Tree classifier and Figure 5.13b shows these for the Random Forest. The two graphs on the left consider the balanced [50,50] turning movement distribution, whereas the two graphs on the right consider the unbalanced [20,80] turning movement distribution.

In the [50,50] case, i.e. the graphs on the left, the Decision Tree shows a horizontal training-score of roughly 0.5. The horizontal trend demonstrates that no learning occurs based on an increase in training-data, as was expected by the accuracy score of 0.5 for both the Decision tree and RF model in Section 5.3.4. The Random Forest classifier has a large gap between the training -and validation score, which could indicate a lack of training data. Given that the validation score is almost horizontal, it is again unlikely that any learning is going on, so more training data is unlikely to increase the performance. Since there is no actual learning process observed, these models are likely to under-fit on the data.

As for the [20,80] case, i.e. the graphs on the right, no proper learning-effect is observed. Both models rapidly converge towards the bias found in the training data with a [20,80] distribution of turns. The rapid convergence of training -and validation-scores to a value of 0.8 indicates that learning is not as intended, i.e. it is not based on the behaviour of each driver. The models are highly biased by the distribution of turns found in the training-set and are therefore likely to over-fit on the training-data.

In conclusion, for these two learning models there is most likely too little variance in the feature-vector which describes the driving behaviour of the Krauss driver-model. A more advanced machine-learning model might learn driving behaviour from the current feature vector, and would demonstrate this most in the balanced [50,50] case with an accuracy above 0.5. Secondly, the feature vector could be enhanced to contain additional variables.

We consider the identified enhancements for Lane Change Prediction (LCP) with the Krauss driver model as out of scope for this research project. And instead, we focus on the higher accuracies as obtained with the IDM driver model in order to answer our next research question which considers the traffic flow improvement of LCP when it is applied to SCHIC.

5.3.6 Predictive accuracies without acceleration features

The proposed acceleration based features (see Section 4.2) could be quite noisy when obtained in reality. Therefore, we measure the predictive accuracy of our models without these two acceleration based features:

1. Leading Vehicle Acceleration.
2. Relative Acceleration to a leading vehicle.

The predictive accuracies without these two features are listed in Table 5.6 and Table 5.7.

When including the above two features, the Random Forest achieved the following accuracies during both turning-movement distributions, i.e. *[0.5, 0.5]* and *[0.2, 0.8]*, respectively $IDM_{acc} = 0.81 \textit{ and } 0.89$ and $Krauss_{acc} = 0.51 \textit{ and } 0.80$. It can be concluded that the accuracies of the Random Forests remain almost identical when removing these acceleration based features. We can thus observe that our predictive models are not aided much by the proposed acceleration based features.

In conclusion, it is strongly advised to remove the acceleration based features since they (1) do not increase the predictive accuracy too much and (2) are difficult and noisy to obtain in reality.

The above two features are still used by LCP in the next chapter where we evaluate the Traffic Flow Improvement of SCHIC with LCP. These two features are still in use, since this test without them has been performed afterwards.

Additionally, Appendix D shows the importance of each individual feature in a graphical manner, by plotting the individual importance of each feature. There, we also remove the derived features, Speed Change and Relative Speed to a leading vehicle.

Turning Movement distribution	Fixed label	Probability	SVM	Decision Tree	Random Forest
$[50, 50]$	0.50	0.51	0.57	0.78	0.79 (before: 0.81)
$[20, 80]$	0.80	0.67	0.80	0.89	0.89 (before: 0.89)

Table 5.6: Accuracies for the IDM driver model, without acceleration based features.

Turning Movement distribution	Fixed label	Probability	SVM	Decision Tree	Random Forest
$[50, 50]$	0.49	0.50	0.50	0.53	0.49 (before: 0.51)
$[20, 80]$	0.80	0.67	0.80	0.79	0.80 (before: 0.80)

Table 5.7: Accuracies for the Krauss driver model, without acceleration based features.

5.4 Conclusion

This chapter covered the training procedure of classification models for Lane Change Prediction (LCP). These classification models are trained on two commonly used driver-models within the traffic simulator SUMO, i.e. deterministic IDM and the stochastic Krauss driver-model. Two scenarios are explained and each uses a single driver model, *Scenario A: IDM* and *Scenario B: Krauss driver-model.* For each of these two driver-models two popular turning-movement distributions are considered which are either balanced or un-balanced, i.e. the ratio of *[left-turning, straight-ahead]* is either *[0.5, 0.5]* or *[0.2, 0.8]*.

The most accurate predictive models are the Decision Tree and Random Forest, which achieved comparable accuracies. The predictive accuracies of the Random Forest during both turning-movement distributions *[0.5, 0.5]* and *[0.2, 0.8]*, are respectively $Krauss_{acc} = 0.51$ *and* 0.80, $IDM_{acc} = 0.81$ *and* 0.89. Given the predictive accuracy on the Krauss driver-model, it is expected that no learning is achieved based on the anticipative driving style. Since the Random Forest has simply reverted to a fixed label prediction in the unbalanced [20,80] scenario. As concluded with learning curves, an accurate Lane Change Prediction (LCP) based on the anticipative driving style is only achieved for the deterministic Intelligent Driver Model (IDM). The Random Forest is chosen, since it is expected to generalize better in a real-world scenario where different driving styles are actually expected to occur.

In the next chapter, the performance of SCHIC is evaluated by applying the predictions of LCP to SCHICs scheduling algorithm for both *Scenario A: IDM* and *Scenario B: Krauss driver-model.* Although the Intelligent Driver Model (IDM) is deterministic this does enable a clear reasoning on the results of SCHIC, since stochastic behaviour is not present in these IDM simulations.

Chapter 6

Traffic Flow Improvement with LCP

In the preceding chapters we described how Lane Change Prediction (LCP) can be formalized as a classification task. We performed simulations in SUMO to generate training-data, which was used to learn a Lane Change Predictor. We evaluated the performance of LCP for two turning-movement distributions at the intersection, i.e. the ratio of *[left-turning, straight-ahead]* is either *[0.5, 0.5]* or *[0.2, 0.8]*. In this chapter, we again evaluate two driver-models, the deterministic Intelligent Driver Model (IDM) and stochastic Krauss driver-model.

We can accurately predict lane changes for IDM, since the predictive accuracies during both turning-movement distributions of *[0.5, 0.5]* and *[0.2, 0.8]* are $IDM_{acc} = 0.81$ *and* 0.89, respectively. However, with the Krauss driver-model the predictive accuracies are just $Krauss_{acc} = 0.51$ *and* 0.80 which only match the bias as found in the turning-movement distribution.

However, ultimately we are interested in evaluating if these Lane Change Predictions can increase SCHICs performance. We consider an intersection with commonly found short entry-lanes. Here, LCP is compared to SCHICs baseline methods of *fixed-label and split-vehicles*. Next to these methods, which only estimate or predict lane changes, we apply perfect predictions by means of an *Oracle*.

This chapter continues by briefly explaining the chosen performance metrics in Section 6.1. Afterwards, we state all parameter-settings of the Time-Gap Controller to which SCHIC is compared as well in Section 6.2. Section 6.3 presents the results of the aforementioned *Scenario A: IDM*, where each driver is controlled by the Intelligent Driver Model (IDM). Section 6.4 presents the results of *Scenario B: Krauss driver-model*, where all drivers are now controlled by the Krauss driver-model. Thereafter, this chapter concludes in Section 6.5.

6.1 Performance metrics

The criteria on which this evaluation focusses are threefold. Namely, during each hourly vehicle demand we evaluate the following three metrics which are then averaged over all vehicles of this hourly vehicle demand: **(1)** average speed **(2)** average number of stops **(3)** average waiting time. In this section, these three metrics are briefly explained and related to SCHICs optimization-function.

Recall that SCHICs optimization-function aims to minimize the cumulative delay for all observed vehicles. In this simulation, all vehicles drive a route which is identical in length. Therefore, a minimization of cumulative delay will be reflected in a higher average speed and a reduced average waiting time for all vehicles in simulation.

One could argue that the most intuitive (or visible) result of LCP and the baseline methods is to prevent that an approaching vehicle will need to stop at the intersection while it wants to join the currently active green-phase. In order to evaluate this intuitive result, the average number of stops per vehicle is measured. Lastly, one should note that reducing the average number of stops is an indirect result of LCP given SCHICs optimization-function. As stated before, SCHICs optimization-function actually only optimizes the cumulative delay for all observed vehicles, i.e. it does not optimize for the number of stops directly.

6.2 Parameter settings of the Time-Gap Controller

SCHIC is compared to a *Time-Gap Controller*, which working principle is based on time-gaps as observed between subsequently arriving vehicles (Section 2.1). Recall, that this Time-Gap Controller extends the current green phase if each consecutive vehicle arrives at its associated detector (specified by $detector_{gap} = 2.2s$) within a threshold value of $max_{gap} = 3.0s$. This Time-Gap Controller uses the same phase set as SCHIC with the same minimum and maximum phase durations, respectively $[5s,\ 55s]$.

The implementation of SUMOs time-gap control is used, unfortunately only a single value can be specified for the location of all advance detectors. Therefore, the comparison to SCHIC is not completely fair. For this Time-Gap Controller the advance detectors are placed on each lane as limited by the shortest lanes. These shortest lanes are thus found after a lane-split, and the advance detectors are place here at the maximum allowed distance from the stop-line, i.e. $2.2s$ or $30m$ from the stop-line. This value of $2.2s$ is also used by SCHIC on these shortest lanes when it does not use any predictions for lane changes. However, on the longer and vertically oriented lanes without a lane-split SCHIC already detects vehicles at $3.0s$, as explained in Section 5.1 and listed in Table 5.1.

The default settings of SUMOs Time-Gap Controller are: $max_{gap} = 3.0s$ and $detector_{gap} = 2.0s$. Therefore, the used scenario only differs ever so slightly since the detector-gap increased to $detector_{gap} = 2.2s$. In conclusion, the comparison between SCHIC and the Time-Gap Controller is not completely fair given the differences in look-ahead time for vehicle detection.

6.3 Scenario A: Intelligent Driver Model (IDM)

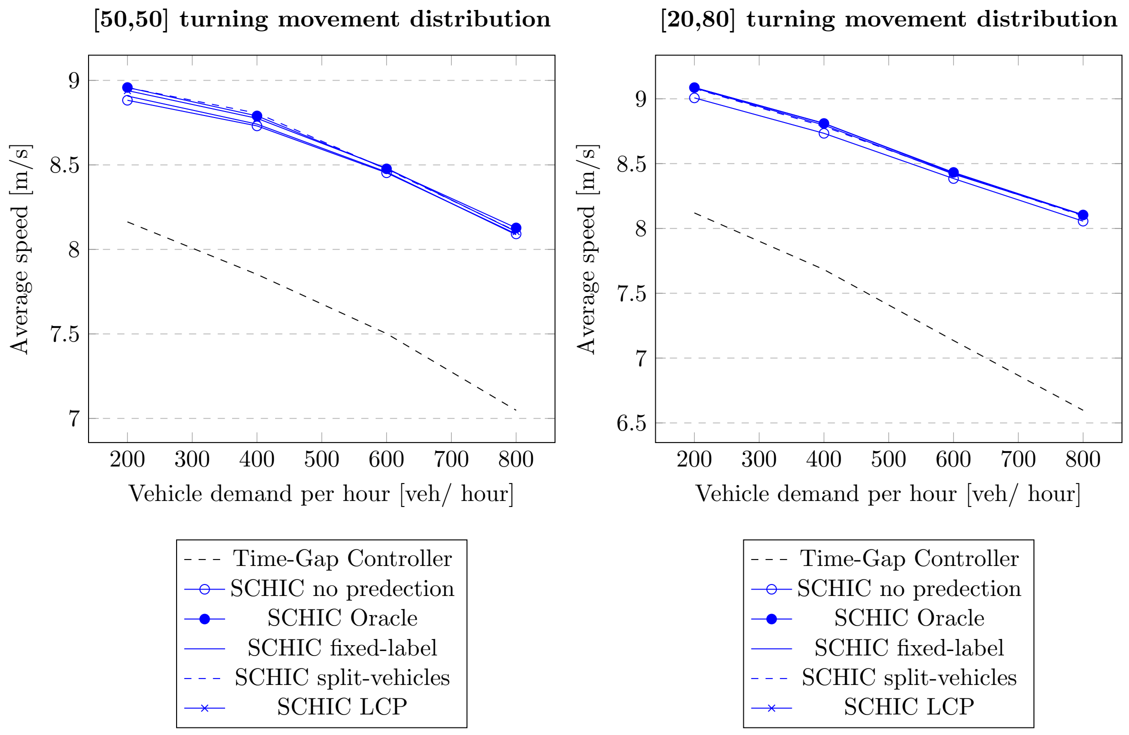


Figure 6.1: Average speed per vehicle in the hold-out test-set with 15 hours of simulation per hourly vehicle demand. Scenario A: Intelligent Driver Model (IDM).

(1) Average speed: Figure 6.1 contains the results of all six control algorithms and plots the average speed of all vehicles in simulation.

The Time-Gap Controller (dashed, black) clearly performs worse than all SCHIC methods. Possibly due to the fact that it only uses a look-ahead time of 2.2s on the (vertical) full length lanes, whereas SCHIC uses 3.0s on these lanes. Or, simply due to the simpler logic. Recall that the Time-Gap Controller terminates each green-phase after it observes a threshold duration of inactivity on the currently serviced lane, $th_{gap} = 3.0s$. Whereas SCHIC terminates a green-phase directly after the last observable vehicle has left, so potentially SCHIC services the other (waiting) vehicles sooner.

Moving on to SCHICs control now, SCHIC Oracle (blue, filled circle) shows slightly better performance when compared to fixed-label, split-vehicles and LCP. SCHIC Oracle's performance increase over SCHIC no prediction (blue, empty circle) is compared across all hourly vehicle demands within the [50,50] and [20,80] turning movement distributions, the average speed increases with 0.56% and 0.73% respectively.

However, split-vehicles (blue, dashed) outperforms the Oracle ever so slightly during the vehicle demand of 400 (veh/hour) in the balanced [50,50] turning movement distribution. A possible reason is given here, by stating the conceptual difference between both approaches. The Oracle terminates or extends each green-phase based on the correct upcoming lane change of a single detected vehicle. Split-vehicles divides the detected vehicle in two split-vehicles, and each split-vehicle corresponds to a single green-phase. Thus, split-vehicles, acts less strict since it actually considers the two potential green-phases that this vehicle can request. Therefore, SCHIC computes a schedule and cumulative delay based on two split-vehicles which each correspond to a potential green-phase. A specific scenario in which this conceptual difference might change the results is described below in ◇ *Thought experiment 1.*

Overall, it is clear that the baseline methods *(Oracle, fixed-label, split-vehicles)* and *LCP* (blue, $*$) perform very similar. However, the performance of these baseline methods and LCP can't be properly compared here, since the worse performing Time-Gap Controller is plotted as well which reduces the scale of the graph. Therefore, Figure 6.3 omits the Time-Gap Controller but visualizes the same metric of average speed in a bar graph only for all SCHIC controllers.

◇ **Thought experiment 1**, extension dilemma for closely arriving vehicles.

In this thought experiment, *Vehicle A, B and C* are considered which can approach the intersection on the same road as well as on opposing roads. See Figure 6.2.

Current time: 10.0 sec (Figure 6.2a)

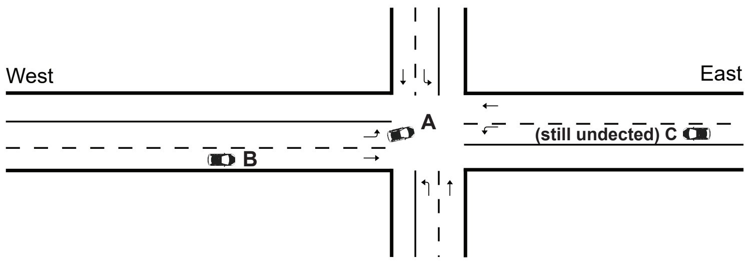
Consider that vehicle B and C will each request a different green-phase. The first vehicle to arrive at the intersection is {*vehicle B; phase: 0*} and a brief moment later {*vehicle C; phase: 1*} arrives. Consider that currently green-phase 1 is active, but could be terminated in the next time-step, since the last vehicle (vehicle A) for this green-phase has now left its entry-lane. When the Oracle detects the firstly arriving *vehicle B* at its detection-loop, it will communicate to SCHIC that this vehicle will not request the currently active green-phase.

Current time: 10.5 sec (Figure 6.2b)

Therefore, SCHIC Oracle will now terminate the current green-phase and select the next green-phase (green-phase 2), since all observed vehicles for the current green-phase have left their entry-lane. However, split-vehicles might allow SCHIC to prolong the current green-phase a bit longer when *vehicle B* is detected. Prolonging the current green phase is in favour of *vehicle C* which arrives closely after *vehicle B.* Recall that *vehicle B* had to stop at the intersection regardless. Since SCHIC does not skip green-phases and *vehicle B's* requested green-phase (green-phase 0) will only be active after green-phases 1,2 and 3 are have at least run for a minimum duration of 5 *sec.*

To conclude, this specific scenario might occur quite often in the the balanced [50,50] turning movement distribution. Hence, split-vehicles might have a conceptual advantage here since it allows the current green-phase to be extended for a brief moment in which subsequently arriving vehicles can join this extended green-phase.

(Note that *Vehicle A* is not required in this scenario, since each phase needs to run for a minimum duration of 5 *sec.*)

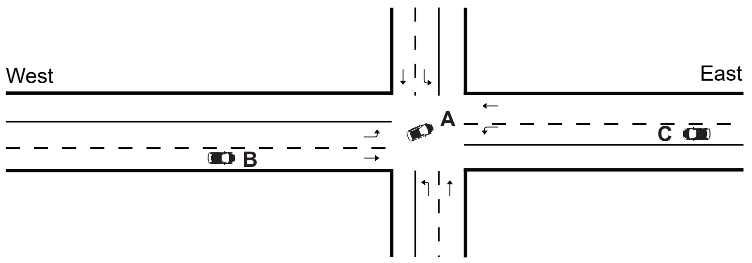


Time: 10.0 sec
Phase: 1 (for left-turning traffic)

Detected vehicles by Oracle: {vehicle: B, phase: 0, arrival: 13.0 sec}

Detected vehicles by split-vehicles: {vehicle: B^0, phase: 0, arrival: 13.0 sec}
{vehicle: B^1, phase: 1, arrival: 13.0 sec}

(a) Time-step 10.0 sec, vehicle C is still undetected.



Time: 10.5 sec
Phase: 2 (for straight-ahead traffic, vertical lanes) is selected by the Oracle.
Phase: 1 (for left-turning traffic) is likely extended by split-vehicles.

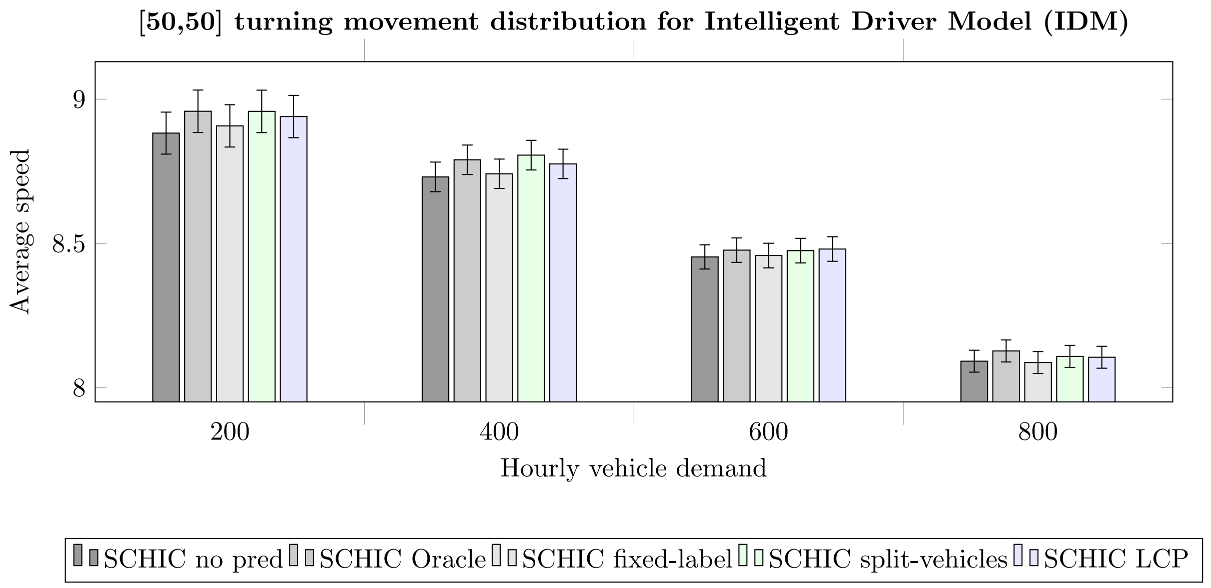
Split-vehicles could thus allow vehicle C to join the extended phase 1.

Detected vehicles by Oracle: {vehicle: B, phase: 0, arrival: 13.0 sec}
{vehicle: C, phase: 1, arrival: 13.5 sec}

Detected vehicles by split-vehicles: {vehicle: B, phase: 0, arrival: 13.0 sec}
{vehicle: C^0, phase: 0, arrival: 13.5 sec}
{vehicle: C^1, phase: 1, arrival: 13.5 sec}

(b) Time-step 10.5 sec, the Oracle switched to the next green-phase. While split-vehicles likely extended the previous green-phase, allowing vehicle C to join.

Figure 6.2: Thought experiment 1 for the Oracle and split-vehicles, showing time-steps 10.0 sec and 10.5 sec.



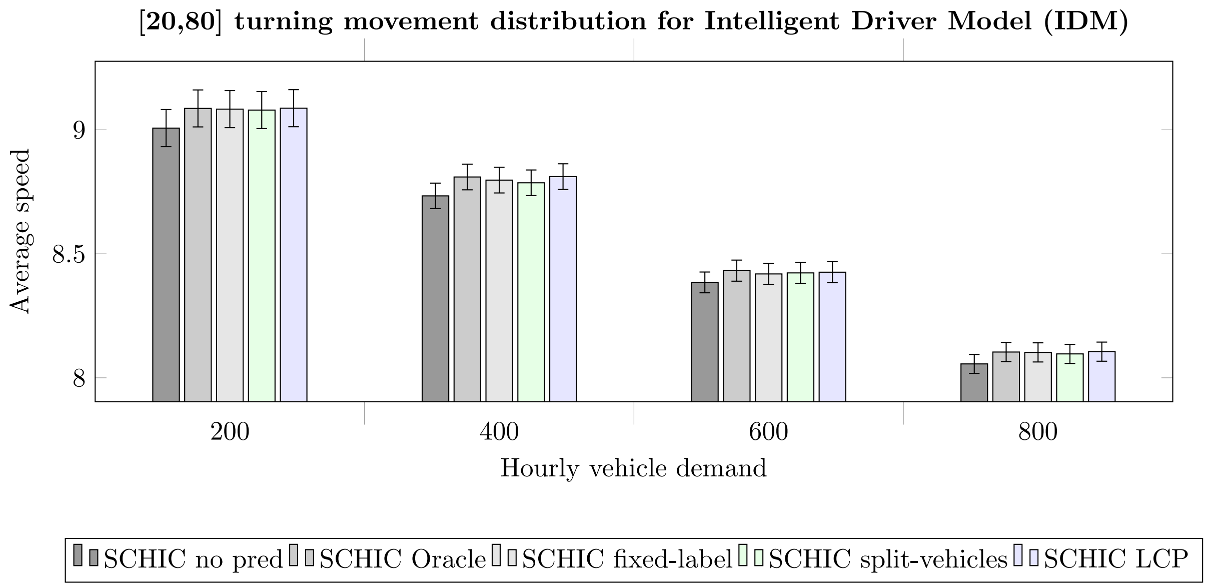


Figure 6.3: Average speed per vehicle in the hold-out test-set with 15 hours of simulation per hourly vehicle demand. Error-bars are 95% confidence intervals. Scenario A: Intelligent Driver Model (IDM).

