# Automated modelling of traffic signal controller behaviour from floating car data using Hidden semi-Markov Models

Paving the way for large scale implementation of Green Light Optimal Speed Advisory systems

MSc. Thesis

B.B.C. Harleman

# Automated modelling of traffic signal controller behaviour from floating car data using Hidden semi-Markov Models

Paving the way for large scale deployment of Green Light Optimal Speed Advisory systems

by

B.B.C. Harleman

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Whenever, 2021 at Whatever time CET.

| | | |
|---|---|---|
| Student number: | 4361105 | |
| Project duration: | June 11 , 2020 – August 30, 2021 | |
| Thesis committee: | Dr. ir. A. Hegyi, | TU Delft CEG, supervisor |
| | Dr. ir. J. Kober, | TU Delft 3ME, supervisor |
| | Dr. A. Dabiri, | TU Delft 3ME |

An electronic version of this thesis is available at http://repository.tudelft.nl/.

# Acknowledgements

First of all I would like to thank Andreas for inspiring me with his lectures and for providing me with the opportunity to write a thesis on a topic that suits my interests more than I ever hoped for. Also, I am grateful for the kind and empathic way in which you have guided me throughout the project.

Secondly, I want to thank Jens for making the effort to immerse himself in the subject of traffic signals and for his sharp analyses of my work. His feedback always lead to meaningful insights, which, combined with his relaxed and humorous way of communicating, meant that I would always leave our meetings replenished with both energy and new ideas.

Also, I have to thank my parents for their limitless support. I can not express how blessed I feel to know that I can always count on them to back me up in any way that I need.

Lastly, I want to thank Emma for being personally responsible for making every day of the past year a happy day, no matter how dim and hopeless the rest of the world sometimes seemed.

B.B.C. Harleman
Son, August 2021

# Abstract

Research has shown that GLOSA systems can help reduce the emission of $CO_2$ by motor vehicles and improve flow on urban road networks. However, existing GLOSA systems only work in combination with a limited selection of Traffic Signal Controllers (TSCs) and therefore have not been widely implemented. This research describes and evaluates the design of a system to model the behaviour of the most common types of TSCs using data that is shared by road users, also known as Floating Car Data (FCD). Almost every road user is able to share his location using a smartphone, so this means that a system of this kind can be implemented everywhere in the world where people are willing to participate, without requiring any changes to the infrastructure. It is chosen to model TSC behaviour using Hidden Markov Models (HMMs), because they can be generated from unlabelled data, as opposed to other machine learning techniques. More specifically, Explicit Duration Hidden semi-Markov Models (EDHSMMs) are used, so that timing behaviour can be captured more accurately than with standard HMMS. An adaptation to an existing learning algorithm is made to decrease amount of data required for learning and to make sure the algorithm can cope with the sparse nature of FCD. Additionally, a system is developed to combine generally known intersection data and FCD to generate models that have transition parameters with low entropy and thus high predictive powers. In this way, the system produces models that are accurate enough to be used for the generation of GLOSA from relatively sparse data. In a simulated setting, it is shown that the system can identify the true switching behaviour of the most common types of TSCs and that it produces models whose duration probability distributions converge to the true situation when provided with realistic amounts of data.

# Contents

# Glossary

$D$ Maximum state duration.

$M$ Number of possible observations.

$N$ Number of model-states.

$O$ Full observation sequence from time-step $t = 1$ to $t = T$.

$Q$ Markov chain (state sequence) resulting from the HMM for observation sequence $O$.

$S$ Set of size $N$ containing all possible states.

$T$ Length of observation sequence in time-steps.

$V$ Set of size $M$ containing all possible observations.

$\tau_t$ Remaining duration of $q_t$ in number of time-steps.

$\mathbf{A}$ transition probability matrix.

$\mathbf{B}$ observation emission probability matrix.

$\mathbf{D}$ duration probability matrix.

$a_{ij}$ Element $(i, j)$ of $\mathbf{A}$: probability of transition from state $s_i$ to state $s_j$.

$b_{ik}$ Element $(i, k)$ of $\mathbf{B}$: probability that state $s_i$ emits observation $v_k$.

$d$ State duration.

$n_{sig}$ Number of traffic signals at modelled intersection.

$o_t$ Observation at time-step $t$.

$p_{i,d}$ Element $(i, d)$ of $\mathbf{D}$: probability that state $s_i$ lasts $d$ time-steps.

$q_t$ State at time-step $t$.