(1) Average speed: the previous observations remain clear. However, for a comparison of SCHIC and its baseline methods, Figure 6.3 is used. The bar on the left shows SCHIC without any predictions on the lanes that contain a lane-split (SCHIC no pred). So the bars, moving from left to right, correspond to the methods of: *no pred, Oracle, fixed-label, split-vehicles and LCP.*

[50,50] turning movement distribution

In the [50,50] case, *fixed-label* shows the poorest performance across the board when compared to *split-vehicles* and *LCP*. Which can be expected given its low accuracy of 50% compared to an 81% accuracy of *LCP*. Since both *LCP* and *fixed-label* perform classification, the differences in accuracy are most likely leading to the performance increase of *LCP* over *fixed-label*.

In the same [50,50] case, *split-vehicles* and *LCP* are now compared. It is observed that *split-vehicles* outperforms *LCP* in the two lowest vehicle demands. However, *LCP* slightly outperforms *split-vehicles* during the vehicle demand of 600 (veh/hour). Both methods achieve almost identical performance in the highest vehicle demand of 800 (veh/hour). *Split-vehicles* does not perform a classification of vehicles, but instead splits each vehicle weighted by the observed balanced turning-movement distribution of *[left-turn = 0.5, straight-ahead = 0.5]*. Therefore, the 81% classification accuracy of *LCP* can't be used directly to explain performance differences of *split-vehicles* and *LCP*. However, by scheduling a *split-vehicle* for each green-phase, *split-vehicles* might cover up for its lower accuracy in this balanced scenario as described prior in ◇ Thought experiment 1. In conclusion, *split-vehicles* has a slight edge over *LCP* during the lower two vehicle demands but their performance converges to each other during the higher two vehicle demands.

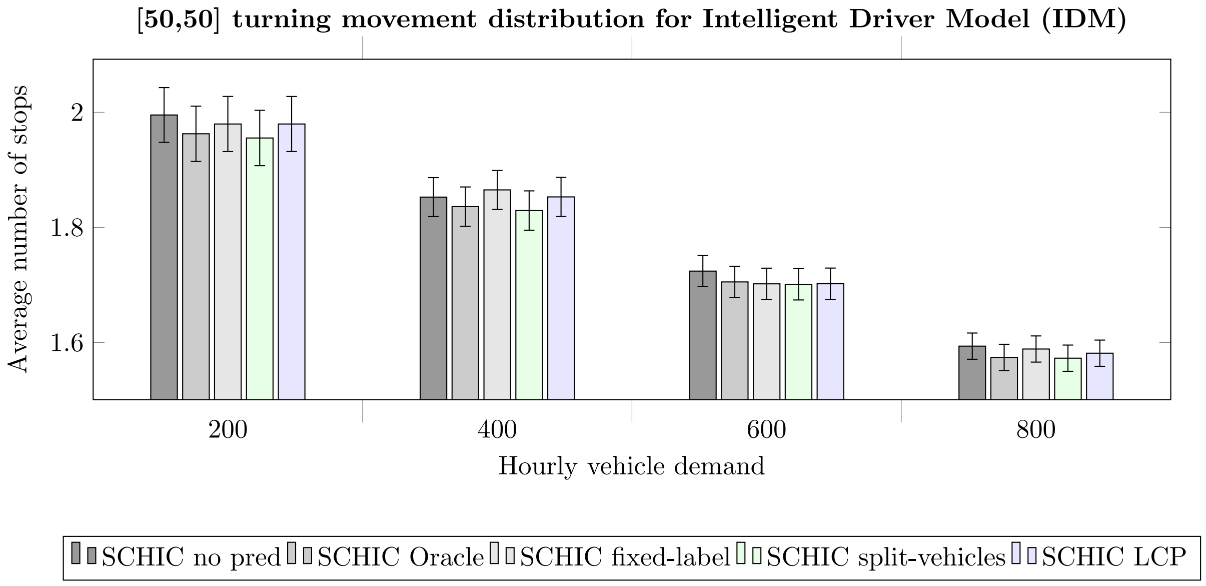
[20,80] turning movement distribution

Lastly, the [20,80] case can be concluded in a more concise manner. The performance of *fixed-label* increases as is expected, since its accuracy rose from 50% to 80%. However, the accuracy of *LCP* rose to 89% and given the fact that both methods are based on classification, *LCP* outperforms *fixed-label*. *LCP* also ever so slightly outperforms *split-vehicles*. Although an accuracy metric is not available for *split-vehicles*, *LCP*s accuracy of 89% is well above the bias found in the [20,80] case which is used by *split-vehicles*.

At 400 vehicles per hour, the *Oracle* is insignificantly outperformed by LCP with a margin of 0.0015 m/s. However, for both the *Oracle* and *LCP* this is well within the 95% confidence intervals of $\pm$ 0.052 m/s. With the deterministic Intelligent Driver Model (IDM), the arrival time of each vehicle at the intersection is identical each time one of the 15 individual hourly simulations is executed. To clarify, the vehicle release times for each *each (of the 15)* individual hourly simulation are randomized a priori with 15 pre-determined seeds. The confidence intervals are however present since *each (of the 15)* individual hourly simulations contain different arrival times at the intersection.

The fact that LCP shows an insignificantly higher *average* speed could be explained by 'lucky' miss-predictions. Each miss-prediction of LCP will increase the duration of the currently active green-phase, this might allow a currently undetected but closely arriving vehicle to join the extended green-phase. Hence, the reference to a 'lucky' miss-prediction by *LCP*. The difference of 0.0015 m/s between the *Oracle* and *LCP* can be considered as insignificant given their 95% confidence intervals of $\pm$ 0.052 m/s. Therefore, the reasoning above should suffice as a potential explanation.

In conclusion, in the unbalanced [20,80] case the *Oracle, fixed-label, split-vehicles and LCP* show a small and comparable performance potential for SCHICs traffic flow.



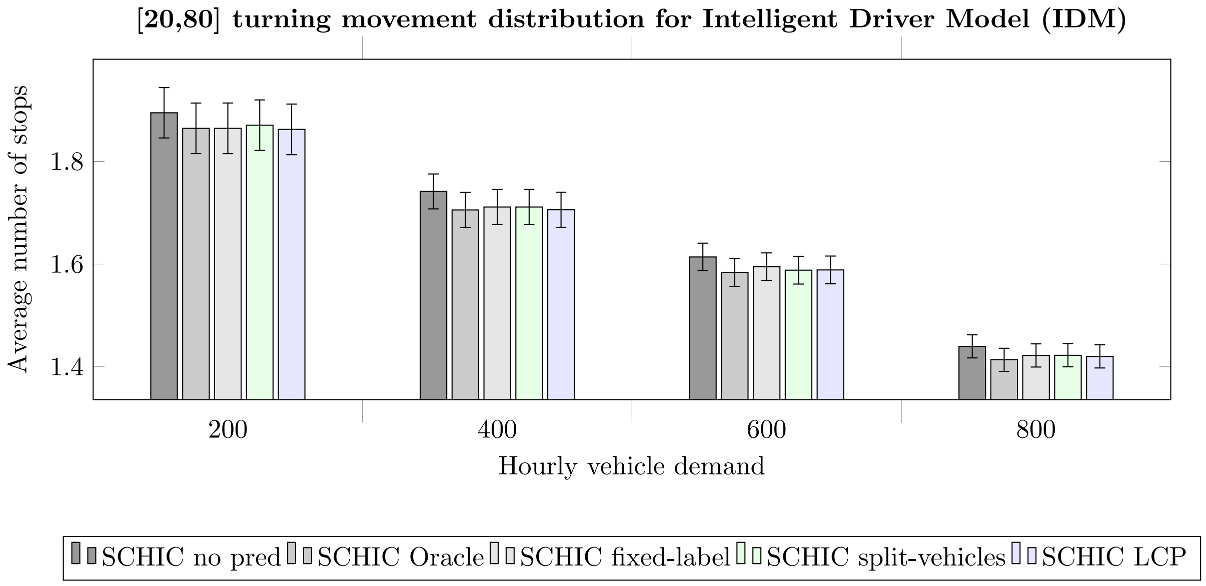


Figure 6.4: Average number of stops per vehicle in the hold-out test-set with 15 hours of simulation per hourly vehicle demand. Error-bars are 95% confidence intervals. Scenario A: Intelligent Driver Model (IDM).

(2) Average number of stops: One could argue that the most visible result of *LCP* and the baseline methods is to prevent that an approaching vehicle will need to stop at the intersection while it wants to join the currently active green-phase. In order to validate this visible result, the average number of stops per vehicle is shown in Figure 6.4.

Overall, the *Oracle* reduces the number of stops compared to *SCHIC no prediction* for both turning-movement distributions of [50,50] and [20,80] with 1.21% and 1.84% respectively.

[50,50] turning movement distribution
As expected in the [50,50] case, *fixed-label* performs poorly given its low accuracy of 50% and causes costly additional stops compared to the *Oracle*. Again, *fixed-label* performs poorest when compared to the other two methods of *split-vehicles* and *LCP*. Additionally, in this balanced [50,50] case with 400 (veh/hour) *fixed-label* even causes more vehicle stops than *SCHIC no pred*. This might be explained with *thought experiment 2* as shown below, however, this is not supported by additional measurements.

At 600 (veh/hour), *fixed-label, split-vehicles and LCP* ever so slightly outperform the *Oracle*. Again, these differences might be explained by 'lucky' miss-predictions of *fixed-label and LCP* as explained before. During all hourly vehicle demands, *split-vehicles* outperforms the *Oracle* ever so slightly which could again be explained by ⋄ *Thought experiment 1*. Furthermore, for all three methods, the differences are well within the 95% confidence intervals.

In the same [50,50] case, *split-vehicles* and *LCP* are now compared. Again, it is observed that *split-vehicles* outperforms *LCP* during some demands, namely [200, 400, 800] (veh/hour). However, these methods perform almost similar during the vehicle demand of 600 (veh/hour). Again, it can be reasoned that the high accuracy of LCP (accuracy = 81%) is counteracted by its strictness. *Split-vehicles* is challenged by the balanced turning movement distribution, on which it bases the split-ratio for each predicted vehicle. However, it seems to be aided by its fuzzy control approach since it expects two vehicles to arrive instead of just one as with *LCP*.

[20,80] turning movement distribution
Lastly the [20,80] case can be concluded in a more concise manner. The *Oracle* is insignificantly outperformed by *LCP* during the demand of 200 vehicles per hour. This marginal difference might again be explained by 'lucky' miss-predictions and is well within the confidence intervals. The performance of *fixed-label* improved, as is expected since its accuracy rose from 50% to 80%. *Fixed-label* (accuracy = 80%) and *LCP* (accuracy = 89%) are identical approaches which only differ in their classification accuracy, the accuracy of *LCP* surpassed that of *fixed-label* and slightly outperformed *fixed-label* during all hourly demands.

In general, a slight increase in the average number of stops on the vertical lanes might enable a higher average speed for vehicles on the horizontal lanes. The different methods are likely to increase the duration of a green-phase for traffic on the horizontal lanes, so new vehicles can still join this green-phase. Therefore, the number of stops on the vertical lanes is likely to increase since their green-phase is now delayed by a small amount of time, this is in favour of vehicles on the horizontal lanes. However, the average speed of all vehicles could be higher since the additional stops are only brief and the average speed is measured on the entire route of each vehicle.

In conclusion, in the unbalanced [20,80] case all three methods have almost converged to the performance of the *Oracle*.

◇ **Thought experiment 2, extension dilemma with miss-classification-burst.**

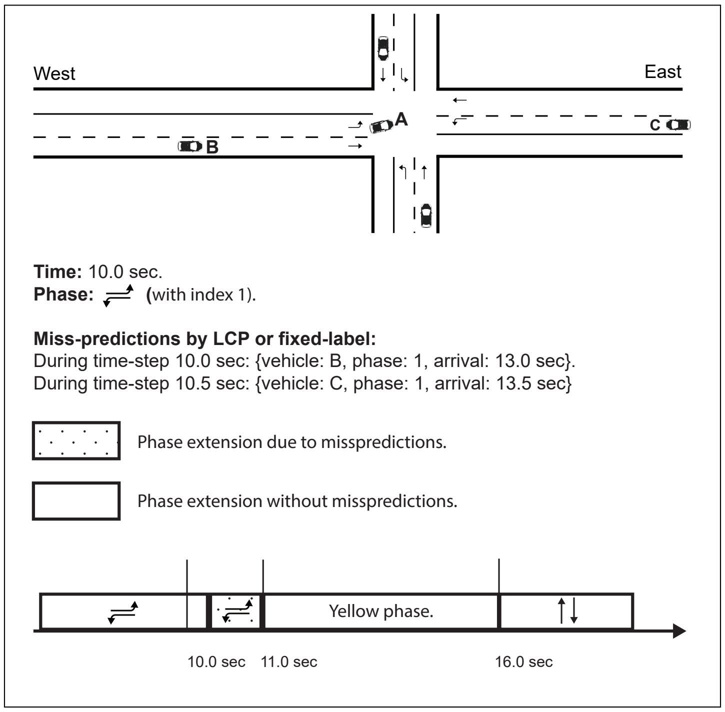
See Figure 6.5.

Consider that the green-phase for left-turning traffic on the horizontal lanes is currently active. But this green-phase is about to be terminated since the last vehicle for this green-phase is about to leave its entry-lane. *Fixed-label* performs classifications for new vehicles which could join this current green-phase. A miss-classification in this scenario, is predicting that a new vehicle will join the currently active green-phase while it will actually request the different green-phase for straight-ahead traffic.

A miss-classification in this scenario, will therefore extend the duration of the current left-turning green-phase on the horizontal lanes. Even when all vehicles on the horizontal lanes request a different green-phase for continuing straight-ahead.

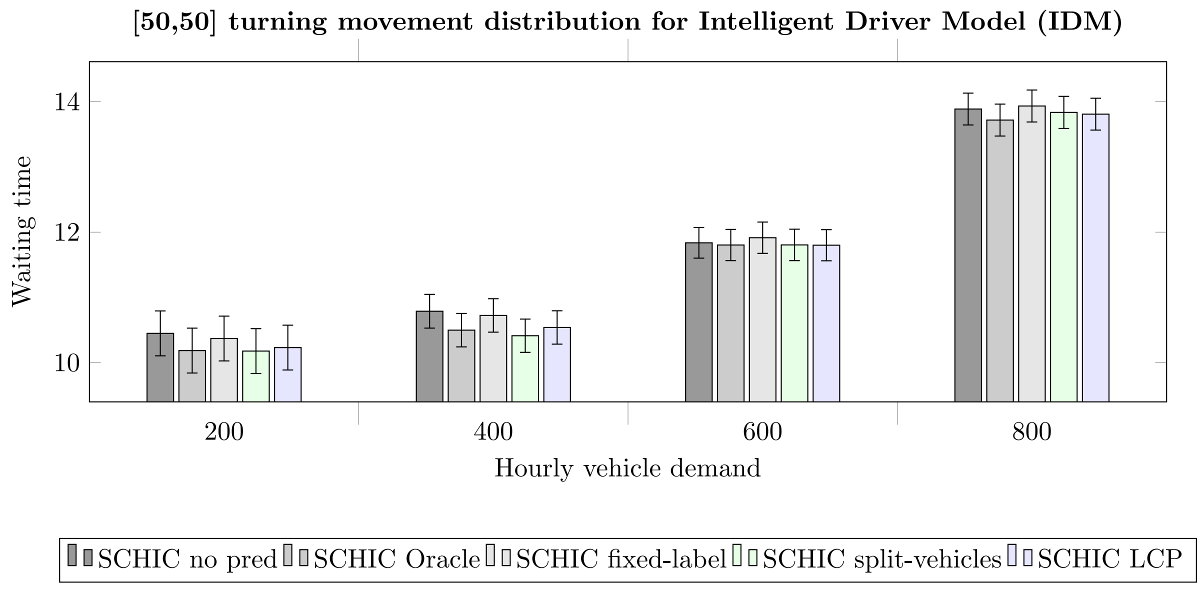
This extended duration can cause two problems: (1) it can cause the queues of "vertical" traffic to wait longer (2) can cause additional stops for new "vertical" vehicles. Without a burst of miss-classifications, "vertical" vehicles would have observed their green signal sooner, and could thus have joined an accelerating queue instead of joining this still standing queue which causes additional stops.

In conclusion, *fixed-label* operates at two lane-splits, when enough miss-predictions occur in a row at either of these two lane-splits, it could increase the number of stops on the vertical lanes.



Miss-predictions by LCP or fixed-label:
During time-step 10.0 sec: {vehicle: B, phase: 1, arrival: 13.0 sec}.
During time-step 10.5 sec: {vehicle: C, phase: 1, arrival: 13.5 sec}

Figure 6.5: Two mis-classifications extend the current green-phase, causing an unnecessary delay for the two vertical vehicles.



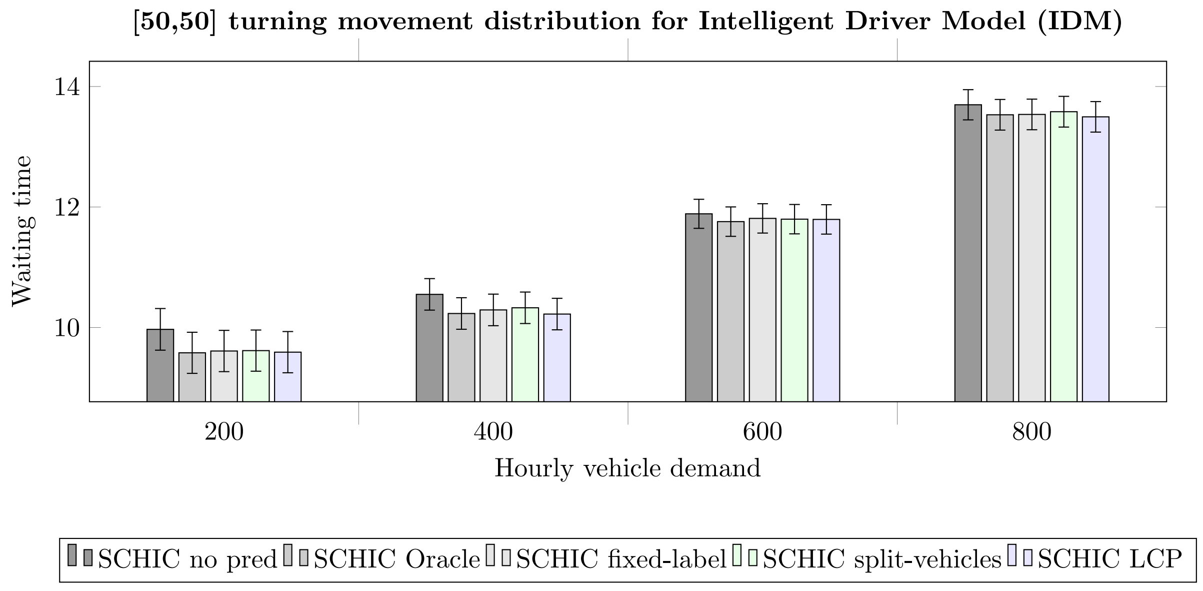


Figure 6.6: Average waiting time per vehicle in the hold-out test-set with 15 hours of simulation per hourly vehicle demand. Waiting time is defined as the total time that a vehicle has a speed below 0.1 m/s. Error-bars are 95% confidence intervals. Scenario A: Intelligent Driver Model (IDM).

(3) Average waiting time: lastly, one could also argue that the average number of stops is only an important result of *LCP* since it is a directly observable result. To clarify, SCHICs cost-function only optimizes the cumulative delay of all observed vehicles, i.e. it does not optimize for the number of stops directly.

Therefore, Figure 6.6 evaluates the average waiting time per vehicle. Waiting time is defined as the total time that a vehicle has a speed below 0.1 m/s.

When observing the average waiting time, during both the [50,50] and [20,80] turning movement distributions, the same overall trend is seen as for the average speed metric. When comparing the *Oracle* to SCHIC no prediction, the performance potential of the *Oracle* decreases slightly when the vehicle demand increases. It could be argued that optimizing the traffic flow is increasingly difficult when the vehicle demand increases.

[50,50] turning movement distribution

The *Oracle* shows a slightly higher average waiting time than *split-vehicles* at 200, 400 (veh/hour) and LCP at 600 (veh/hour). However, the differences are both negligible and well within the 95% confidence intervals.

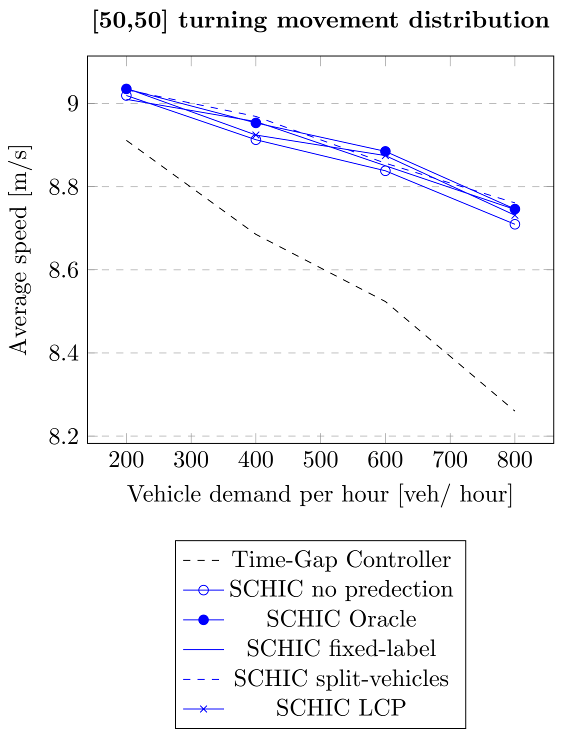
During the balanced [50,50] scenario at 600 and 800 vehicles per hour, *fixed-label* shows the poorest performance of all methods. This is most likely due to *fixed-labels* low 50% classification accuracy. Since its performance relative to the other methods increases in the [20,80] scenario, where its classification accuracy rises to 80%.