$s_i$ State i from the set $S$ $(i = 1:N)$.

$v_k$ Observation k from the set $V$ $(k = 1:M)$.

$\pi_i$ Probability that state $s_i$ is state $q_1$ $(i = 1:N)$.

$\mathbf{\Lambda}$ Set of all model parameters $\{\mathbf{\Pi}, \mathbf{A}, \mathbf{B}, \mathbf{D}\}$.

$\mathbf{\Pi}$ Vector of length $N$ containing the probabilities for each state of being state $q_1$.

# Acronyms

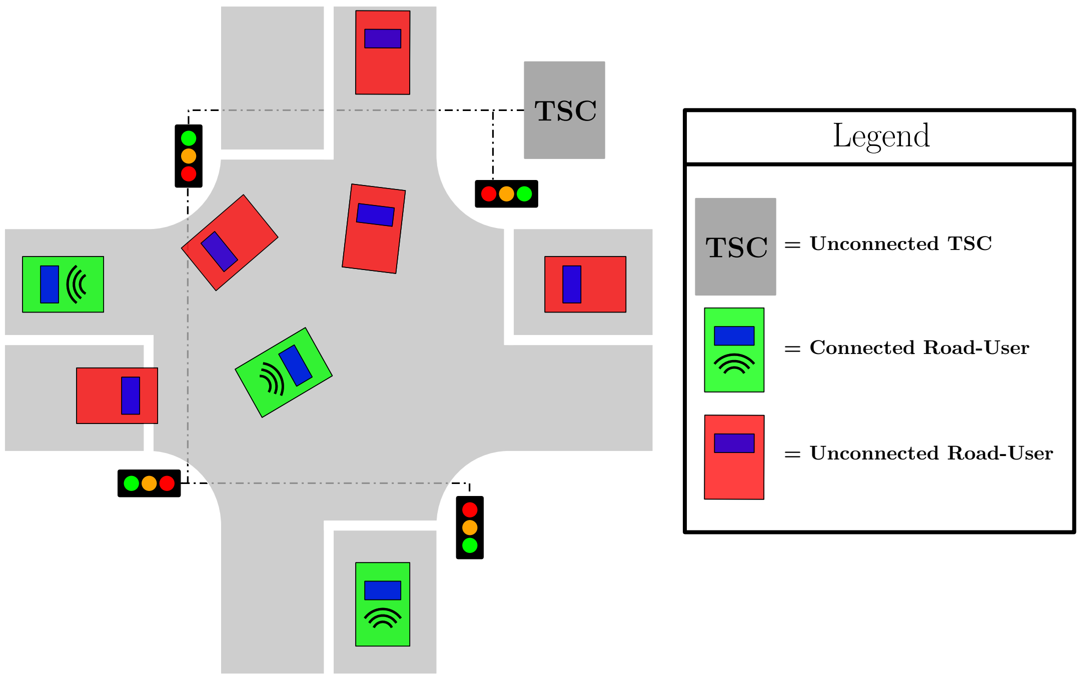
| | |
|---|---|
| **FCD** | Floating Car Data. |
| **FTTSC** | Fixed-Time TSC. |
| **GLOSA** | Green Light Optimal Speed Advice. |
| **HMM** | Hidden Markov Model. |
| **HSMM** | Hidden semi-Markov Model. |
| **TSC** | Traffic Signal Controller. |
| **VATSC** | Vehicle-Actuated TSC. |