[20,80] turning movement distribution

At 400 and 800 (veh/hour) the *Oracle* is insignificantly outperformed by LCP, which is again well within the confidence intervals and might be explained by 'lucky' miss-predictions of LCP.

Especially in the unbalanced [20,80] turning movement distribution, the *Oracle, fixed-label, split-vehicles and LCP* show a comparable performance potential for SCHIC.

6.4 Scenario B: Krauss driver-model



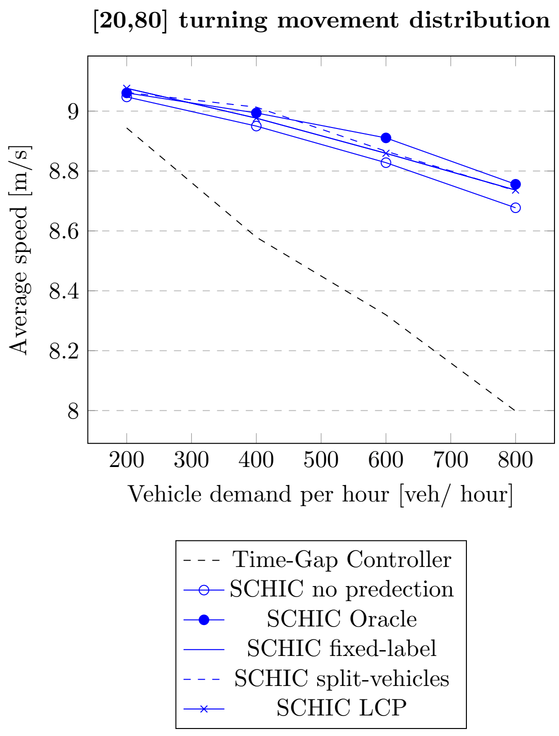


Figure 6.7: Average speed per vehicle in the hold-out test-set with 15 hours of simulation per hourly vehicle demand. Scenario B: Krauss driver-model.

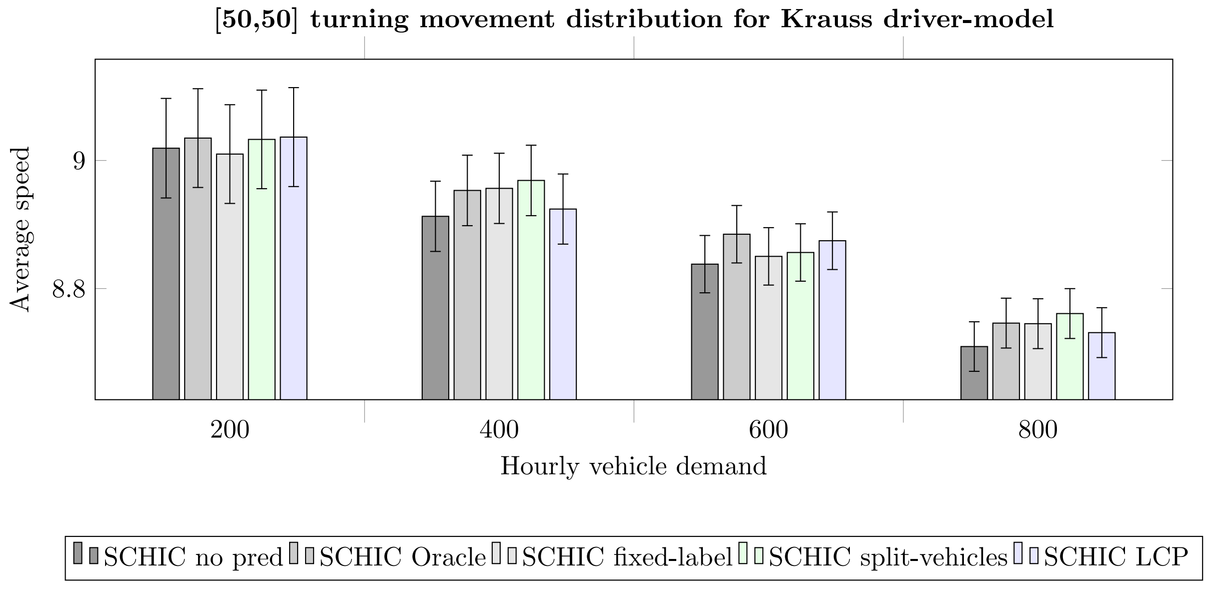
(1) Average speed: Figure 6.7 contains the results of all six control algorithms under the same vehicle release times as before, however all vehicle dynamics are now determined by the Krauss driver-model.

As in the previous IDM-scenario, the Time-Gap Controller (dashed, black) clearly performs worse than all SCHIC methods.

In general, the performance potential of the Oracle (blue, filled circle) over SCHIC no prediction (blue, unfilled circle) averaged over all hourly vehicle demands is 0.40% in the balanced [50,50] scenario and 0.62% in the unbalanced [20,80] scenario. These performance improvements are slightly lower than those obtained in the previous IDM scenario, but remain comparable.

Given the stochastic properties which are applied on a vehicles' speed by the Krauss driver-model, the average speed might fluctuate a bit during its entire $400m$ route. Recall that the average speed is measured for a vehicle when it approaches and exits the intersection, so over a total distance of $400m$. A fluctuation in the average speed is observed in the lower vehicle demands up to 400 $[veh/hour]$, since the Oracle's performance shows a slight drop here.

The performance of the SCHIC controllers are further evaluated on the next pages.



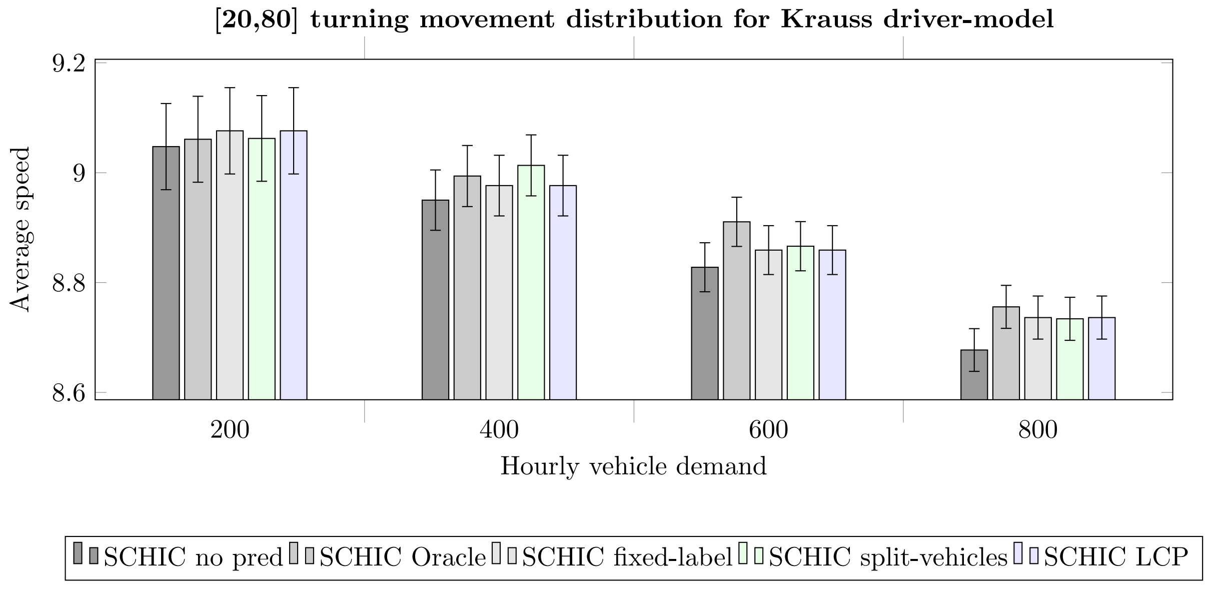


Figure 6.8: Average speed per vehicle in the hold-out test-set with 15 hours of simulation per hourly vehicle demand. Error-bars are 95% confidence intervals. Scenario B: Krauss driver-model.

Figure 6.8 shows the average speed as obtained with all SCHIC controllers, the worse performing *Time-Gap Controller* is removed, in order to compare the SCHIC Controllers more clearly.

The 95% confidence intervals for these stochastic Krauss vehicles are considerably larger than with the deterministic IDM vehicles. This is probably due to the previously mentioned stochastic properties which are applied on the speed of a Krauss vehicle during its entire route of $400m$.

[50,50] turning movement distribution

In the [50,50] scenario, the Oracle performs worse than *split-vehicles* during the demands of 400 and 800 [veh/hour], with a margin of 0.0150 m/s *to* 0.0156 m/s. However, these margins remain well within the confidence intervals of ± 0.0550 m/s and ± 0.0389 m/s for both vehicle demands. The same trend is seen for *LCP* (at 200 [veh/hour]) and *fixed-label* (at 400 [veh/hour]), which outperform the *Oracle* with a negligible margin with respect to their confidence intervals.

These negligible differences are likely a result of (1) the limited performance potential of all methods, combined with, (2) the stochastic properties in the speed profiles of the Krauss driver-model. The small differences could be appointed to statistical insignificance on this performance metric.

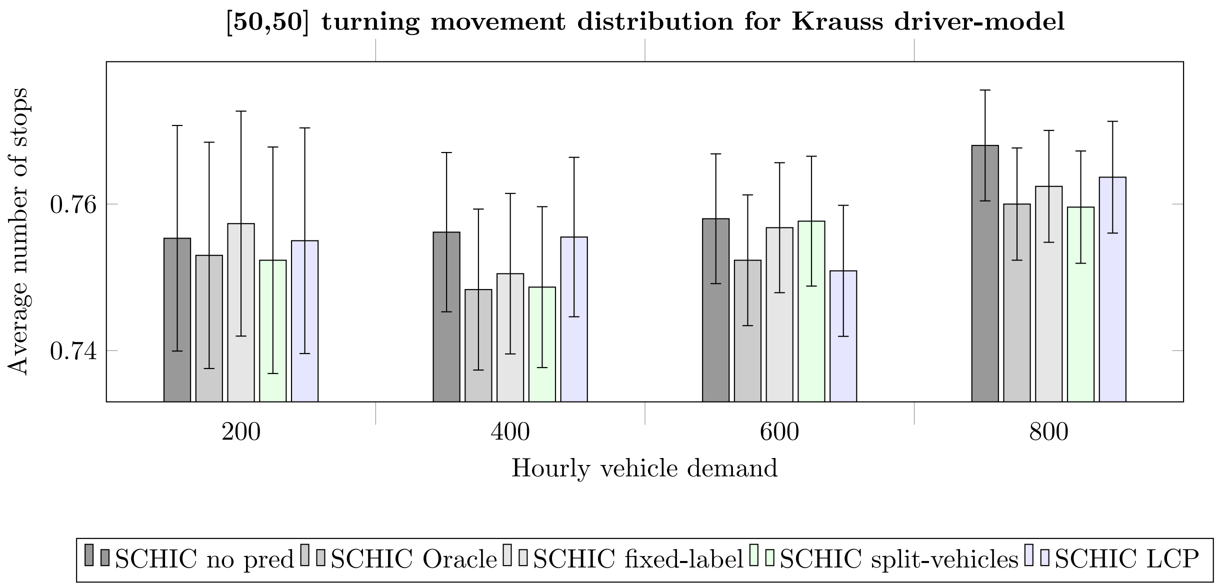
As before, *split-vehicles* might have a conceptual advantage in this balanced scenario thanks to its fuzzy approach of splitting vehicles and therefore allowing SCHIC to schedule two green-phases for this single vehicle. In conclusion, *split-vehicles* performs quite well in this balanced scenario.

A second general note considers the low accuracies of both *fixed-label* and *LCP*, their accuracies are respectively just 49% *and* 51% during the balanced [50,50] turning movement distribution. The increase of the average speed under these methods fluctuates a bit, e.g. during the different vehicle demands *fixed-label* and *LCP* win and lose from each other in an alternating order. The fluctuation in their performance potential is likely due their low classification accuracies, which likely causes quite some miss-predictions.

[20,80] turning movement distribution

Lastly, the [20,80] scenarios are observed. Again, the Oracle performs poorly in the two lower vehicle demands, however its performance is superior in the higher two vehicle demands. When comparing *fixed-label, split-vehicles and LCP*, there is a slight fluctuation of performance observed among them.

Overall, it can be concluded that *split-vehicles* performs best in the balanced [50,50] case. Lastly, in the unbalanced [20,80] scenario *fixed-label, split-vehicles and LCP* perform quite comparable.



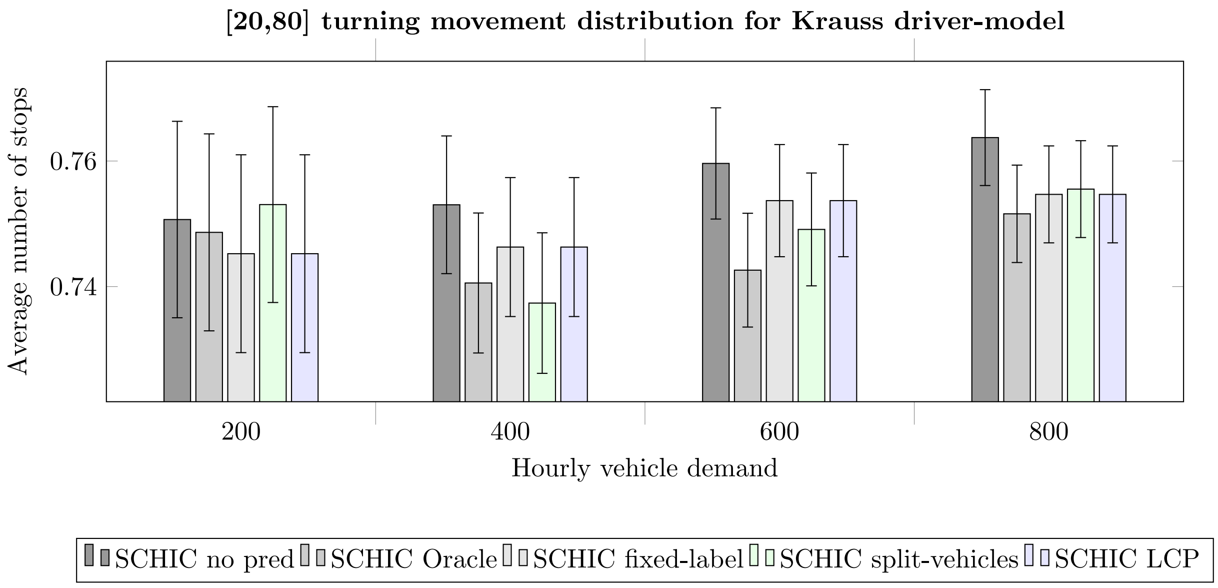


Figure 6.9: Average number of stops per vehicle in the hold-out test-set with 15 hours of simulation per hourly vehicle demand. Error-bars are 95% confidence intervals. Scenario B: Krauss driver-model.

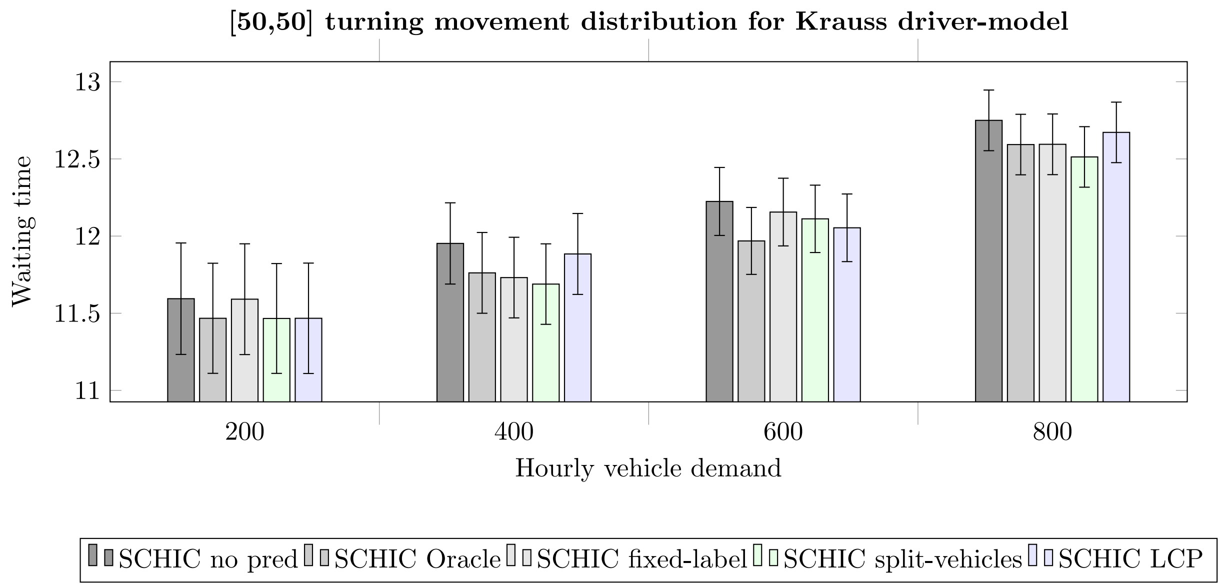
(2) Average number of stops: Figure 6.9 shows the average number of stops per vehicle for both the balanced [50,50] scenario and the unbalanced [20,80] scenario.

On average, the *Oracle* reduces the number of stops compared to SCHIC no prediction for both turning-movement distributions of [50,50] and [20,80] with 0.79% and 1.44% respectively.

During the different hourly demands in the [50,50] and [20,80] scenarios, the performance potential of the *Oracle* over *SCHIC no prediction* varies between between $\{0.27\% \textit{ and } 2.24\%\}$.

The relative performance potential of the *Oracle* as expressed in percentages, is comparable for both the Krauss and IDM scenario. However, the average number of stops for Krauss vehicles are roughly one entire stop less. For IDM vehicles, the value range on the Y-axis for both the balanced and unbalanced case varied between $\{2.0 \textit{ and } 1.60\}$ and $\{1.80 \textit{ and } 1.40\}$ respectively. Whereas this range for Krauss drivers is between $\{0.76 \textit{ and } 0.74\}$, for both turning movement distributions. Thus, there is a range difference of roughly one additional stop between Krauss and IDM vehicles, which is likely to be explained by the fact that IDM vehicles always perform a brief stop at the intersection. Even when their lane is currently serviced with a green signal. During this brief stop an IDM vehicle might check if there are no conflicting vehicles on the intersection, i.e. it can confirm that it is safe to pass the intersection.

Overall, and for both the [50,50] and [20,80] turning movement distributions, a similar trend is shown by all methods when comparing these to SCHIC no prediction. The 95% confidence intervals of the *Oracle, fixed-label, split-vehicles and LCP* are quite wide compared to their performance improvements. The performance potential of the *Oracle* is negligible at 200 [veh/hour], but increased for the higher three vehicle demands (400, 600, 800 [veh/hour]). *Fixed-label, split-vehicles and LCP* show some fluctuating performance, however, during the [20,80] turning movement distribution their performance differences get smaller for the higher two vehicle demands. In conclusion, applying either of the four methods is beneficial to improve the performance as compared to SCHIC no prediction.



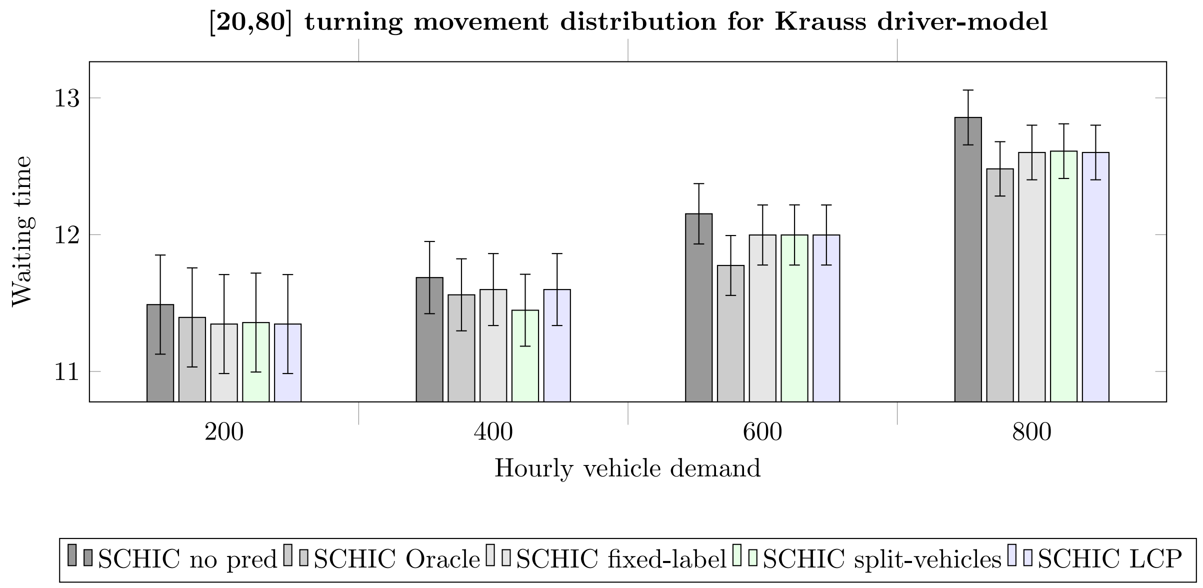


Figure 6.10: Average waiting time per vehicle in the hold-out test-set with 15 hours of simulation per hourly vehicle demand. Waiting time is defined as the total time that a vehicle has a speed below 0.1 m/s. Error-bars are 95% confidence intervals. Scenario B: Krauss driver-model.

(3) Average waiting time: Figure 6.10 shows the average waiting time per vehicle, which is defined as the total time that a vehicle has a speed below 0.1 m/s.

When observing the *Oracle* during the balanced [50,50] case, it is clear that it is outperformed during the vehicle demands of (400, 800) [veh/hour]. At 400 [veh/hour] the *Oracle* is slightly outperformed by both *fixed-label* and *split-vehicles*. At 800 [veh/hour] only *split-vehicles* outperforms the *Oracle*.

The slightly poorer performance of the *Oracle* might be explained as follows. For all methods, the 95% confidence intervals are quite large in relation to the average differences between these methods. Two potential reasons for these large confidence intervals are given below. All vehicle release times into the network are randomized a priori, and therefore identical. However, each vehicle might arrive at the intersection at different moments in time due to the stochastic speeds on the 200 m approach towards the intersection. Secondly, and arguably less probable, the braking profiles of Kraus vehicles are stochastic. Thus, the amount of time that each vehicle has a speed below 0.1 m/s (i.e. waiting-time) might differ too, causing fluctuations in the measured waiting-time.

The fuzzy approach of *split-vehicles* might be beneficial again in the balanced [50,50] scenario, as explained prior in: ◇ Thought experiment 1, extension dilemma for closely arriving vehicles.

In the unbalanced [20,80] case, the *Oracle* is outperformed during the lower vehicle demands of (200, 400) [veh/hour]. However, in the higher vehicle demands of (600, 800) [veh/hour] the *Oracle*'s performance is superior.

In conclusion, *split-vehicles* again performs well in the balanced [50,50] scenario. In the unbalanced [20,80] scenario, *fixed-label, split-vehicles and LCP* show similar performance but do not reach the performance of the *Oracle* during the vehicle demands above 600 [veh/hour].

6.5 Conclusion

This chapter discussed the results of the proposed method LCP by comparing it to SCHIC and its baseline methods: {*Oracle, fixed-label and split-vehicles*}. The *Oracle* is implemented here as a perfect Lane Change Prediction by means of Vehicle to Intersection Communication (V2I). The *Oracle* is applied when a vehicle reaches the same detection sensor as used by the other methods. An *Oracle* is a perfect classifier which demonstrates the maximum performance potential of the classification models: *fixed-label and LCP*. Secondly, it serves as an interesting comparison to the non-classification based method: *split-vehicles*. Two turning movement distributions are considered in this work, *[left-turn: 0.5, straight-ahead: 0.5]* and *[left-turn: 0.2, straight-ahead: 0.8]* denoted as [50,50] and [20,80] respectively.

The first set of results are obtained in Scenario A: Intelligent Driver Model (IDM). In this scenario the dynamic behaviour of all vehicles is deterministically controlled by IDM.

SCHIC Oracle is compared to *SCHIC no prediction*, across all hourly vehicle demands within the [50,50] and [20,80] turning movement distribution, the *Oracle* increases the average speed here with 0.56% and 0.73% respectively. The average number of stops per vehicle is reduced by 1.21% and 1.84% respectively. Although the behaviour of IDM is deterministic, this does allow for a clear determination of the *Oracle*'s performance.

At first glance, the results on the average speed and average waiting time indicate that LCP and the baseline methods of {*fixed-label and split-vehicles*} perform quite similar for both the balanced and unbalanced turning movement distributions. However, during the balanced turning movement distribution, i.e. *[left-turn: 0.5, straight-ahead: 0.5]*, the accuracy of *fixed-label* is reduced to just $acc_{fixed-label\ IDM} = 50\%$ which reduces its performance compared to the other methods.

In this balanced [50,50] scenario, the method of *split-vehicles* performs quite well. In conclusion, *LCP* reached a high predictive accuracy of $acc_{LCP\ IDM} = 81\%$, however it only matches or outperforms *split-vehicles* during the two higher vehicle demands.

In the unbalanced [20,80] scenario LCP reaches an accuracy of $acc_{LCP\ IDM} = 89\%$, however, all methods, i.e. the {*Oracle, fixed-label, split-vehicles and LCP*} show a comparable performance potential for SCHIC.

The second set of results are obtained in Scenario B: Krauss driver-model. In which all vehicles are controlled by the stochastic Krauss driver-model, which applies a slightly more aggressive driving style than IDM.

During a few vehicle demands, the *Oracle* is outperformed by *split-vehicles* as well as by *fixed-label and LCP* on the average speed, but the margins are slim with just $0.0150\ to\ 0.0156\ m/s$. In addition, these margins remain well within the confidence intervals of $\pm 0.0389\ to\ \pm 0.0550\ m/s$. Recall that the average speed of each vehicle is measured on its entire route of $400m$ throughout the network. The exit-lanes are $200m$ in length, where the vehicles are likely able to travel at the speed-limit of $50km/h\ or\ 13.9m/s$. But these speeds fluctuate now due to the stochastic properties of the Krauss driver-model. Lastly, Krauss vehicles might arrive at the intersection at different moments in time, since their speed fluctuates during their $200m$ approach towards the intersection.

Therefore, these negligible differences are likely a result of (1) the limited performance potential of all methods, combined with, (2) the stochastic properties in the speed profiles of the Krauss driver-model on the entry -and exit-lanes (3) a potential fluctuation in arrival times at the intersection. The small performance differences could be appointed to statistical insignificance on this performance metric.

Overall, *SCHIC Oracle* increases the average speed compared to *SCHIC no prediction* for both turning-movement distributions of [50,50] and [20,80] with 0.40% and 0.62% respectively. The average number of stops reduced by 0.79% and 1.44% respectively. Although these improvements are slightly lower than with the *IDM* scenarios, they are comparable.

It can be concluded that *LCP* and *fixed-label* show fluctuations in the balanced Krauss scenarios where their accuracies are just 51% and 49%, respectively. Nonetheless, it can be concluded that *fixed-label, split-vehicles and LCP* perform similarly with a slight advantage to *split-vehicles*. This advantage might be present since *LCP* only achieved an accuracy of $acc_{LCP\ Krauss} = 51\%$.

As for the unbalanced [20,80] scenario, the accuracy of LCP rose to $acc_{LCP\ Krauss} = 80\%$, however, it does not surpass the bias in the observed turning movement distribution. As a logical result, LCP does not outperform *split-vehicles and fixed-label* which are all based on the same bias as found in the turning movement distribution. The performance of these three methods is similar, and converge to each other when the vehicle demands increases in the unbalanced [20,80] scenario.

In conclusion, for both the IDM -and Krauss scenarios the performance potential of all methods, i.e. the {*Oracle, split-vehicles, fixed-label and LCP*}, are quite comparable. Although the *Oracle* increases the average speed (between 0.40% to 0.73%) and reduces the average number of stops (between 0.79% to 1.84%) for all scenarios, it does contain relatively wide confidence intervals. In retrospection, the performance gain of the *Oracle* might be increased in a more isolated scenario, where the *Oracle* is applied on all entry-lanes of the intersection instead of on half of its entry-lanes as is done now. However, this scenario requires a lane-split on all approaches towards the intersection, which is quite uncommon.

In the next and final chapter this study will be concluded.

Chapter 7

Conclusion

In this chapter we reflect on the research process and its outcomes.

For this thesis, the adaptive traffic light controller SCHIC was analysed. This controller is well-documented and forms the basis of a controller which is used in real-life, namely SURTRAC. SCHICs performance could potentially be hindered by two uncertainties which are commonly found in reality. These uncertainties are (1) inter-phase uncertainty introduced by the presence of left-turning traffic and (2) typical lane dimensions which limit the look-ahead time for vehicle detection to realistic values.

Although SCHIC is a promising controller, its performance is not well reported for the above two uncertainties, which are however, commonly found in reality. This knowledge gap is also present for the baseline methods which can cope with these problems, as explained in the next paragraph. Therefore, the aim of this thesis is to evaluate SCHIC with both (1) inter-phase uncertainty and (2) realistic look-ahead times, this is achieved by applying SCHIC on a (simplified) typical Dutch urban intersection located in The Hague.

In order to deal with these two problems, the following standard statistical methods were selected as baselines: *fixed-label* and *split-vehicles*. However, these do not utilize the potential sensing capabilities at the intersection (e.g. camera/radar) and often do not achieve a satisfying accuracy. The reason for their unsatisfying accuracy is the fact that these baseline methods only use the bias which is found in the observed turning movement distribution at the intersection.

In this thesis, a Lane Change Prediction (LCP) method based on machine learning was proposed and implemented. LCP extends its input information compared to the baselines, since it uses the turning movement distribution (just as the baselines do), but also uses the observed anticipative driving style of approaching vehicles, which might increase its accuracy. The main research question was whether the potentially higher accuracy of LCP would improve SCHICs traffic flow performance as compared to the baselines.

This paragraph partly repeats the experimental setting, by giving a brief summary. This study uses simulated driving behaviour which is obtained with Simulation of Urban Mobility [19] (SUMO). Within SUMO, the dynamic driving behaviour of a vehicle is determined by a driver-model. Two driver-models are evaluated in: Scenario A: Intelligent Driver Model (IDM) and Scenario B: Krauss driver-model. IDM is a deterministic driver-model, whereas Krauss also applies stochastic properties to the driving behaviour. In both Scenario A and B, a balanced and unbalanced turning-movement distribution is evaluated, respectively *[left-turn: 0.5, straight-ahead: 0.5]* and *[left-turn: 0.2, straight-ahead: 0.8]*. Each turning-movement distribution is also referred to as [50,50] and [20,80], respectively. For each turning-movement distribution the vehicle release times are identical, i.e. Scenario A and B only differ in the selected driver-model.

In this study it is shown that the accuracy of LCP in simulation for one driver model, i.e. for IDM, is indeed higher than the statistical baseline methods. These predictive accuracies for IDM vehicles in the balanced turning movement scenario are: $acc_{LCP,IDM} = 81\%$ and in the unbalanced scenario: $acc_{LCP,IDM} = 89\%$, where *fixed-label* only reaches accuracies of 50% and 80% respectively. Despite these much higher accuracies, LCP does not significantly improve SCHICs traffic flow performance over its baseline methods, which answers the research question.

Next to this study on the baseline methods and LCP, this thesis also examined whether perfect knowledge on lane changes (potentially given by Vehicle to Intersection Communication (V2I)) - an *"Oracle" method*, would improve SCHICs performance. The performance potential of the *Oracle* is quite low, but does decrease the number of stops for both IDM -and Krauss drivers with 1.53% and 1.12%, respectively. However, with the confidence intervals being quite wide and the performance potential of the four methods being comparable, it was concluded that the performance potential of all methods, i.e. {*fixed-label, split-vehicles, LCP*} and even the {*Oracle*}, is comparable.

Overall, it can be concluded that LCP did not improve SCHICs performance compared to its baseline methods. Furthermore, the theoretical performance potential of LCP (i.e. the *Oracle*) is inherently limited in this scenario with (1) the lane-splits being very close to the intersection and (2) given our evaluated traffic scenarios. The performance potential of the *Oracle* could be increased when the lane-splits are placed further from the intersection. Additionally, there could be more performance potential for LCP when all entry-lanes of the intersection contain a lane-split, instead of half of the entry-lanes. Further research will have to examine whether the above suggestions improve the performance potential of the *Oracle* and LCP.

Based on these results, the four research questions can be answered.

- R.Q. 1. Given a typical Dutch intersection with limited lane-dimensions and hence limited look-ahead times, how does SCHICs local performance relate to a simpler but commonly applied *Time-Gap Controller*?

Prior to this work, SCHIC has not been well evaluated under these short look-ahead times of 3 *seconds*, combined with inter-phase uncertainty. This thesis shows that SCHIC significantly outperforms the simpler *Time-Gap Controller* (Section 6.2) which is most notably shown in the average speed of all vehicles in simulation (see Figure 6.1 and Figure 6.7).

The poor performance of the *Time-Gap Controller* is possibly due to the fact that it only uses a look-ahead time of 2.2 *seconds* on half of its entry-lanes, whereas SCHIC uses 3.0 *seconds* on these lanes. Some more reasoning for these performance differences is given in Section 6.3.

• R.Q. 2. What is the performance potential for SCHIC when perfect Lane Change Predictions are applied on this typical Dutch intersection with two lane-splits?

The maximum performance potential of LCP is demonstrated by means of perfect predictions as obtained with an *Oracle*. This means that 3.0 *seconds* before a vehicle would reach the intersection it communicates its desired entry-lane directly to SCHIC by means of Vehicle to Intersection Communication (V2I). This maximum performance potential of LCP is quite low, so it is likely unnoticed by a single driver when he/she passes the intersection. However, averaged over all vehicles in simulation, there is an increase in the average speed and a reduction of the number of stops.

The results are explained below and shown in Table 7.1.

In Scenario A, the Intelligent Driver Model (IDM), the *Oracle* increased the average speed over *SCHIC no prediction* across the different turning-movement distributions with 0.56% in the balanced [50,50] scenario and 0.73% in the unbalanced [20,80] scenario. Averaged over both the balanced and unbalanced scenario, the average number of stops decreased by 1.53% for the deterministic IDM vehicles.

Similar performance is shown in Scenario B, the Krauss driver-model. The *Oracle* increased the average speed over *SCHIC no prediction* across the different turning-movement distributions, with 0.40% in the balanced [50,50] scenario and 0.62% in the unbalanced [20,80] scenario. Averaged over both the balanced and unbalanced scenario, the average number of stops decreased by 1.12% for Krauss vehicles.

Performance of SCHIC Oracle over SCHIC no prediction.	**Scenario A: IDM**	**Scenario B: Krauss**
Average speed increase in [50, 50]	0.56%	0.40%
Average speed increase in [20, 80]	0.73%	0.62%
Average number of stops reduction in [50, 50] and [20, 80] combined	1.53%	1.12%

Table 7.1: Performance metrics of SCHIC Oracle over SCHIC no prediction.

• R.Q. 3. How accurate are Lane Change Predictions (LCPs) in SUMO?

A Random Forest is selected as our classifier in order to obtain Lane Change Predictions (LCPs). In this thesis an accurate LCP based on an anticipative driving style is only realised for the deterministic Intelligent Driver Model (IDM).

The predictive accuracy for IDM vehicles is $acc_{LCP,IDM} = 81\%$ in the balanced turning movement scenario and $acc_{LCP,IDM} = 89\%$ in the unbalanced scenario. The accuracy is expected to be higher in the unbalanced scenario, as during its training phase the machine learning model observed the unbalanced lane-occupancy.

The Krauss driver-model contains stochastic properties and LCP has the following accuracies: $acc_{LCP,\,Krauss} = 51\%$ in the balanced scenario and $acc_{LCP,\,Krauss} = 80\%$ in the unbalanced scenario. Based on these accuracies and learning-curves of the predictive models for Krauss, it is assumed that these models have only applied the bias found in their training-set. It is possible that the current feature vector contains too little information on the driving style of a Krauss vehicle. The Random Forest might be able to predict lane changes based on an enhanced feature vector which contains more variables. Lastly, a more advanced learning model might learn to predict the lane changes of a Krauss vehicle based on the current feature vector.

- R.Q. 4: How does SCHIC in combination with LCP perform in comparison to the existing statistical baseline methods of *split-vehicles* and *fixed-label*?

Finally, SCHICs traffic flow performance with LCP has been compared to its baseline methods *fixed-label* and *split-vehicles*. These two baseline methods rely on the historic traffic demands per turning-movement, i.e. {*left-turn, straight-ahead*}.

In Scenario B, for the Krauss driver-model with a balanced distribution of turning-movements, LCP only reaches an accuracy of $acc_{LCP,\,Krauss} = 51\%$. Which is almost similar to the accuracy of a *fixed-label* classifier. Given this low accuracy, mis-classifications are expected to occur quite often and might explain the more unpredictable performance increase of SCHIC when LCP or *fixed-label* is applied. Overall, *fixed-label* and LCP show a similar performance increase but win and lose from each other throughout the balanced turning movement scenario. In the unbalanced Krauss scenario, the accuracy of both LCP and *fixed-label* rise to: $acc_{LCP,\,Krauss} = acc_{fixed-label,\,Krauss} = 80\%$. In this unbalanced scenario, it can be concluded that all three methods {*fixed-label, split-vehicles and LCP*} show some fluctuations during the different hourly demands but perform quite similar overall.

In Scenario A, IDM, with turning movement distributions *[left-turn: 0.2, straight-ahead: 0.8]*, LCP reaches an accuracy of $acc_{LCP\ IDM} = 89\%$ and ever so slightly outperforms *fixed-label* and *split-vehicles*. However, all four methods, including the *Oracle*, perform almost comparable. It can be concluded that LCP ever so slightly outperforms these statistical baseline methods when its accuracy is both above 80% and above the accuracy of a fixed-label classifier.

For both Krauss and IDM, during the balanced turning-movement distributions *split-vehicles* performs quite well. Its fuzzy approach seems advantageous in a balanced turning-movement scenario compared to the binary classification approach of LCP. *Split-vehicles* fuzzy approach allows SCHIC to consider two split vehicles which are both scheduled by SCHIC. By considering these two split vehicles, the duration of a current phase might be extended more often when compared to the classification approach of LCP. This could allow closely arriving vehicles to join this extended green phase more often in the balanced [50,50] scenario, resulting in a potential conceptual advantage for *split-vehicles.*

In conclusion, despite LCPs higher accuracies for IDM, it does not significantly improve SCHICs performance over its baseline methods *fixed-label* and *split-vehicles.* For both the IDM and Krauss scenarios the performance potential of all methods, i.e. the {*Oracle, split-vehicles, fixed-label and LCP*}, are quite comparable.

7.1 Discussion & Further Research

This section first discusses a limitation of this thesis ($\diamond$). Subsequently, some suggestions for further research are given ($*$).

- $\diamond$ The prediction horizon of Lane Change Prediction (LCP) is inherently limited to a short future time-frame on which each human driver anticipates. As shown in literature, these accuracies reduce rapidly when the prediction horizon is increased beyond 3.0 *seconds* [18].

- $*$ Two features are used by LCP which are based on acceleration measurements, these measurements are obtained without noise from the traffic simulator SUMO. However, obtaining acceleration measurements in reality, with real-world sensors, is difficult and leads to noisy measurements. Furthermore, it is shown that the acceleration based features do not aid the predictive accuracy much (Section 5.3.6). Given the above two reasons, it is advised to not use the acceleration based features in further research on LCP.

- $*$ The potential societal impact of this research will be more relevant when LCP is not applied to SCHIC, but applied to an actual commercially deployed controller. LCP brings a limited performance potential to SCHICs traffic flow. However, SCHIC is not deployed commercially and its scheduling algorithm is not as flexible as some (commercial) controllers. To clarify, SCHIC cycles through its green phases in a fixed order and each green phase runs for at least 5 *sec* [1]. However, a more flexible controller which can consider a green signal for any given entry-lane at any given moment in time, could harness more performance potential from both the *Oracle* and LCP. Secondly, given the ongoing development to Machine Learning models and sensor technologies, it could be possible that lane changes can be predicted further in advance with higher accuracies in the near future. Predictive accuracies might improve thanks to improved Machine Learning models, improved feature selection (e.g. by AutoML) or more accurate (radar)-sensors. The above two arguments, i.e. (1) a more flexible (commercial) controller and (2) more accurate Lane Change Predictions, could both increase the traffic flow performance of LCP and its societal relevance.

- $*$ Although there is only a slight traffic flow improvement possible when applying Lane Change Predictions (LCPs) to SCHIC. The study on LCP can also be used in further research which is solely aimed at autonomously driving vehicles.

- $*$ The performance potential of perfect Lane Change Predictions is evaluated with an *Oracle* (or V2I) being applied quite close to the intersection at 3.0 *seconds* from the stop-line. Vehicle detection on the short entry-lanes after the lane-split is realized at 2.2 *seconds* from the stop-line. So, the *Oracle* increases the look-ahead time from 2.2 *seconds* to 3.0 *seconds*, which is an increase of roughly 36%. However, the absolute increase is only 0.8 *seconds*. Thus, it can be interesting to investigate the performance potential of the *Oracle* when it is applied further from the stop-line, e.g. at {4.0 *seconds or* 5.0 *seconds*}. Lane change predictions will be less accurate at these prediction points, but the performance potential for SCHIC with an *Oracle* and even its statistical baseline methods might be higher.

* The evaluated intersection contains two lane-splits, so it contains a lane-split on just half of its entry-lanes. There could be more performance potential for SCHIC in combination with LCP when all entry-lanes of the intersection contain a lane-split. However, an intersection with a lane-split on all of its entry-lanes is arguably less commonly found than the intersection examined in this research.

* SCHIC is compared to a simple and commonly used *Time-Gap Controller*, which is expected to reach its own optimal performance, since it is used with its close to default parameter settings. However, the look-ahead time of SCHIC is slightly larger on half of its entry-lanes. It would therefore be interesting to evaluate the performance of the *Time-Gap Controller* when it uses the same look-ahead times as SCHIC on all entry-lanes.

Abbreviations

COP	Combined Optimization of Phases [5].
ERIS	Expressive Real-time Intersection Scheduling [7].
IDM	Intelligent Driver Model.
LCP	Lane Change Prediction.
PSS	Phase Switching Sequence.
SCHIC	Schedule-driven Intersection Control [1].
SCOOT	Split-Cycle Offset Optimization Technique [23].
SUMO	Simulation of Urban Mobility [19].
SVM	Support Vector Machine.
TSC	Time Series Classification.
V2I	Vehicle to Intersection Communication.
VA	Vehicle Actuated Control.

Glossary

The following terms are directly derived from the original publication of SCHIC by Xie et al. [1]. Behind each term, a page number is listed in the form of: *(p xyz)* which corresponds to the first occurrence of this term in [1]. Two terms (Inflows (IF), Road Flow List (RF)) are however only found in SCHICs coordination paper [27].