# 1 Introduction

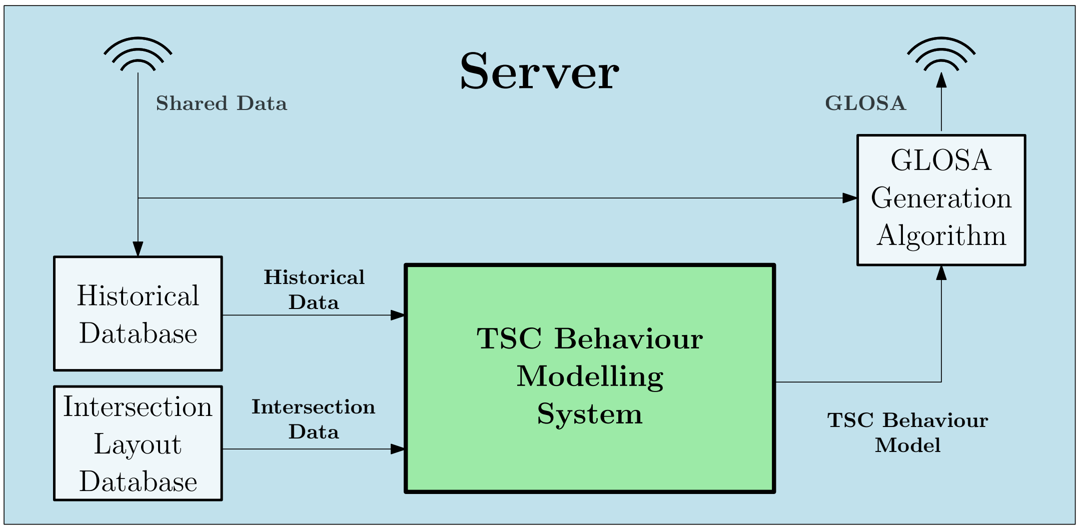
## 1.1. Motivation

As a result of urbanization and growing welfare, the pressure on urban road networks has been increasing over the past decades, leading to more congestion and pollution. Hence, solutions such as Green Light Optimal Speed Advice (GLOSA) systems are researched to make more efficient use of the available roads in urban environments. GLOSA systems aim to improve traffic flow and minimize energy usage by advising road-users on the speed they need to travel at in order to catch a green light without stopping. By doing so, braking and acceleration actions can be minimized, which reduces harmful emissions and increases traffic flow at intersections. This is because vehicle emissions are highest under acceleration and capacity drop negatively correlates with velocity, meaning that queue discharge rates are higher if the queue is moving at a higher velocity [1]. Although empirical relationships between velocity and queue discharge rates have thus far only been established for highway traffic, it can be assumed that this is also the case for urban traffic. Several GLOSA systems have been developed around infrastructure that broadcasts the current state of traffic signals and time to green using Signal Phase and Timing messages [2–11]. Although these systems promise to yield up to 22% reduction in $CO_2$ emissions, they require new hardware to be installed with estimated costs of roughly $50,000 per intersection [12]. Therefore it will take a long time and considerable funds before these connected signals become the norm. In the Netherlands, which has one of the most modern road networks in the world, there are roughly 5600 Traffic Signal Controllers (TSCs), of which only 707 are able to generate Signal Phase and Timing messages at the time of writing [13]. At the current rate of installation, even traffic in countries with modern road networks will be controlled by TSCs without communication possibilities for at least the next decade. During this decade, the world is facing a turning point when it comes to climate change, so there is a direct need for methods to lower the emission of greenhouse gasses. This provides an incentive to research GLOSA systems that rely on a different, more readily available source of data. Developments in the area of communication technologies mean that vehicles and people in general are becoming more connected. There are already more than twice as many people with a smartphone worldwide than there are cars, meaning that a large portion of road-users is expected to be able to share their location in the form of Floating Car Data (FCD). Therefore, a system to model the phasing and timing behaviour of common TSCs from FCD is developed in this research, with the goal to pave the way to large scale implementation of GLOSA in the near future.

Figure 1.1 sketches the context of the system developed in this research, which is indicated in green in Figure 1.1b. It can be seen that some road-users share FCD with a server, which in turn provides the connected road-users with GLOSA. The GLOSA is generated based on a TSC-behaviour model. Designing a system to generate these models from historical FCD and generally-available map data is the goal of this research.



(a) Simplified representation of the assumed traffic situation. Both connected and unconnected road-users travel over a signalized intersection. The connected road-users share their data wirelessly with the server depicted in Figure 1.1b, which in turn provides them with GLOSA if it has enough data to do so for this intersection and the current traffic situation.



(b) Overview of the system to provide connected road-users with GLOSA, without communicating with the TSC. The TSC Behaviour Modelling System is highlighted in green, because the development of this system is the goal of this thesis research.

Figure 1.1: (a) shows a schematic representation of the traffic situation for which the system depicted in (b) provides GLOSA.

Several attempts have been made towards developing GLOSA systems by using shared data in the form of FCD, but only for fixed-time TSCs [14–17]. Although fixed-time signals are common some parts of the world, 85% of traffic signals in the Netherlands are vehicle-actuated [18]. Therefore it is desirable that the developed system can also model the behaviour of vehicle-actuated TSCs, which have the ability to change the combination of signals that is set to green based on the current presence of vehicles.