(X_o, s_o) the previous *state groups* that can reach the current *state* (X,s) are (X_o, s_o), with o referring to *old*. To compute the *value rows* of (X, s), X_o = X with $x^{(s)} = x^{(s)} - 1$ is used. Here, s_o is the phase index of the previous state group (X_o, s_o) that can reach the current state (X,s) (p 177). 96

$C^{(i)}$ a *cluster* sequence on *route* i. Which is an aggregate flow representation by aggregating vehicles on *routes* into sequences of *clusters* (p 172). 96

$G^{(i)}_{min}$, $G^{(i)}_{max}$ Minimum and maximum green times for phase with *phase index i* (p 170). 96

H_O optimization horizon in seconds. 96

$H_O^{\#}$ finish time of the optimal *state* with minimal cumulative delay (p 171). 96

H_P limited prediction horizon in seconds in which new vehicles are detected (p 170). 96

X_{empty}, X_{full} special X arrays for the empty and full status, which have $x^{(i)} = 0$ and $x^{(i)} = |C^{(i)}|$ for $\forall$ i, respectively for X_{empty} and X_{full} (p 173). 96

$c^{(ij)}$ this is the j^{th} *cluster* in $C^{(i)}$, where $j \in [1, |C^{(i)}|]$ and $|C^{(i)}|$ is the number of *clusters* on the *route* i (p 173). 96

k^{th} **job** the k^{th} job is the cluster $c^{(s^{(k)}, x^{s^{(k)}})}$, see also *schedule status X* (p 173). 96

$q^{(i)}$ queue size on *route* i (p 171). 96

$X^{(k)}$ all *schedule statuses X* which sum to integer value k. I.e. $X^{(k)} = \{X : \sum_{i=1}^{|I|} x^{(i)} \equiv k\}$. The scheduler explores all possible states in *state space D* using $\forall$ X $\in X^{(k)}$ and $\forall s \in [1, |I|]$ (p 176). 96

$|H_O|$ the number of individual time-steps in optimization horizon H_O (p 171, 177). 96

state space Ω state space based on the unclustered vehicle flow and using timesteps of size $\Delta =$ 0.5 sec. This state space is only used when we do not apply vehicle clustering (p 171). 96

state space D state space containing all possible schedules but based on the clustered vehicles. This space D is a subspace of the actual state space Ω. *State variables* are (X, s, t, d) (p 171). 96

cluster a representation of arriving vehicle(s), each cluster (c) has five attributes: ($|c|$, arr, dep, dur, fr) $|c|$ = vehicle count, arr = arrival of first vehicle, dep = departure of last vehicle, dur = dep - arr, fr = $|c|$ / dur (p 172). 34, 96

cycle the total time it takes for all phases of one intersection, to receive their allocated green -and inter-green times (inter-green time is only the yellow-time for SCHIC). 17, 18, 96

full-clearance state a *state* that clears all vehicles in the prediction horizon (p 171). 96

Inflows (IF) a green phase can service multiple entry-lanes (each for a single movement m) at the same time. E.g. two opposing entry-lanes for straight-ahead movements can be serviced by the same green phase (as seen in Figure 3.1). All individual road flows with an individual movement m ($C_{RF,m}$) which are serviced by the same green phase are simply merged into a separate *Inflow List*: $C_{IF,i}$. So, for each individual green phase with index i, a separate list is maintained: $C_{IF,i}$. Here, the Inflows (IF) is used to describe all approaching traffic, i.e. IF is a set containing all individual *Inflow Lists*: $C_{IF,i}$. So, the following set is collected $IF = \{C_{IF,i} : \forall i \in |I|\}$. Where i is an individual green phase index and $|I|$ is the total number of green phases in the phase set I. (p 324 of SCHICs coordination paper [27]). 26, 30, 34, 96, 98

lane a portion of the pavement which is allocated to a single line of traffic which originates from the same direction, as often indicated by painted longitudinal lines or markers. 7, 24–28, 45, 96, 99

movement traffic flow from (some number of lanes of) an entry approach to an exit approach (p 170). 24, 26, 27, 29, 96, 99, 100, 118

offset the difference in starting times of two consecutive phases along a road. 17, 18, 96

phase defined as (i,g) or $g^{(i)}$ in which i is the phase index, g is a variable phase duration. g_c is the green time of the currently active phase (p 170). 24, 26, 30, 33, 34, 96, 99

phase design containing a set of phases I in which each phase with *phase index* i $\in$ [1, I] corresponds to the right-of-way for route i (p 170). 96

phase index each *phase index* i $\in$ [1, I] corresponds to the right-of-way for route i (p 170). 96

Phase Switching Sequence (PSS) defined as a sequence of phases. Each *PSS* is built upon an initial phase condition (i_c, g_c), where i_c is the current green phase index, and g_c is the the time that the current phase i_c has been green (p 170). 30, 33, 34, 96

Road Flow List (RF) a cluster sequence ($C_{RF,m}$), where each individual $C_{RF,m}$ contains the vehicles travelling towards the intersection on the entry-lane associated to movement m (p 324 of SCHICs coordination paper [27]). 26, 96, 98

route each route contains a set of non-conflicting *movements* that allow safe passage of vehicles simultaneously through the intersection. A simplifying restriction is applied since each movement is only allowed to be found in one route, i.e. routes do not share any movements (p 170). 96

samp samping interval of vehicle detection method, the temporal arrival distribution of vehicles in the prediction horizon H_P is divided into time segments. The number of time-segments is [1, H_P / *samp*] (p 172). 96

saturation flow rate $sfr = N_{lanes}/saturation\ headway$, where N_{lanes} is the number of lanes and saturation headway is the average headway between vehicles during saturated flow. (p 179). 96

schedule S specifies the order in which the clusters will pass through the intersection one by one. It is represented as a sequence of route indices ($s^{(i)}$, ..., $s^{(k)}$, ... , $s^{(|S|)}$) (p 173). 96

schedule status X a list which is linked to a *schedule* $S^{(k)}$ and defined as: X = ($x^{(i)}$, .. , $x^{(|I|)}$) where $x^{(i)} \in [\ 0,\ |C^{(i)}|]$ counts the number of elements in $S^{(k)}$ that are equal to i. In other words, each $x^{(i)}$ counts the number of clusters that have been scheduled for *route i*. The k^{th} job is *cluster* $c^{(s(k),x^{(s(k))})}$. (p 173). 96

split the amount of time within a phase where a traffic signal is green. 17, 18, 96

stage at the k^{th} stage, k clusters have been scheduled in schedule S so (k $\in [1, |S|]$). At *stage k*, the schedule lets the k^{th} job, which is the earliest cluster that remains on the route $s^{(k)}$, leave from the intersection at the earliest time (p 173). 30, 96

start-up lost time if we anticipate a queue this start-up lost time (a fixed time) is added to the job duration. Since the queue needs time to get up to speed when it is finally serviced with a green signal (p 171). 96

state a state in *state space D* representing a partial solution with a cumulative delay value (p 171). 34, 96

state group defined as a tuple (X,s) in which X is the *schedule status* and $s \in [1, |I|]$ means that the last job is scheduled on route s. Each state group also contains a table of *value rows* that allow to track back to all previous *state groups* to obtain a *schedule S* (p 176). 96

state variables in the *state space D* of the scheduling model, each *state* is associated with a partial *schedule* $S^{(k)}$. For each $S^{(k)}$ the *state variables* are defined as a tuple: (X, s, t, d) where k is the sum of all values in $X = [x^0, .., x^{|I|}]$, s = $s^{(k)}$, t is the actual finish time of the k^{th} job and d is the cumulative delay for all k scheduled jobs. (p 173). 96

Switchback(i) Switchback(i) = MinCyle - $G^{(i)}_{min}$ is the minimum time required for the traffic light to return to the phase index i in one cycle (p 178). 96

temporal arrival distribution arrival distribution of vehicles between $[0,\ H_p^{(i)}]$(p 171). 96

time resolution Δ all temporal values are rounded into a single time-step value by dividing it with this time resolution value Δ (p 171). 96

TL-PSS A *Phase Switching Sequence (PSS)* for a specific traffic light (p 170). 96

value rows Each state group contains a table of value rows in which each value row contains four elements ($\tilde{\mathrm{t}}$, $\tilde{\mathrm{d}}$, $\tilde{\mathrm{s}}$, $\tilde{\mathrm{y}}$). Where $|(X, s)|$, is the actual number of value rows for a single state group. These value rows are used to track back to the root-state when obtaining the *schedule S* from a certain *state group* (p 176). 96

Bibliography

[1] X.-F. Xie, S. F. Smith, L. Lu, and G. J. Barlow, "Schedule-driven intersection control," *Transportation Research Part C: Emerging Technologies*, vol. 24, pp. 168–189, 2012, ISSN: 0968090X. DOI: 10.1016/j.trc.2012.03.004. [Online]. Available: http://dx.doi.org/10.1016/j.trc.2012.03.004.

[2] Dutch Central Agency for Statistics, *Aantal wegvoertuigen blijft stijgen*, https://www.cbs.nl/nl-nl/nieuws/2019/14/aantal-wegvoertuigen-blijft-stijgen, [Online; accessed 20-05-2020], 2019.

[3] European Commission, "White Paper European transport policy for 2010: Time to decide," Brussels, Tech. Rep. 12-09-2001, 2001, pp. 1–179. [Online]. Available: https://ec.europa.eu/transport/sites/transport/files/themes/strategies/doc/2001_white_paper/lb_com_2001_0370_en.pdf.

[4] ——, "Transport in the European Union: Current Trends and Issues," Tech. Rep. April, 2018, pp. 1–144. [Online]. Available: https://ec.europa.eu/transport/sites/transport/files/2018-transport-in-the-eu-current-trends-and-issues.pdf.

[5] S. Sen and K. L. Head, "Controlled optimization of phases at an intersection," *Transportation Science*, vol. 31, no. 1, pp. 5–17, 1997. [Online]. Available: https://EconPapers.repec.org/RePEc:inm:ortrsc:v:31:y:1997:i:1:p:5-17.

[6] S. Dhamija and P. Varakantham, "TuSeRACT: Turn-Sample-Based Real-Time Traffic Signal Control," pp. 1–15, 2018. arXiv: 1812.05591. [Online]. Available: http://arxiv.org/abs/1812.05591.

[7] R. Goldstein and S. F. Smith, "Expressive Real-time Intersection Scheduling," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 01, pp. 9882–9883, 2018. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/12073.

[8] S. F. Smith, "Coping with Real-World Challenges in Real-Time Urban Traffic Control," *Transportation Research Board 93rd Annual Meeting*, vol. 93, no. January 2014, 2015. [Online]. Available: https://trid.trb.org/View/1288134.

[9] M. Diamantis, P. Theodora, D. Christina, P. Ioannis, and M. Papageorgiou, "Simulation Study of Centralized Versus Decentralized Approaches to Signal Control of Large-Scale Urban Networks," *Transportation Research Board (TRB)*, no. January, pp. 1–14, 2017.

[10] S. Baluja, M. Covell, and R. Sukthankar, "Traffic Lights with Auction-Based Controllers: Algorithms and Real-World Data," 2017. arXiv: 1702.01205. [Online]. Available: http://arxiv.org/abs/1702.01205.

[11] M. Covell, S. Baluja, and R. Sukthankar, “Micro-Auction-Based Traffic-Light Control: Responsive, Local Decision Making,” *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, vol. 2015, no. October, pp. 558–565, 2015. DOI: `10.1109/ITSC.2015.98`.

[12] Z. H. Khattak, M. J. Magalotti, and M. D. Fontaine, “Operational performance evaluation of adaptive traffic control systems: A Bayesian modeling approach using real-world GPS and private sector PROBE data,” *Journal of Intelligent Transportation Systems*, no. May, pp. 1–15, 2019, ISSN: 1547-2450. DOI: `10.1080/15472450.2019.1614445`. [Online]. Available: `https://www.tandfonline.com/doi/full/10.1080/15472450.2019.1614445`.

[13] Y. Hou, S. M. Seliman, E. Wang, J. D. Gonder, E. Wood, Q. He, A. W. Sadek, L. Su, and C. Qiao, “Cooperative and Integrated Vehicle and Intersection Control for Energy Efficiency (CIVIC-E2),” *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 7, pp. 2325–2337, 2018, ISSN: 15249050. DOI: `10.1109/TITS.2017.2785288`.

[14] Q. V. Dang and H. Rudová, “Enhanced Scheduling for Real-Time Traffic Control,” *Proceedings of the 2018 IEEE Symposium Series on Computational Intelligence, SSCI 2018*, pp. 578–585, 2019. DOI: `10.1109/SSCI.2018.8628731`.

[15] S. Chen and D. J. Sun, “An Improved Adaptive Signal Control Method for Isolated Signalized Intersection Based on Dynamic Programming,” *IEEE Intelligent Transportation Systems Magazine*, vol. 8, no. 4, pp. 4–14, 2016, ISSN: 19391390. DOI: `10.1109/MITS.2016.2605318`.

[16] D. Manolis, T. Pappa, C. Diakaki, I. Papamichail, and M. Papageorgiou, “Centralised versus decentralised signal control of large-scale urban road networks in real time: a simulation study,” *IET Intelligent Transport Systems*, vol. 12, no. 8, pp. 891–900, 2018, ISSN: 1751-956X. DOI: `10.1049/iet-its.2018.0112`.

[17] D. Manolis, D. Christina, P. Ioannis, and P. Markos, “Comparative evaluation of two alternative approaches for the decentralized real-time signal control of urban networks,” *Proceedings of the 4th International Symposium & 26th National Conference on Operational Research*, no. JUNE, 2015.

[18] M. Garcia Ortiz, J. Fritsch, F. Kummert, and A. Gepperth, “Behavior prediction at multiple time-scales in inner-city scenarios,” *IEEE Intelligent Vehicles Symposium, Proceedings*, no. IV, pp. 1068–1073, 2011. DOI: `10.1109/IVS.2011.5940524`.

[19] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y. P. Flotterod, R. Hilbrich, L. Lucken, J. Rummel, P. Wagner, and E. Wiebner, “Microscopic Traffic Simulation using SUMO,” *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, vol. 2018-Novem, pp. 2575–2582, 2018. DOI: `10.1109/ITSC.2018.8569938`.

[20] M. C. Dunne and R. B. Potts, “Algorithm for Traffic Control,” *Operations Research Special Transportation Science Issue*, vol. 12, no. 6, pp. 870–881, 1964. [Online]. Available: `https://www.jstor.org/stable/168173`.

[21] D. Pescaru, “Urban traffic congestion prediction based on routes information,” *SACI 2013 - 8th IEEE International Symposium on Applied Computational Intelligence and Informatics, Proceedings*, pp. 121–126, 2013. DOI: `10.1109/SACI.2013.6608951`.

[22] Wikipedia, *Grid plan*, `https://en.wikipedia.org/wiki/Grid_plan`, [Online; accessed 17-12-2020], 2020.

[23] P. Hunt, D. Robertson, R. Bretherton, and R. Winton, "Scoot - a traffic responsive method of coordinating signals," *Transportation Research Board*, p. 41, 1981. [Online]. Available: `https://trid.trb.org/view/179439`.

[24] E. S. J. Raphael, "Towards Dynamic Coalition Formation for Intelligent Traffic Management," *Eumas 2017*, vol. 1, pp. 172–186, 2018. DOI: `10.1007/978-3-030-01713-2`. [Online]. Available: `http://dx.doi.org/10.1007/978-3-030-01713-2_13`.

[25] H.-C. Hu and S. F. Smith, "Softpressure: A Schedule-Driven Backpressure Algorithm for Coping with Network Congestion," 2019. arXiv: `1903.02589`. [Online]. Available: `http://arxiv.org/abs/1903.02589`.

[26] A. Czechowski, X. Zhang, and R. Blokpoel, "Cooperative queue data for adaptive traffic control," *ITS World Congress*, vol. 25, no. September, 2018. [Online]. Available: `http://adas.cvc.uab.es/maven/wp-content/uploads/sites/16/2018/10/ITSC2018_Czechowski_Zhang_Blokpoel.pdf`.

[27] X.-F. Xie, S. F. Smith, and G. J. Barlow, "Schedule-Driven Coordination for Real-Time Traffic Network Control," *Twenty-Second International Conference on Automated Planning and Scheduling*, vol. 15213, pp. 323–331, 2012. [Online]. Available: `https://www.aaai.org/ocs/index.php/ICAPS/ICAPS12/paper/viewPaper/4701`.

[28] Y. Hou, P. Edara, and C. Sun, "Modeling mandatory lane changing using bayes classifier and decision trees," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 2, pp. 647–655, 2014, ISSN: 15249050. DOI: `10.1109/TITS.2013.2285337`.

[29] T. J. Gates, S. D. Schrock, and J. A. Bonneson, "Comparison of portable speed measurement devices," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1870, pp. 139–146, 2004, ISSN: 03611981. DOI: `10.3141/1870-18`.

[30] J.J. Olstam and A. Tapani, "Comparison of car-following models for simulation," *Transportation Research Record*, no. 1678, pp. 116–127, 1999, ISSN: 03611981. DOI: `10.3141/1678-15`.

[31] S. Krauß, "Microscopic modelling of traffic flow: Investigation of collision free vehicle dynamics," 1998, LIDO-Berichtsjahr=1999, [Online]. Available: `https://elib.dlr.de/8380/`.

[32] M. Pourabdollah, E. Bjarkvik, F. Furer, B. Lindenberg, and K. Burgdorf, "Calibration and evaluation of car following models using real-world driving data," *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, vol. 20, no. 10, pp. 1–6, 2017. DOI: `10.1109/ITSC.2017.8317836`.

[33] Martin Treiber *, Ansgar Hennecke and D. Helbing†, "Congested traffic states in empirical observations and microscopic simulations," *Physical Review E*, vol. 62, no. 2, pp. 1805–1824, 2000, ISSN: 1475-3219. DOI: `10.1103/PhysRevE.62.1805`. [Online]. Available: `https://link.aps.org/doi/10.1103/PhysRevE.62.1805`.

[34] M. Treiber and A. Kesting, "The Intelligent Driver Model with Stochasticity –New Insights Into Traffic Flow Oscillations," *Transportation Research Procedia*, vol. 23, pp. 174–187, 2017, ISSN: 2352-1465. DOI: `10.1016/j.trpro.2017.05.011`. [Online]. Available: `http://dx.doi.org/10.1016/j.trpro.2017.05.011`.

[35] Wikipedia, *Support-vector machine*, `https://en.wikipedia.org/wiki/Support-vector_machine`, [Online; accessed 17-05-2020], 2020.

[36] S. Russel and P. Norvig, *Artificial Intelligence A Modern Approach, 2nd ed.* Englewood Cliffs, NJ, USA: Prentice-Hall, 2003, pp. 653–663.

[37] scikit-learn, *Documentation on sklearn.tree.DecisionTreeClassifier*, `https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html`, [Online; accessed 13-01-2021], 2021.

[38] Wikipedia, *Decision tree learning, Gini impurity*, `https://en.wikipedia.org/wiki/Decision_tree_learning`, [Online; accessed 13-01-2021], 2020.

[39] scikit-learn, *Documentation of sklearn on ensemble methods.* `https://scikit-learn.org/stable/modules/ensemble.html`, [Online; accessed 13-01-2021], 2021.

[40] X.-F. Xie, S. F. Smith, and G. J. Barlow, "Coordinated Look-Ahead Scheduling for Real-Time Traffic Signal Control (Extended Abstract)," *AAMAS*, pp. 1271–1272, 2012. [Online]. Available: `https://dl.acm.org/doi/pdf/10.5555/2343896.2343958`.

Appendix A

Literature study A: Related work on SCHIC

The three papers which are briefly summarized in Section 2.2 are explained in a more elaborate manner in this appendix.

Paper 1: lane-based job-scheduling by Dang [14]
Dang [14] improved SCHIC both on the traffic-input -and scheduling level in 2018 for a common real-world occurrence, namely vehicles which approach the intersection on opposing lanes and share the same green phase. Figure 1 (Fig. 1) in [14] clarifies this visually, i.e. cluster sets $\{a, b, e\}$ and $\{c, d\}$ both approach the intersection on opposing lanes and share the same green-phase for straight-ahead traffic. By original design, SCHIC does not consider the lane-direction l_d of clusters which share the same green-phase.

The objective function of SCHIC, i.e. to minimize the estimated cumulative delay of all observed vehicles, is now made more accurate by Dang [14] since the lane-direction of each entry-lane l_d is taken into account as well. Two improvements are evaluated by Dang [14]: SCHIC-new-clustering (SCHIC-nc) which considers the lane-direction l_d during the clustering process and SCHIC-2 which includes the new clustering method and an enhanced scheduling method.

Clustering of approaching vehicles is already explained in more detail in section 3.1.2, however in this paragraph a brief recap is given in order to understand the extensions of Dang [14]. SCHIC detects approaching vehicles and saves each of them in a data-structure known as a cluster. Each cluster is defined as (c, arr, dep), where c is the number of vehicles in the cluster with arr being the arrival time of the first vehicle and dep the departure time of the last vehicle. In order to reduce the problem size of the scheduler, SCHIC aggregates multiple clusters into a single cluster if their vehicles arrive within a threshold time: $thc = 3.0$. This threshold based clustering process is applied for each phase, regardless if clusters are found on opposing lanes. For example the merge process of c_1 and c_2 into c_0 is as follows: $arr_{c_0} = min(arr_{c_1}, arr_{c_2})$, $dep_{c_0} = max(dep_{c_1}, dep_{c_2})$ and $c_{c_0} = c_{c_1} + c_{c_2}$. Vehicles of the merged cluster (c_0) are assumed to be uniformly distributed between the arrival and departure time which is unlikely to be true when vehicles are found on opposing lanes.

SCHIC-new-clustering (SCHIC-nc) therefore only applies threshold based clustering on vehicles which travel in the same lane direction: l_d. SCHIC-new-clustering (SCHIC-nc) is able to use the original scheduling algorithm of SCHIC, since it still meets the requirement of this original scheduling algorithm, i.e. a single cluster sequence per phase index (i).

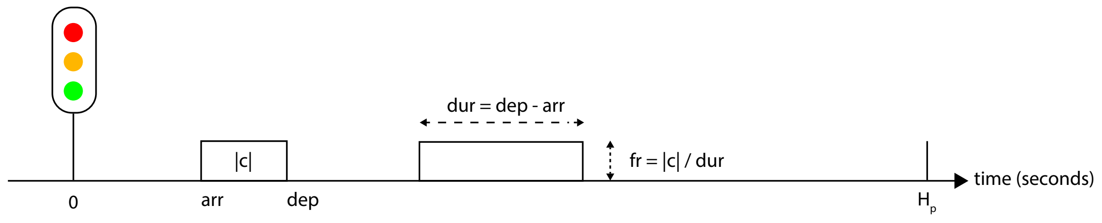


Figure A.1: Visual explanation of clusters – note that clusters can be found on opposing lanes. Each cluster is defined by (arr, dep and $|c|$). Image reproduced from Xie et al. [1].

SCHIC-2 extends the default scheduling algorithm by allowing to individually schedule each cluster on each lane direction l_d. The default scheduling algorithm only allows to schedule a single green phase for vehicles on opposing lane directions which share the same green phase. The default scheduler of SCHIC is extended to allow the scheduling of multiple cluster sequences per phase.

Therefore, a separate cluster sequence is now maintained for each lane direction l_d of each phase i. The scheduler of SCHIC-2 is referred to as a *two machine scheduling* method in Figure 1 (Fig. 1) in [14].

To visualize and more clearly clarify SCHIC-2, Figure 1 (Fig. 1) in [14] shows four lanes in total which are serviced in two individual green phases. The two horizontal -and two vertical lanes are associated with one of the two green phases. SCHIC-2 maintains four cluster sequences, i.e. one for each of the four individual lane-directions l_d. Whereas SCHIC-nc maintains only two cluster sequences for the two green phases. Both SCHIC-nc and SCHIC-2 evaluate different potential schedules by iteratively adding clusters one by one from the available cluster sequences. SCHIC-2 and SCHIC-nc evaluate these different potential schedules to select the one with lowest cumulative delay.

Performance metrics are given by considering two green-phases where only straight-ahead and right-turn movements are allowed (phases with indices 0,2 in Figure 1.4a). The 5x5 grid-network is identical to the one used by SCHIC [40]. The communication protocol of SCHIC [40] is implemented as well. SCHIC [40] communicates projected traffic outflows to directly neighbouring intersections to extend the local observation of each controller.

To indicate the performance potential an example is given in case of no communication between intersections (CS0). Performance is reported on the average waiting time. SCHIC-2 (*two machine scheduling*) contains the improvements of SCHIC-nc (new-clustering) implicitly, i.e. SCHIC-2 contains SCHIC-nc and adds a lane-based scheduler. As a result performance metrics are given for "SCHIC-2 over SCHIC" and "SCHIC-2 over SCHIC-nc" in [14]. The improvement of SCHIC-2 over SCHIC grows from 9% to 15%, when the vehicle demand increases from a low to high value. The performance improvement of SCHIC-2 over SCHIC-nc, indicating the performance gain of the new scheduler, grows from 4% to 6% when the vehicle demand increases.

Inter-phase uncertainty omitted
Dang [14] has shown considerable improvements to SCHIC, but omitted inter-phase uncertainty since the only allowed movement at the intersection is the movement for straight-ahead traffic, this can be considered as a large simplification. Dang [14] evaluates intersections which are identical to the ones originally proposed for SCHIC by Xie et al. [27]. By only allowing straight-ahead traffic and scheduling it with two dedicated green-phases, vehicles on each of the four entry-lanes can only be associated with just a single green-phase for straight-ahead traffic. By removing inter-phase uncertainty a commonly found real-world uncertainty is also removed, i.e. inter-phase uncertainty occurs when vehicles which travel towards the intersection on the same road can request different and competing green-phases by changing lanes on this single road. For example by entering the lane dedicated to left-turning traffic, or by deciding to stay on the lane for straight-ahead traffic.

With inter-phase uncertainty removed vehicle detection can be performed at the beginning of the lane, since each vehicle on this lane will request the same phase from the controller. Therefore, the length of each lane is now the only limiting factor to maximize the look-ahead time for vehicle detection. In Dangs [14] 5x5 grid network, vehicles on 20 out of the 25 lanes are are detected at at least 7.5 seconds from the intersection. This look-ahead time is reasonable given that inter-phase uncertainty is eliminated. However, when vehicles are able to request different phases by switching lanes, inter-phase uncertainty is quite an important factor to consider at 7.5 seconds from the intersection.

Paper 2: a more flexible phase set with ERIS by Goldstein and Smith [7]
As mentioned by Manolis et al. [9], a major limitation of SCHIC is found in the restriction which is placed on the phase set. This restriction is as follows, by initial design of SCHIC in 2012 the following restriction is present – each movement (or serviced incoming lane) is only allowed to be present in one of the green phases [1]. This restriction is also explained prior and supported with a figure in section 3.1.1. In this thesis, it is referred to as a non-overlapping phase set. A non-overlapping phase set is visualized in Figure A.2 where each movement (shown as an arrow) is only found in one green-phase.

In 2018, Goldstein and Smith [7] introduced ERIS – Expressive Real-time Intersection Scheduling [7]. ERIS uses a more flexible phase set for lane-based job-scheduling by allowing phase overlap, i.e. each lane can now be serviced in multiple phases. The afore-mentioned method of Dang [14] is still restricted to a non-overlapping phase set. ERIS maintains lane-specific traffic estimates and uses an A* algorithm to evaluate different possible combinations of two compatible lanes in the next green phase.

A dual ring barrier is used by ERIS where any combination two compatible / non-conflicting lanes can receive a green signal. Green signals for turning lanes are allowed to be skipped if no vehicles are present, however straight-ahead lanes are not allowed to be skipped. A visual example of the allowed transitions is shown in Figure 2 of [7].

The A* algorithm allows for real-time decisions by bounding the large search space. A heuristic function estimates the cost of executing future actions. This cost is computed as the expected delay-time of jobs (clusters of vehicles per lane). The heuristic function is based on the earliest start time of jobs. This heuristic function is evaluated using pre-emptive job-scheduling, thus showing resemblance to SCHICs job-scheduling algorithm.

Unfortunately inter-phase uncertainty is omitted in this research, ERIS assumes a 100% adoption rate of communicating vehicles [7]. All vehicles communicate their desired movement (straight-ahead or turning left), speed and location without noise at each second to the traffic controller. ERIS outperforms the commercial version of SCHIC named SURTRAC and a simpler adaptive controller based on time-gaps [7]. Both SURTRAC and the adaptive controller have access to the connected vehicles in a for them suited form as well. However the 100% adoption rate of communicating vehicles can be seen as a perfect information, connected vehicle scenario – in reality this is highly uncommon to date.

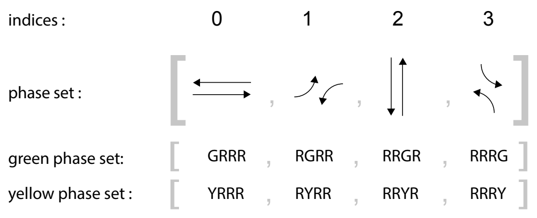


Figure A.2: Default phase set of Simulation of Urban Mobility [19] (SUMO), which is a non-overlapping phase set. Since each movement (indicated with an arrow) is only found in one of the green phases.

Paper 3: poor performance of SCHIC during saturated traffic demands by Hu and Smith [25]
In 2019, Hu and Smith [25] proposed a method which gradually shifts from SCHICs job-scheduling control to queue-based control. SCHIC is based on job-scheduling which works quite well for low vehicle demands. But job-scheduling has been proven to be sub-optimal when the traffic condition reaches saturation. Therefore Hu and Smith [25] proposed to only use SCHICs job-scheduling method for low-vehicle demands.

Appendix B

Algorithms of SCHIC

Chapter 3 has introduced the working principles of SCHIC in a high-level manner. In this appendix we give a more detailed explanation, mostly by means of detailed pseudo-code.

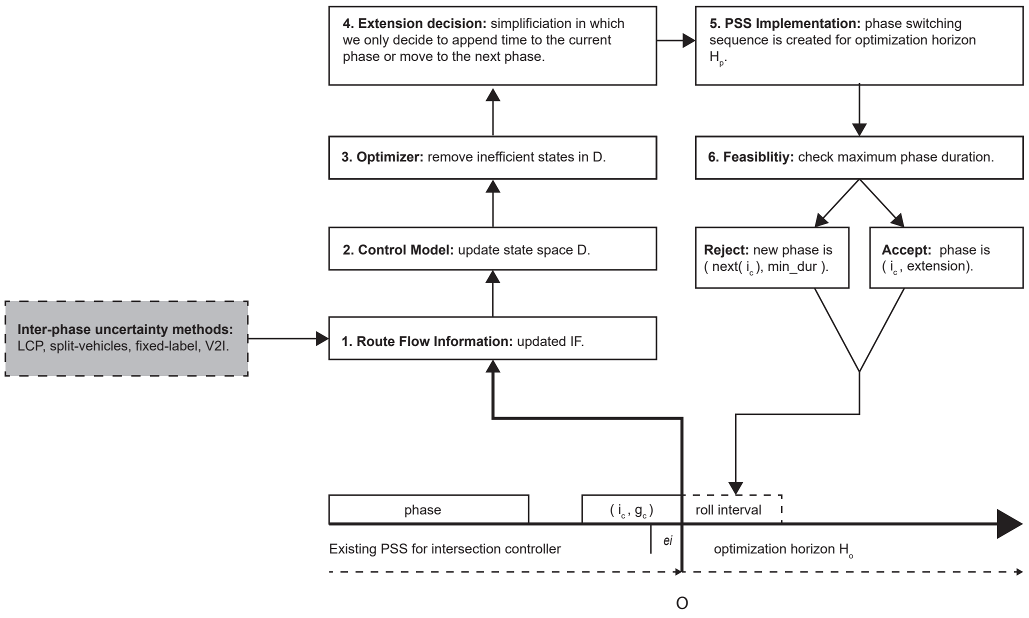

4. Extension decision: simplificiation in which we only decide to append time to the current phase or move to the next phase.
5. PSS Implementation: phase switching sequence is created for optimization horizon H_p.
3. Optimizer: remove inefficient states in D.
6. Feasiblitiy: check maximum phase duration.
2. Control Model: update state space D.
Reject: new phase is (next(i_c), min_dur).
Accept: phase is (i_c , extension).
Inter-phase uncertainty methods: LCP, split-vehicles, fixed-label, V2I.
1. Route Flow Information: updated IF.
phase
(i_c , g_c)
roll interval
ei
Existing PSS for intersection controller
optimization horizon H_o
O

Figure B.1: A repeated figure which is included for clarity of this appendix. Flowchart of SCHICs components, in a high level abstraction. The image is re-created without any intended changes from SCHICs original publication by Xie et al. [1].

Important variables

Some important variables are left as a reference below, they will be explained in the chapter later on.

- *cdt* (int): current decision time
- *cpi* (int) : current phase index
- *cpd* (int): current phase duration
- *phase* (list): a list of light signals (red, yellow or green) for each allowed movement at the intersection
- Δ (float): time resolution in seconds
- D (list): state space used to store different orderings of observed clusters
- *cluster* (list): stores information about arriving vehicles: <$|c|$, arr, dep, dur, fr, road_ratio={entry_road_m: x, entry_road_n: y} >
- $|c|$ (int): number of vehicles in the cluster
- *arr* (int): expected arrival time of the first vehicle
- *dep* (int): expected departure time of the last vehicle
- *dur* (int): duration of the cluster = dep - arr
- fr (float): flow rate of the cluster = $|c|$ / dur
- *road_ratio* (list of vehicle counts per lane): keeps a vehicle count for each entry-lane found in a cluster
- RF (list of clusters): road flows are all observed clusters on an incoming lane
- IF (list of clusters): Inflows are all RFs that can pass the intersection during the same green-phase
- o (list of different variables): observation o is a list of $[RF, IF, cdt, cpi, cpd]$
- th_c (float): threshold time-gap for clustering individual vehicles into one arriving cluster
- *sult* (float): start-up lost time, the time that is lost before a queue of cars reaches the speed limit
- *shw* (float): saturation headway, the average head-way between vehicles in saturated flow

Parameter	Full name	Setting
th_c	threshold clustering	3 seconds
$sult$	start-up lost time	3.5 seconds
shw	saturation headway	2.5 seconds

Table B.1: Fixed parameter settings

Local Traffic Observation

The induction loops have a fixed sampling interval: *samp* [seconds]. The prediction horizon H_p is divided into time-segments where the total segment count = H_p / *samp*. In every time-segment h we have $a^{(h)}$ as the number of vehicles arriving in time-segment h. If $a^{(h)} > 0$: it becomes an arriving cluster, c.

Each arriving cluster c has the following variables. Firstly, $|c|$ (integer) is the number of vehicles in this cluster, arr(c) (integer) is the arrival time of the first vehicle and dep(c) (integer) the departure time of the last vehicle. The flowrate (fr(c)) and duration (dur(c)) can be derived from these variables see Figure 3.2.

Algorithm 3 present the details on insertion and removal of vehicles.

Algorithm 3: insertion and removal of vehicles on road flows

```
1  insert_remove_vehicles(m)
   Input  : m is the lane index associated to movement m
   Output : insertion and removal of vehicles on road flows
2
   // Insertion of new vehicle(s) which arrive during time-segment $h$.
3  if $a^{(h)} > 0$ then
4      c = initialize_an_empty_new_cluster()
5      count(c) = $a^{(h)}$
6      arr(c) = cdt + $H_{ext}$
7      dep(c) = cdt + $H_{ext}$ + samp
8      dur(c) = dep(c) - arr(c)
9      fr(c) = count(c) / dur(c)
10     insert c into $C_{RF,m}$
11     sort $C_{RF,m}$ on increasing arrival time
12 end
   // Removal of departed vehicles.
13 dep_count = get_count_of_departed_vehicles(m)
14 while $dep\_count > 0$ do
15     c = first_cluster_in($C_{RF,m}$)
16     if $dep\_count < count(c)$ then
           // Remove the departed vehicles from the first cluster.
17         count(c) -= dep_count
18         arr(c) -= dep_count / fr(c)
19         dur(c) -= dep_count / fr(c)
20         fr(c) = count(c) / dur(c)
21     else
22         remove c from $C_{RF,m}$
23     end
24     dep_count -= count(c)
25 end
26 return
```

Queues The formation of queues is explained prior in Section 3.1.2.

Anticipated queues As stated prior in Section 3.1.2, here we supply the pseudo-code on the formation of the anticipated queue.

Vehicles which are expected to join the current queue c_q before before it is able to clear are called the anticipated queue. The anticipated queue is added to the current queue cluster, i.e. the current queue c_q will now be extended.

The arrival time and flow rate of the original queue cluster c_q remain unchanged as stated in [1]. Only the vehicle count of the queue cluster increases or decreases and as a result the arrival, departure and duration scale accordingly.

A more detailed explanation is presented in algorithm 4.

Algorithm 4: Anticipated queues on $C_{RF,m}$.

```
1  anticipated_queue($cdt, m$)
   Input  : $cdt$ is current decision time, $m$ is the lane index associated to movement m
   Output : Extend the current queue with anticipated queues on $C_{RF,m}$
2  end_condition = false
3  for $\forall c \in C_{RF,m}$ do
4      if $arr(c) < cdt$ then
           // c is a queue cluster and we need a queue cluster in order to extend it.
5          $c_q$ = c
6          $c^j$ = next($c_q$)
7          while $arr(c^j) \leq dep(c_q)$ and end_condition == false do
8
9              if $dep(c^j) \leq dep(c_q)$ or $fr(c^j) \geq fr(c_q)$ then
                   // Merge $c^j$ into $c_q$, but keep fr of $c_q$
10                 count($c_q$) += count($c^j$)
11                 dep($c_q$) = arr($c_q$) + (count($c_q$) / fr($c_q$))
12                 dur($c_q$) = dep($c_q$) - arr($c_q$)
13                 remove $c^j$ from $C_{RF,m}$
14
15             else
                   // A part or all of $c^{(j)}$ might join $c_q$ and dur is extended with $\delta$dur
                   // sfr is the estimated flow rate of a queue.
16                 $\delta$ dur = (dep($c_q$) - arr($c^j$)) / (1 - (fr($c^j$) / sfr))
17
18                 if $\delta$ dur $\geq$ dur($c^j$) then
                       // Join $c^j$ into $c_q$ as in lines 10 - 13.
19                 else
                       // Only the first part of $c^j$ joins into $c_q$.
20                     vehicles_that_join = count($c^j$) * ($\delta$ dur / dur($c^j$))
21                     count($c_q$) += vehicles_that_join
22                     dep($c_q$) = arr($c_q$) + count($c_q$) / fr($c_q$))
                       // Keep the last part of $c^j$ as its own cluster.
23                     count($c_j$) -= vehicles_that_join
24                     arr($c_j$) += $\delta$ dur
25                     dur($c_j$) = dep($c_j$) - arr($c_j$)
26                     fr($c_j$) = dur($c_j$) / count($c_j$)
                       // The anticipated queue process is terminated after this logic is reached
                          [1].
27                     end_condition = true
28                 end
29             end
30         end
31     end
32 end
33 return
```

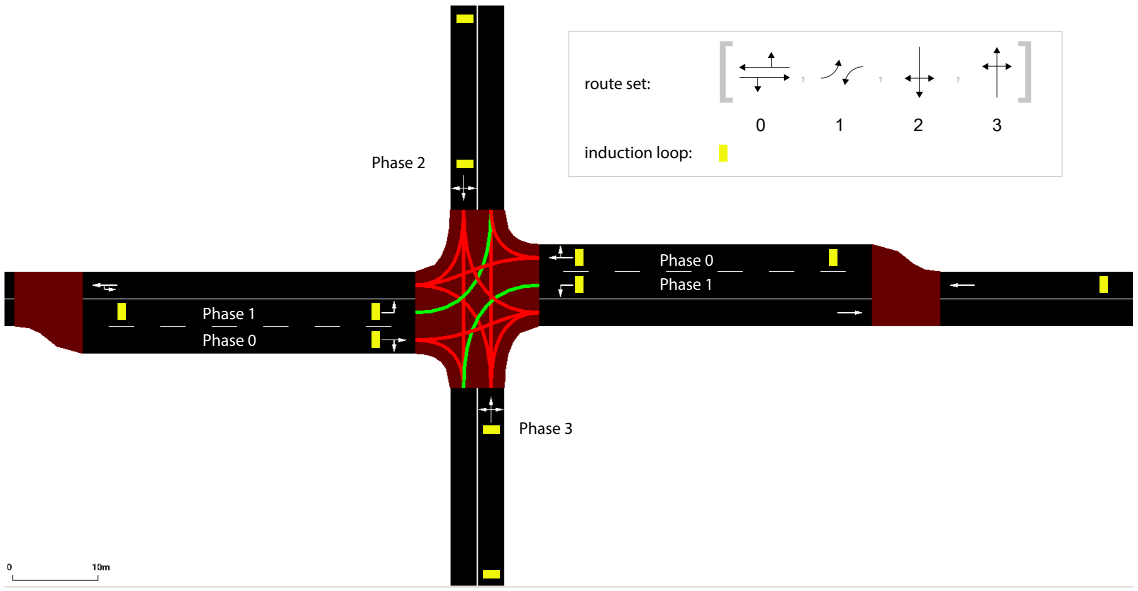


Figure B.2: Some lanes can be serviced at the same time in one phase. A shown on this simple intersection from The Hague.

Inflows (IF)

It is often true that multiple movements are serviced during one green-phase. This is illustrated in Figure B.2.

The actual clusters of interest for the scheduler are the Inflows IF which are based on the available green-phases. $C_{IF,i}$ is a list of all clusters that can pass the intersection safely during green-phase i. I.e. IF contains a list of clusters: $C_{IF,i}$ for each green-phase i. See Figure 3.1.

Assumption 1 *The authors of [1] [27] assume that each lane (or movement) is often serviced in only one green-phase.*

Assumption 1 makes the road to phase mapping $RtoP()$ trivial. $C_{IF,i}$ is a concatenation of all road flows $C_{RF,m}$ that have the right of way during the phase with index i. Formally $IF = (C_{IF,1}, .., C_{IF,|I|})$.

Road_ratio{} keeps count of the number of vehicles that originate from each entry lane. This information is necessary for communication with downstream neighbours.

A cluster is thus extended with road_ratio to the form:

cluster in IF: $<|c|$, arr, dep, dur, fr, road_ratio {entry_lane_m: (int) x, entry_lane_n: (int) y} $>$

Aggregate clustering on IF

The second aggregation technique resembles a feature used by Vehicle Actuated logic (VA) in which vehicles are continuously serviced if they arrive within a critical interval.

Vehicles that are expected to arrive within a threshold time-gap ($th_g = 3$ seconds) are merged together. This resembles the continous servicing found in VA since clusters are treated as non-divisable in SchIC.

The merging procedure is straightforward but described in algorithm 5 below.

Algorithm 5: Threshold-based clustering for each green-phase with index i

```
1  threshold_based(i)
   Input  : $C_{IF,i}$, i.e. the Inflows (IF) where i refers to the green-phase with index $i$ from
            the phase set $I$
   Output : threshold clustering applied to $C_{IF,i}$
2
3  c_last = last_element($C_{IF,i}$)
4  for $\forall c \in C_{IF,i}$ do
5      if $c \neq c\_last$ then
6          c_current = c
7          c_next = next(c_current)
8          time_difference = arr(c_next) - dep(c_current)
9
10         if $time\_difference \leq th_g$ then
               // merge two clusters
11             arr(c_current) = min(arr(c_current), arr(c_next))
12             dep(c_current) = max(dep(c_current), dep(c_next))
13             count(c_current) += count(c_next))
14             dur(c_current) = dep(c_current) - arr(c_current)
15             fr(c_current) = count(c_current) / dur(c_current)
16             remove c_next from $C_{IF,i}$
17         end
18     end
19 end
```

Call graph for clustering The previous algorithms are called in the following order.

1. Update all Road Flows (RF) by adding or removing vehicles on each lane with Algorithm 3
2. Create a single queue on each lane using Algorithm 1
3. Append the estimated queue to the single queue on each lane using Algorithm 4
4. Concatenate multiple RF's into a single IF using $RtoP()$
5. Threshold based clustering on the Inflows (IF) with Algorithm 5

Scheduler

The scheduler creates the previously described state space D (Section 3.2). States that have serviced all observed clusters are called full-states. A full state with minimal cumulative delay is selected to retrieve the schedule S.

Assumption 2 *Clusters in state space D are non-divisible. If we split a cluster to service another we introduce a large delay. The next phase(s) block the currently waiting cars with their green and yellow time.*

Assumption 2 limits the number of operations on D and makes each operation quite simple. The size of the state space is determined by the number of clusters in IF and the number phases in phase set I.

Current observation o

The current observation o is a list that contains $[cdt, cpi, cpd, IF]$. cdt is the current decision time, cpi the current phase index, cpd the duration of the currently active phase and IF the Inflows for the prediction horizon H_p. More details can be found in Section 3.1.2.

Control Flow

The control flow contains all clusters from IF but reordered them. The control flow is defined as: $CF = (S, C_{CF})$. Schedule S is a list of phase indices which determines the ordering of C_{CF}. $C_{CF} = (c_{CF,1}, ..., c_{CF,|S|})$. Each entry: $c_{CF,i}$ is a cluster that can depart during phase i.

Partial schedule

A partial schedule is defined as S_k and contains k clusters of IF, where k is smaller than the total cluster count in IF. The schedule status X is an array which has an entry for every phase. $X = (x_1, .., x_{|I|})$ where each entry in the array: x_i counts the number of clusters from phase i that have currently been serviced. To clarify, with each partial schedule S_k the sum of all entries in its X array is k. The kth cluster in C_{CF} is the x_{s_k}th cluster from C_{IF,s_k}.

State groups

A state group is defined by $\{X, s, s_{prev}\}$, i.e. the X array, the current phase s and the previous green-phase s_{prev}. X alone does not make a state unique. For example, consider the two states with $X = [1, 1]$ in Figure B.3. However, these states do have a different current phase s, which makes them unique.

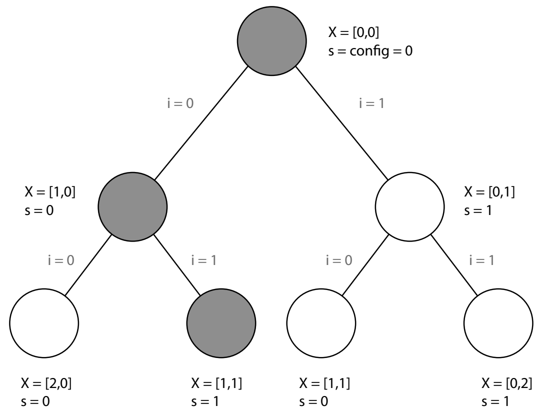


Figure B.3: A repeated figure which is included for clarity of this appendix. The implemented state space D, where states are grouped by $\{X, s, s_{prev}\}$. Without s_{prev} one cannot track back to previous states. The current phase index is s.

Call graph scheduler An overview of the order of calling each algorithm presented below.

1. Forward recursion for calculating all state groups in D using Algorithm 6.
2. Remove inefficient state groups using the dominance criterion with Algorithm 7.
3. Compute the cumulative delay for each partial schedule with Algorithm 8.
4. Track back from the optimal state group to find the schedule S with minimal cumulative delay with Algorithm 9.
5. During runtime: an extension decision is used to determine the new phase duration in Algorithm 10.

Forward recursion with Algorithm 6

Based on observation o, a forward recursion process is used to build state-space D. And hence, to create different partial schedules $S^{(k)}$. Two special X arrays are used: X_{empty} and X_{full} which respectively have: $\forall i : x_i = 0$ and $\forall i : x_i = |C_{IF,i}|$. These arrays indicate the empty and full schedule status. States that have reached X_{full} have serviced all observed clusters in IF. The state space D is initialized with the start node: (X_{empty}, cpi) which has value row: (cpd, cdt, 0,-).

The logic is explained in Algorithm 6 which calls Algorithm 7 to compute the value rows in state space D.

Algorithm 6: forward recursion for calculating all state groups

```
1  forward_recursion_state_groups(o)
   Input    : o is current observation (cdt, cpi, cpd, IF)
   Output   : optimal schedule S
   Require: $(X_{empty}, i_c)$ has one value row (0,0,-,-). So we start from the empty schedule.
2
3  for $k = 1$ to $|IF|$ do
4      $X^{(k)} = \{X : \sum_{i=1}^{|I|} x^{(i)} \equiv k\}$        // Get all unique X arrays that sum to k.
5
6      for $\forall X \in X^k$ do
7          for $\forall s \in [1, |I|]$ do
8              cluster_count = $X_{full}[s]$
9              $x^{(s)} = X[s]$
10             if $x^{(s)} > 0$ and $x^{(s)} \leq$ cluster_count then
11                 update_state_group(X,s)    // Append the $x^{(s)}$th cluster to a partial schedule.
12
13             end
14         end
15     end
16 end
17 return the solution S using Algorithm 9 (by tracking back from the state with $X_{full}$ and
     minimal cumulative delay $d$.)
```

Algorithm 7: calculation of the value row in (X, s)

```
1  update_state_group(X,s)
   Input    : Schedule status X which is reached by adding latest job on route s
   Output   : Updated state space by adding job $x^{(s)}$ to reach schedule status X
   Require: The old state groups $(X_o, s_o)$ $\forall s_o \in [1, |I|]$, where $X_o = X$ with $x^{(s)} = x^{(s)} - 1$.
            Recall, $x^{(s)}$ refers to $X[s]$.
2
3  $X_o = X$ with $x^{(s)} = x^{(s)} - 1$
4  $d_{min} = \infty$
5  $c = c^{(s, x^{(s)})}$
6  for $s_o = 1$ to $|I|$ do
7      for $y_o = 1$ to $|(X_o, s_o)|$ do
           // For each old state found in $D$ we append cluster c on route s.
8          (pd, t, d) = get_cumulative_delay($s_o, pd_o, t_o, d_o$, c) // Compute the finish time and
              cumulative delay for state (X,s) with algorithm 8.
9          Save (pd, t, d, $s_o$) for (X,s)
10     end
       // At this point all possible (X,s) states have been computed.
11     state_manager_greedy(X,s).
        // state_manager_greedy(X,s) collects all (X,s) states in $D$ and it removes all that have
        a longer cumulative delay than the optimal (X,s) state.
12 end
```

Update state groups with Algorithm 8

Each partial schedule S_k has the following state variables: $(X, s, pd, t, d)_k$. Where s and pd are the index and duration of the last phase, t is the finish time of the kth job and d is the cumulative delay for all k jobs.

Updating the state variables of S_k can be done if we know the kth phase index (s_k) and the state variables of S_{k-1}. The schedule status $X_k = (X_{k-1}$ with $x_{s_k} + 1)$.

The input of algorithm 8 is known in algorithm 7. The state variables of S_k are computed using $(X, s, pd, t, d)_{k-1}$ and s_k by calling algorithm 8.

Lines 4,5 ensure that each phase duration is not shorter than G_{min}.

Lines 7 and 14: MinSwitch(s,i) and SwitchBack(i) MinSwitch(s,i) returns the minimum time for switching from phase s to i. Phase skipping is not allowed. It returns the minimum duration (G_{min}) of intervening phases and yellow times between s and i.

Both MinSwitch(i) and SwitchBack(i) only include yellow and minimum green times (G_{min}).

SwitchBack(i) returns the minimum time required for the traffic light to return to phase index i. Switchback(i) = MinCycle - G_{min}

Lines 9, 10 adds the startup lost time *sult* which is needed for clearing the queue.

Line 19 computes the cumulative delay for the kth job.

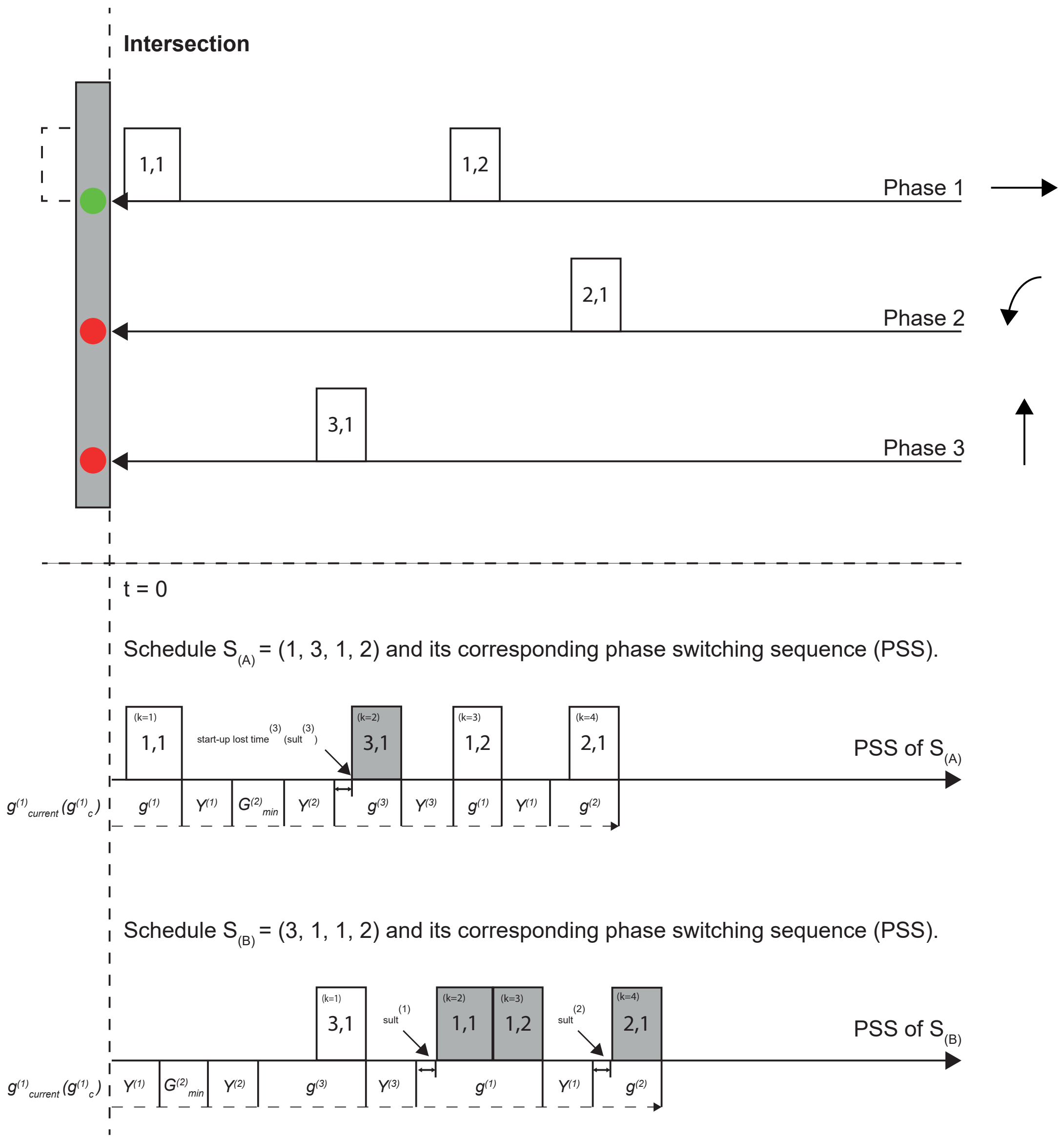
Algorithm 8: Subroutine to calculate (t,d) of partial schedule $S^{(k)}$

```
get_cumulative_delay($s, pd, t, d$, c)
Input  : (s, pd, t, d) of $S^{(k-1)}$, c is the kth cluster to be added
Output : (pd, t,d) of $S^{(k)}$

i = route(c)
if ($s \neq i$) and ($pd < G_{min}$) then
    t = t + ($G_{min}$ - pd)
end
pst = t + MinSwitch(s,i)
ast = max(arr(c), pst)
if ($s \neq i$) and ($pst > arr(c)$) then
    ast = ast + sult // Queue correction:  add startup lost time for clearing the queue.

end
t = ast + dep(c) - arr(c)
if ($s \neq i$) or ($arr(c) - pst > SwitchBack(s)$) then
    pd = t - pst
else
    pd = pd + (t-pst)
end
$\delta d = |c| * max(ast - arr(c), 0)$ // Delay for cluster k.

$d = d + \delta d$
return (pd,t,d) of $S^{(k)}$
```

Elimination Criterion

Theorem 1 *If we have two partial schedules $S^{(k)}_{(A)}$ and $S^{(k)}_{(B)}$ which share the same (X,s). Where (t of $S^{(k)}_{(A)}$) $\leq$ (t of $S^{(k)}_{(B)}$) and (d of $S^{(k)}_{(A)}$) $\leq$ (d of $S^{(k)}_{(B)}$). Then $S^{(k)}_{(B)}$ can be eliminated without loss of optimality [1].*

Figure B.4 shows the Inflows on three routes and two possible schedules $S_{(A)}$ and $S_{(B)}$. At stage k=3 we can remove $S_{(B)}$ since $S^{(3)}_{(A)}$ and $S^{(3)}_{(B)}$ have the same (X,s) but (t,d) of $S^{(3)}_{(B)}$ is larger than (t,d) of $S^{(3)}_{(A)}$. Having the same (X,s) means that the same clusters have been serviced, but in different orderings. The first k clusters of both schedules will be the same as k increases to $|S|$ for a full schedule S. Therefore, the delay will always be larger if we would continue with the inefficient $S^{(3)}_{(B)}$ instead of with the more efficient partial schedule $S^{(3)}_{(A)}$ [1].



Intersection
1,1
1,2
Phase 1
2,1
Phase 2
3,1
Phase 3
t = 0
Schedule S(A) = (1, 3, 1, 2) and its corresponding phase switching sequence (PSS).
(k=1) 1,1
start-up lost time(3) (sult(3))
(k=2) 3,1
(k=3) 1,2
(k=4) 2,1
PSS of S(A)
g(1)current (g(1)c)
g(1) Y(1) G(2)min Y(2) g(3) Y(3) g(1) Y(1) g(2)
Schedule S(B) = (3, 1, 1, 2) and its corresponding phase switching sequence (PSS).
(k=1) 3,1
sult(1)
(k=2) 1,1
(k=3) 1,2
sult(2)
(k=4) 2,1
PSS of S(B)
g(1)current (g(1)c)
Y(1) G(2)min Y(2) g(3) Y(3) g(1) Y(1) g(2)

Figure B.4: *Top half of image:* observed vehicles per individual phase, known as clusters (e.g. for the movement straight-ahead or left-turn). *Lower half of image:* different possible (complete) schedules $S_{(A)}$ and $S_{(B)}$ which (completely) service all observed clusters. Grey rectangles indicate queued vehicles. Among all different schedules, the one with minimal cumulative delay is selected. At stage $k = 3$, both $S_{(A)}$ and $S_{(B)}$ have serviced the same clusters *and* the same phase is active. However, the cumulative delay of $S_{(B)}$ is higher than that of $S_{(A)}$, so schedule $S_{(B)}$ can already be eliminated at stage $k = 3$ – see **Theorem 1.** *(Elimination Criterion)* of [1]. Image reproduced from: Xie et al. [1].

Retrieving the solution S using Algortihm 9

The optimal schedule S is obtained when the state space D is fully built. The state space is fully built if it contains at least one full state. A full state is a state which serviced all clusters in IF.

Algorithm 9 retrieves the optimal the schedule S from the full state with minimal cumulative delay d. By tracking back to the root node, the phase durations pd are saved which have been stored in each state.

The control flow: C_{CF} and corresponding phase durations $PD = (pd_1, ..., pd_{|S|})$ are saved until they are replaced in the next scheduling iteration.

Algorithm 9: Retrieve the solution S

```
1  get_optimal_schedule()
   Input    : State space D.
   Output   : Optimal schedule S.
   Require: State space D needs to have reached depth $|IF|$.
2
   // Select the full-state (X,s) with minimal cumulative delay d.
3  $X = X_{full}$
4  $s = arg\ min_s\ d^{X,s}$
5
   // Track back from the full state (X,s) with minimal cumulative delay.
6  for $k = |IF|$ to 1 do
7      $s_k = s$
       // Obtain the phase ($s_o$) of the previous state, by tracking back to the previous state.
8      $s = s_o\ of\ (X_o, s_o)\ with\ X_o\ having\ x_{s_k} = x_{s_k} - 1$
9  end
10 return $S = (s_1, .., s_{|S|})$
```

Extension decision

Section 3.4 in [1] states that only the first cluster (c_1) of the computed schedule S is used. The current phase is extended until the first cluster (c_1) is expected to have departed. An updated schedule S is obtained after each update of the traffic observation, since clusters are non-divisible, the first cluster (c_1) will be fully finished even when its arrival at the intersection is delayed. See Figure B.1 for a graphical overview.

The first cluster of *schedule* S, c_1, is obtained from C_{CF} and is used for the extension decision, see Algorithm 10.

Algorithm 10: Extension decision

```
1  extension_decision()
   Input    : Current observation $o$.
   Output   : Extension decision.
2
3  if $|S| \equiv 0$ or $s_1 \neq cpi$ or $arr(c_1) \geq SwitchBack(cpi)$ then
4      ext = 0
5  else
6      ext = min(dep($c_1$) - cdt)
7  end
8  return ext
```

Line 4: when $ext = 0$ the next green phase, and its preceding yellow phase, are selected to run for their minimum durations.

Line 6: If $ext > 0$ the current phase duration (cpd) is extended by ext. A violation of the maximum green time is repaired when it occurs. If the current phase duration (cpd) reaches the maximum green time duration, i.e. $cpd = G_{max}$, the current green phase is terminated and the next green phase, and preceding yellow phase, are selected to run for their minimum durations.

Appendix C

Quantitative comparison on published networks

As referred to in Section 3.3, this appendix contains a quantitative comparison between the published results on SCHIC by Xie et al. [1] and the results as obtained with the re-implementation of SCHIC.

The first paper on SCHIC [1] considers two ideal networks, i.e. an isolated intersection and an arterial network, to evaluate the performance of several traffic control algorithms. The isolated intersection demonstrates intersection optimization, the arterial network also incorporates implicit coordination between neighbouring intersections.

Both scenarios can be seen as ideal since each controller uses two phases, where each entry-lane is serviced in one phase. Hence there is no inter -or intra-phase uncertainty for the traffic light controller. All lanes are identical in length (L) and the speed limit is set to $10m/s$. The default signal timing constraints are set for each green (G) and yellow (Y) phase, $G_{min} = 5s$, $G_{max} = 55s$, $Y = 5s$.

In this thesis both networks and traffic scenarios are recreated to compare the performance of the re-implementation with the metrics published in [1]. Unfortunately the original authors have refused to supply the original traffic scenarios, which makes this comparison incomplete. However, it does give an indication whether or not the algorithms are correctly re-implemented.

Traffic scenarios are recreated in the following manner, hourly traffic demands are converted into traffic scenarios with OD2TRIPS [1]. In this thesis, all results are averaged over 10 independent 1-hour simulations for each hourly demand. The default Krauss car-following model of SUMO is used with its default parameter settings[2] , see Table C.1.

Parameter	**Description**	**Value**
accel	Normal (non-emergency) acceleration ability	2.9 m/s^2
decel	Normal (non-emergency) deceleration ability	7.5 m/s^2
maxSpeed	Maximum speed	10.0 m/s
minGap	Minimum gap when standing to leader vehicle	2.5 m
length	Vehicle length	4.3 m
sigma	Driver imperfection with range [0,1] (0 denotes perfect driving)	0.5

Table C.1: Default parameter settings for Krauss driver model during the quantitative comparison.

Comparative traffic controllers
Different responsive controllers are compared to SCHIC in [1], namely vehicle-actuated logic (VA) [20] and a dynamic programming method called Controlled Optimization of Phases (COP) [5]. Lastly, a fixed coordination plan (FIX) is used for the arterial network. The time offsets between the fixed time controllers ensure coordination given that the traffic scenarios do not change over time.

In this thesis the following two controllers are used:

1. VA, which is implemented as a **Time-Gap Controller.**
2. FIX, a fixed time-plan with offsets.

[1]`https://sumo.dlr.de/docs/OD2TRIPS.html`
[2]`https://sumo.dlr.de/docs/Vehicle_Type_Parameter_Defaults.html`

Isolated intersection

The isolated intersection demonstrates intersection optimization since no network level effect is included. Two entry-lanes with a length of $L = 750m$ cross here, the advance detectors are placed at $L_{det} = 700m$ from the intersection which results in a $70s$ look-ahead horizon $(L_{det}/v_{freeflow})$. While the length of these lanes could resemble a rural intersection, a prediction horizon of $70s$ seems uncommonly long in reality. This prediction horizon of $70s$ might enhance the performance of SCHICs job-scheduling algorithm, however it is only possible since there is no inter-phase uncertainty (i.e. each lane is serviced in one phase only).

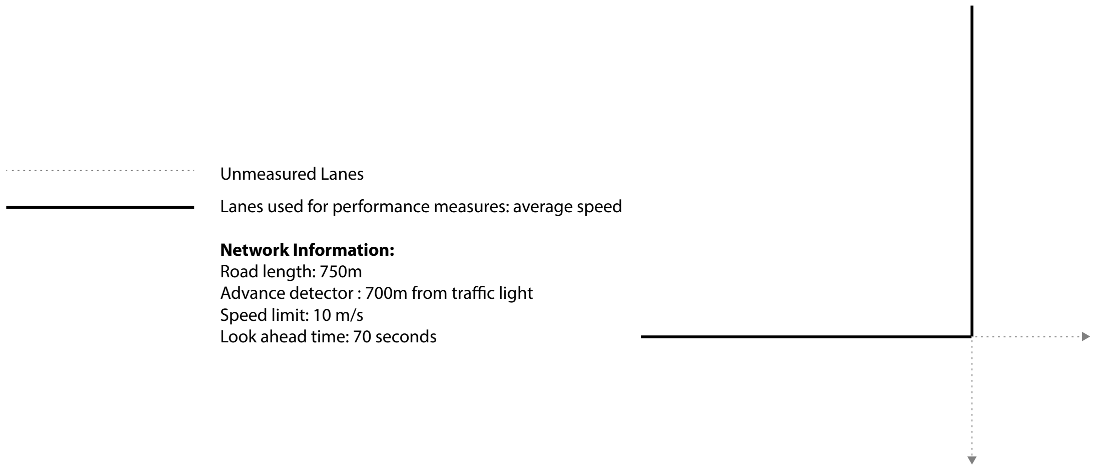
The original SCHIC paper [1] compares two controllers, namely Vehicle Actuated Control (VA) and SCHIC. The performance metric is the average speed on the two incoming lanes. The traffic scenarios are described by an hourly demand, where the demand proportions on both lanes shift every 20 minutes in the following manner: {0.3;0.7}, {0.5;0.5}, {0.7;0.3}. In each run the simulation period is 1 hour, the reproduced runs are evaluated on 10 independent 1 hour simulations for each hourly demand.

Figure C.2 shows the reported performance in the original paper on SCHIC [1] on the left-side and the reproduced performance of the re-implementation on the right-side. Both SCHIC controllers (blue lines) show a close resemblance up until 900 vehicles per hour. The re-implementation of SCHIC shows a much higher performance between 900 and 1200 vehicles per hour. The reported results show a significant performance drop for both SCHIC and VA after 900 vehicles per hour. Both reproduced controllers, SCHIC and VA, do not show this steep performance drop after 900 vehicles per hour. Therefore the performance difference is likely the result of differences in the reproduced traffic demands.

Vehicle Actuated Control (VA) is shown twice since the advance detector placement is not specified in [1]. Here, the two used advance detector placements are $2sec$ and $70sec$ from the intersection, $2sec$ is the default-setting of SUMO. With VA, a green phase is prolonged as long as vehicles are arrive within a time-gap of 3.0 seconds at the detection point. VA with a look-ahead of $70sec$ shows little resemblance to the reported values in [1], which could be due to the fact that $70sec$ is an uncommonly long look-ahead time for VA control.

On the contrary to the reported performance in [1], the default VA controller outperforms SCHIC in the reproduced runs for vehicle demands above 600 vehicles per hour. A possible reason can be found in the robustness of a VA controller, the controller continuously services a phase if vehicles arrive at the detection point within a time-gap that is below a threshold (set to $3sec$). SCHIC determines an arrival time for each vehicle $70sec$ before it is able to reach the intersection. The VA control with a short look-ahead time of $2sec$ cannot look ahead as far as the SCHIC controller, but given the higher average speed it seems to be more responsive to disturbances or uncertainties in traffic flow compared to SCHIC.

In conclusion, the performance of both SCHIC controllers is similar enough to assume that the algorithms are re-implemented correctly.



Unmeasured Lanes
Lanes used for performance measures: average speed
Network Information:
Road length: 750m
Advance detector : 700m from traffic light
Speed limit: 10 m/s
Look ahead time: 70 seconds

Figure C.1: Isolated intersection scenario.

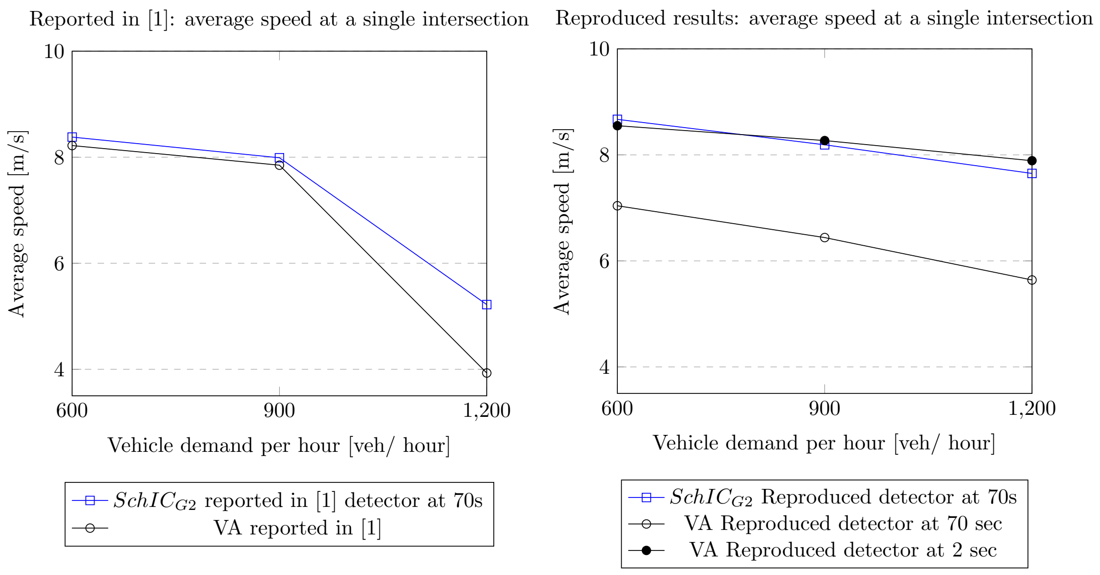


Figure C.2: Comparison of reported and reproduced results. Average speed on an isolated intersection.

Arterial network

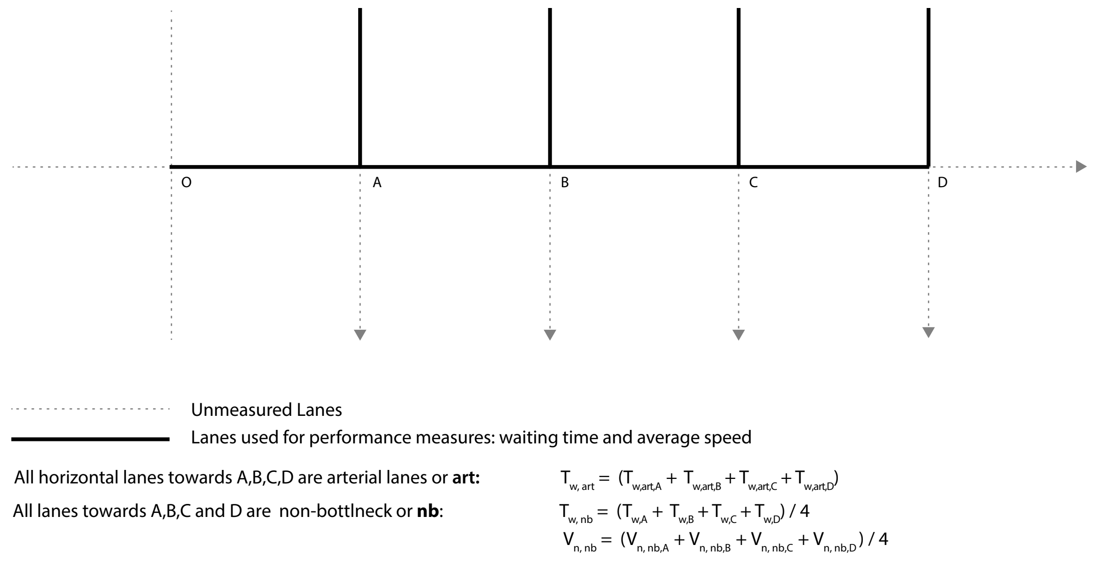

O
A
B
C
D
Unmeasured Lanes
Lanes used for performance measures: waiting time and average speed
All horizontal lanes towards A,B,C,D are arterial lanes or **art:**
$T_{w, art} = (T_{w,art,A} + T_{w,art,B} + T_{w,art,C} + T_{w,art,D})$
All lanes towards A,B,C and D are non-bottlneck or **nb**:
$T_{w, nb} = (T_{w,A} + T_{w,B} + T_{w,C} + T_{w,D}) / 4$
$V_{n, nb} = (V_{n, nb,A} + V_{n, nb,B} + V_{n, nb,C} + V_{n, nb,D}) / 4$

Figure C.3: Recreated arterial network based on [1].

The primary function of arterial roads is to deliver a high throughput between popular points of interest, for example between a city centre and a highway. The arterial road, i.e. main horizontal road, often intersects lower capacity roads which can join the arterial road. Coordination between the intersections is an important tool to deliver high throughput on the arterial road.

The arterial network as shown in Figure C.3 contains a bottleneck intersection O and four intersections $\{A, B, C, D\}$ which will be used to evaluate different controllers. Intersection O uses a fixed time plan, with $35sec$ of green time for the (horizontal) arterial road in a total cycle of $70sec$. The fixed time control limits the number of vehicles that can travel towards intersection A. Intersection O can be seen as the boundary of the control zone.

Several controllers, a Fixed Time Plan (FIX), VA and SCHIC, are evaluated on intersections $\{A, B, C, D\}$. The Fixed Time Plan uses a green split of $\{43sec, 17sec\}$, respectively for the arterial -and vertical road. Each intersection uses an offset of $28sec$ to its neighbour. Xie [1] obtained these parameters through a local search to maximize the average speed on the arterial lane, given that no turns occur in the traffic scenarios. All lanes are $250m$ in length (L) with advance detectors placed at $100m$, resulting in a look-ahead time of $10s$ for the two adaptive controllers: VA and SCHIC.

Performance metrics are only reported for the entry-lanes of intersections $\{A, B, C, D\}$. Figure C.3 shows all entry-lanes in solid black lines and will be referred to as $non-bottleneck$-lanes or nb in short, only the horizontal lanes are referred to as $arterial$-lanes or art.

Traffic is generated for the entire network based on hourly demands which vary from $\{300, 600, .., 1500\}$ vehicles per hour. The hourly demand is distributed among the roads in the following manner, $\frac{7}{16}$ on the arterial road and $\frac{1}{16}$ on each of the four side roads. The remaining fraction of $\frac{5}{16}$ traverses intersection O on the vertical road, but does not affect the performance of the other intersections. All traffic is considered to continue straight-ahead. In this thesis, the performance metrics for each hourly demand is averaged over 10 independent simulations of 1 hour.

Comparison of reported and reproduced metrics

The reported -and reproduced performance metrics will be compared here. Metrics used for comparison are the average speed on non-bottleneck entry-lanes ($v_{n,nb}$) as well as the average waiting times on the non-bottleneck -and arterial lanes ($T_{w,art}, T_{w,nb}$). Waiting times are obtained from SUMO, and defined as the total time that each vehicle has a speed below 0.1 m/s.

For FIX, average waiting times are shown in Table C.2. Most noticeable is $T_{w,art} = 0$ as obtained in [1] which is not realized in the reproduced scenarios. A possible reason for this difference could be that the used offset of $28sec$ and a green split of $\{43sec, 17sec\}$ does not fit the reproduced traffic scenario. Given that driver properties like driver-imperfection are not exactly the same as in [1]. A local search based on the reproduced traffic scenarios and vehicle properties could yield an offset and green split ratio which does result in $T_{w,art} = 0$.

On all non-bottleneck lanes combined the differences between the reported and reproduced results are smaller, see $T_{w,nb}$ in Table C.2. At 1500 vehicles per hour the performance difference increases heavily, which might be a result of the previously mentioned misfit in the used offset.

The lower performance of the reproduced fixed time controller is also visible in the average speed on all non-bottleneck lanes, see Figure C.4. The reported graph demonstrates a horizontal performance for fixed time control, whereas the reproduced results show that the fixed time controller becomes gradually worse. In conclusion, small changes to the behaviour of drivers are likely to worsen the results of a fixed time controller.

For VA average waiting times are shown in Table C.3. Two detector placements are used in the reproduced results, since the detector placement is not specified by [1]. Similar results are found when the detector is placed at 2 sec from the stop-line of the intersection. The results of VA Reported [1] and VA Reproduced 2 sec vary little on both reported waiting times: $T_{w,art}$ and $T_{w,nb}$. This is likely to indicate that the reproduced traffic scenarios are quite similar to the ones in use by [1], given that the controller adapts to drivers behaviour in real-time.

For SCHIC, average waiting times are shown in Table C.4. SCHIC Reproduced does not obtain the low waiting times on $T_{w,art}$ as reported in [1]. Large values of $T_{w,art}$ are however also seen for FIX and VA controllers when comparing the Reported and Reproduced values. SCHICs reproduced values of $T_{w,art}$ are considerably lower than the ones reproduced with VA control. This indicates that SCHICs control is advantageous over VA control in this scenario.

$T_{w,nb}$ is comparable between SCHIC Reported and SCHIC Reproduced. These waiting-times, $T_{w,nb}$, consider the waiting time of all vehicles on SCHICs entry-lanes (intersections $\{A, B, C, D\}$). SCHIC Reproduced shows the lowest waiting times when compared to FIX Reproduced and VA Reproduced. These low values indicate that the optimization for all vehicles behaves as expected.

In conclusion, the algorithms of SCHIC seem to be re-implemented correctly.

	300 veh		600		900		1200		1500	
	$T_{w,art}$	$T_{w,nb}$	$T_{w,art}$	$T_{w,nb}$	$T_{w,art}$	$T_{w,nb}$	$T_{w,art}$	$T_{w,nb}$	$T_{w,art}$	$T_{w,nb}$
FIX Reported [1]	0.00	5.96	0.00	6.04	0.00	6.15	0.00	6.07	0.00	6.41
FIX Reproduced	0.76	7.16	1.14	7.19	1.94	7.81	3.59	8.99	5.69	10.32

Table C.2: Performance of Fixed Time Plan (FIX) reported in [1] compared to FIX Reproduced

	300 veh		600		900		1200		1500	
	$T_{w,art}$	$T_{w,nb}$	$T_{w,art}$	$T_{w,nb}$	$T_{w,art}$	$T_{w,nb}$	$T_{w,art}$	$T_{w,nb}$	$T_{w,art}$	$T_{w,nb}$
VA Reported [1]	6.84	5.33	7.13	5.97	7.15	6.68	7.27	7.52	7.67	8.43
VA Reproduced 2 sec	7.27	5.87	8.06	6.56	7.32	6.42	5.84	6.01	5.41	6.56
VA Reproduced 10 sec	16.43	11.9	23.19	16.31	26.78	19.04	27.37	19.96	23.85	18.86

Table C.3: Performance of Vehicle Actuated Control (VA) reported in [1] and reproduced performance which is given for the default detector placement at $2sec$ and at $10sec$.

	300 veh		600		900		1200		1500	
	$T_{w,art}$	$T_{w,nb}$	$T_{w,art}$	$T_{w,nb}$	$T_{w,art}$	$T_{w,nb}$	$T_{w,art}$	$T_{w,nb}$	$T_{w,art}$	$T_{w,nb}$
SCHIC Reported [1]	0.26	0.57	0.35	1.23	0.34	2.05	0.27	2.90	0.29	3.71
SCHIC Reproduced	0.6	0.74	0.63	1.18	0.52	1.69	0.5	2.46	0.69	3.2

Table C.4: Performance of SCHIC as reported in [1] and the re-implementation of SCHIC

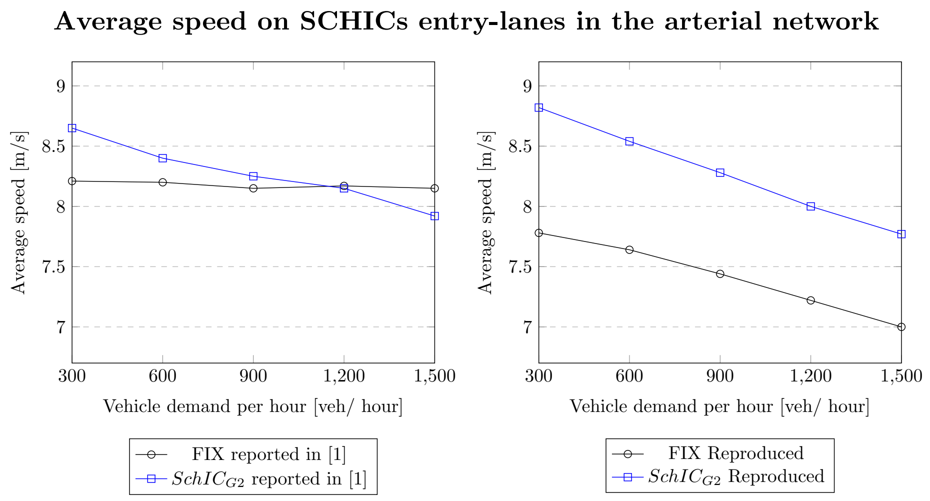


Figure C.4: Average speed on SCHICs entry lanes, i.e. non-bottleneck (nb), for the arterial network.

Appendix D

Feature importance

Here, we evaluate the performance of our predictive models without a set of derived features. Secondly, we plot the relative importance of each individual feature.

Removing acceleration and derived features

In Section 5.3.6 we removed the acceleration based features, due to the expected noise on their measurement. Here, we remove the acceleration-based -and derived features.

The derived features, Speed Change and Relative Speed to a leading vehicle, are derived from other features from the feature-vector. Since these features are derived, they might contain duplicate information and could therefore be obsolete or un-used by the predictive models. Therefore, we observe if they are actually useful for the predictive models by obtaining the accuracy without the following four features.

1. Leading Vehicle Acceleration.
2. Relative Acceleration to a leading vehicle.
3. Speed Change.
4. Relative Speed to a leading vehicle.

The results are shown in Table D.1 and Table D.2.

When including all features, the Random Forest achieved the following accuracies during both turning-movement distributions, i.e. *[0.5, 0.5]* and *[0.2, 0.8]*, respectively $IDM_{acc} = 0.81$ *and* 0.89 and $Krauss_{acc} = 0.51$ *and* 0.80.

It can be concluded that the accuracies of the Random Forests are slightly lower, but remain comparable when removing these features.

Turning Movement distribution	Fixed label	Probability	SVM	Decision Tree	Random Forest
$[50, 50]$	0.50	0.51	0.61	0.74	0.74
$[20, 80]$	0.80	0.67	0.83	0.86	0.86

Table D.1: Accuracies for IDM driver model, without the following features: Leading Vehicle Acceleration, Relative Acceleration to a leading vehicle, Speed Change and Relative Speed to a leading vehicle.

Turning Movement distribution	Fixed label	Probability	SVM	Decision Tree	Random Forest
$[50, 50]$	0.49	0.50	0.49	0.52	0.51
$[20, 80]$	0.80	0.67	0.80	0.78	0.80

Table D.2: Accuracies for the Krauss driver model, without the following features: Leading Vehicle Acceleration, Relative Acceleration to a leading vehicle, Speed Change and Relative Speed to a leading vehicle.

Feature importance graphs

Figure D.1 and Figure D.2 show the individual features and their importance for the Lane Change Prediction models, respectively for the IDM and Krauss driver-model. All features are listed on the Y-axis and their importance is shown on the X-axis.

More information on all features is given in Section 4.2.

In conclusion, we can assume that the acceleration based features are not perceived as important to the models. The *speed_change* features, are however perceived as quite important by the models.

Acceleration based features

We focus here on the possibly noisy acceleration features. The features *speed_change_1*, *speed_change_2* and *speed_change_3* can be considered acceleration based features. To clarify, in order to compute each $speed\_change_i$ the measured speeds of each vehicle before the lane-split are used: e.g. $speed\_change\_1 = \frac{speed\_2}{speed\_1}$.

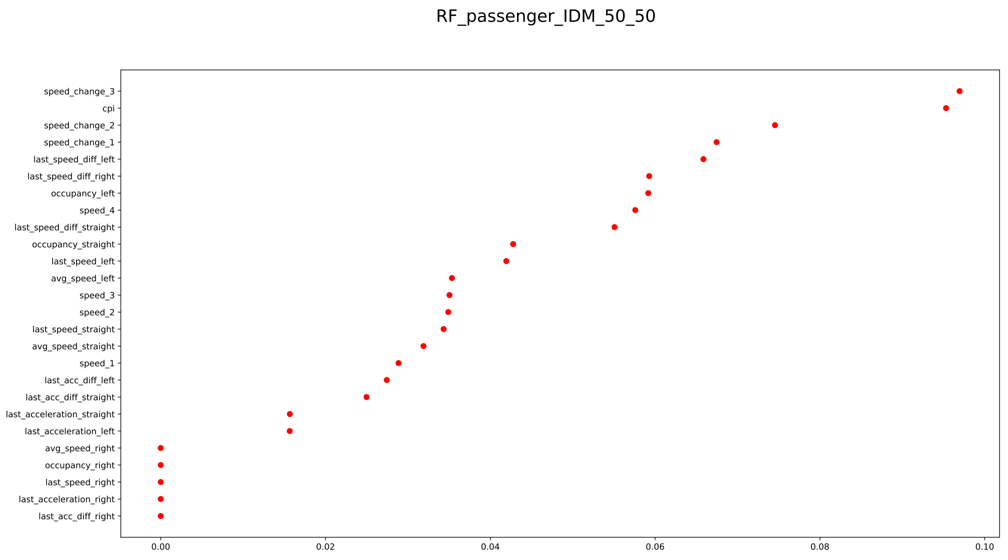
With the Krauss driver-model the observed speeds before the lane-split , e.g. *speed_2* and *speed_1*, are noisy which causes *speed_change_1* to be noisy as well. Nonetheless, the predictive models for both driver-models (Krauss and IDM) seem to see value in these possibly noisy $speed\_change_i$ features since they are among the most important features.

Besides these $speed\_change_i$ features, there are other acceleration based features. These acceleration-based features are sometimes important, but are overall not among the most important features, as described below.

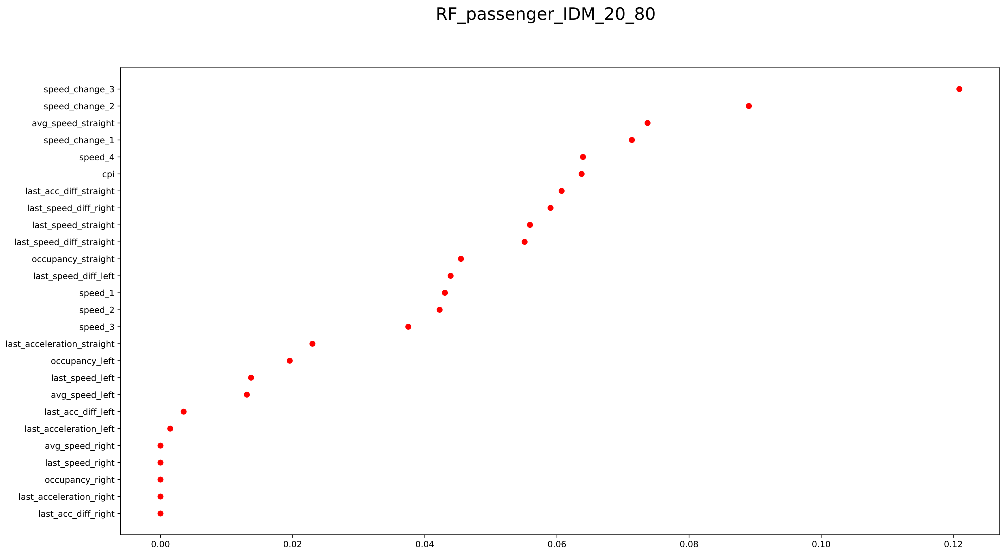
Intelligent Driver Model (IDM) - Figure D.1
With IDM in the balanced [50,50] case, the first occurrence of an acceleration feature is at the 18th

place and out of all features they are the least important. With IDM in the un-balanced [20,80] case, the first acceleration feature is found at the 7th place, in this case the other acceleration features are almost all among the least important features.

Krauss Driver Model - Figure D.2
With Krauss in the balanced [50,50] case, the acceleration based features are found mid-way in the feature importance ranking at place 11 to 15. As for the unbalanced [20,80] case, the first acceleration feature is found at place 8, again the other acceleration features are almost all among the least important features.

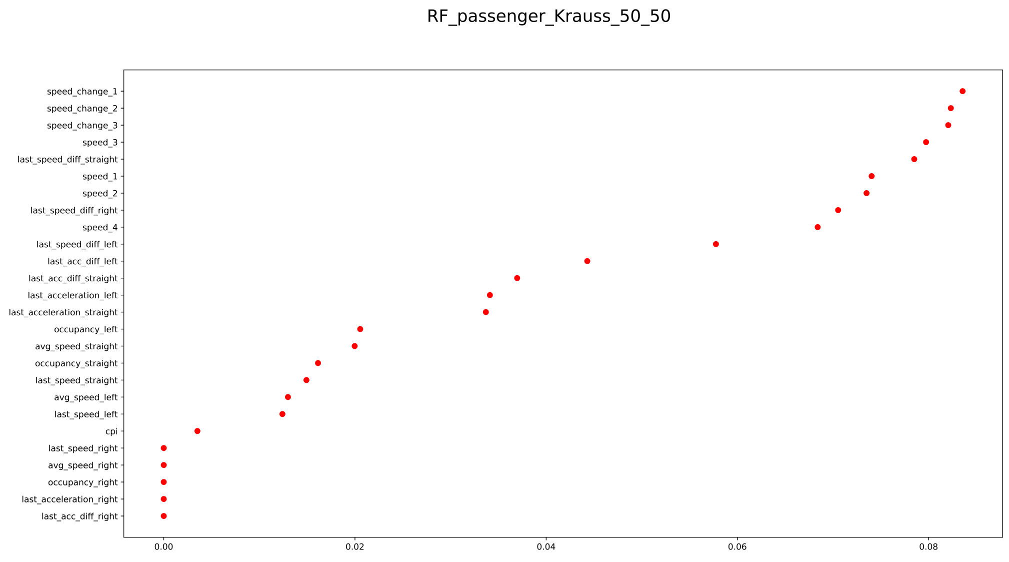


(a) [50,50] turning movement distribution

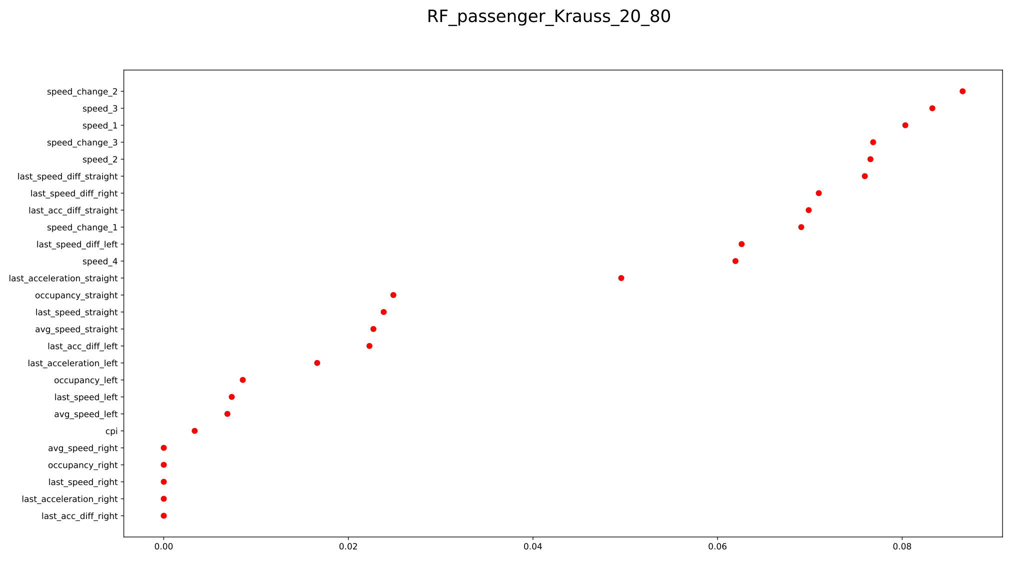


(b) [20,80] turning movement distribution

Figure D.1: Feature importances for Random Forests of the IDM driver-model. Y-axis contains individual features, X-axis the relative importance of an individual feature.



(a) [50,50] turning movement distribution



(b) [20,80] turning movement distribution

Figure D.2: Feature importances for Random Forests of the Krauss driver-model. Y-axis contains individual features, X-axis the relative importance of an individual feature.