### 1.1.1. Motivation for using HMMs

Many modelling techniques could be considered to create the system described above, of which Hidden Markov Models (HMMs) [19] are used for this research. Over the years, several researchers have investigated using some form of Markovian model to describe the behaviour of TSCs [20–22]. The most recent research assumes that the current state of the traffic signals at an intersection is communicated by the TSC and that its control behaviour is known in the form of a transition probability matrix like the ones used in HMMs [22]. However, historical TSC behaviour data is not generally available, so constructing such a transition matrix is not a straightforward task and is also not described in the research. Still, there are four reasons why HMMs seem particularly suited as the basis of a system to model TSC behaviour from FCD.

Firstly, HMMs are specifically intended to infer hidden information about the system that they model [19]. Therefore, it can be expected that a well learned HMM of TSC behaviour can imply the colour of signals, even at moments when these are not observed. This means HMMs can potentially be used to infer the correct colour of all signals at an intersection using real-time FCD, even though FCD typically can only provide explicit information about the colour of a portion of the signals at a given time [14, 17, 23, 24].

Secondly, the parameters of a HMM can be learned using unlabelled data [19], which saves the labour intensive step of labelling data manually. For the context of this research, FCD therefore only has to be converted into observations of the colour of individual signals, but the true state of the TSC does not have to be known, as is required for some existing TSC behaviour modelling methods [25, 26], or general Deep Learning approaches [27]. Automated methods have been developed already to execute this conversion [14, 23, 24].

Thirdly, HMMs are generative models, meaning that they can be used to generate all possible future state sequences from a certain starting point and indicate the probability of each of them [19]. This means that HMMs can be used to predict future TSC behaviour if the current state is known. Additionally, because the probabilities of each future state sequence can be determined explicitly, it can be chosen to only broadcast a speed advice if a certain probability threshold has been exceeded. A speed advice generation method that explicitly takes this uncertainty into account has already been developed [22].

Lastly, because it can be easily interpreted what each parameter of a HMM represents, prior knowledge of the system that is being modelled or certain boundary conditions can be easily included in initial models. This potentially means that little data is needed to learn relatively complex models. Likewise, features of the modelled system can be extracted from learned models. For example, the minimum and maximum green time of each signal at an intersection could be extracted from a HMM of its behaviour, depending on the way the states of this HMM are defined. This information on its own can be used to make rough predictions of the behaviour of the modelled TSC at times when little to no real-time data is available.

## 1.2. Problem statement

As described in Section 1.1, reducing emission of greenhouse gasses is one of humanity's greatest challenges in the near future. GLOSA systems have the potential to contribute to this reduction, but so far they either rely on expensive hardware, or only work for a limited number of TSCs. To make GLOSA available for as many intersections as possible in the near future, a system must be developed to model the behaviour of the most common TSCs, including vehicle-actuated TSCs, using FCD. It must be ensured that the system requires as little data as possible, while generating models with sufficient predictive power to generate GLOSA, i.e., the resulting models must be parsimonious. The modelling system has to be able to cope with the sparse nature of FCD. Section 1.1.1 has explained that HMMs have features which make them an attractive candidate to perform the described modelling of TSC behaviour. The system should therefore generate HMMs. Because not much research has been conducted on the modelling of TSCs with HMMs, it must be focused on laying the foundations of the modelling system and evaluating the data quantities needed to generate accurate models. From these findings, it should become clear which developments are necessary before the system can be used in practice.

## 1.3. Research approach

The main contribution of this thesis research is the design of a Hidden Markov modelling method. This method is designed so that it can model the phasing and timing behaviour of common TSCs, including those that are traffic-actuated. While the phasing behaviour of TSCs can be captured by the transition matrix of general HMMs, they are limited in their ability to model duration behaviour. Therefore, to more accurately model the TSC timing behaviour, Hidden semi-Markov Models(HSMMs) [28] are used, which explicitly model the duration probability distributions of all model states.

An important aspect of the design of the modelling method is finding a trade-off between generating models that are simple enough to be learned using realistic quantities of FCD, while being flexible enough to accurately model true-TSC behaviour. To this end, an existing algorithm to learn the parameters of Hidden semi-Markov Models (HSMMs) [28] is adapted so that instead of one, it can also process none or multiple observations at a single time-step. By doing so, models that are a combination of a Dynamic Naive Bayesian Classifier (DNBC) [29] and a HSMM are obtained. It will be explained why this approach yields models with far fewer parameters, without them losing predictive power, which means that they are more parsimonious.

This document is structured as follows; first, Chapter 2 presents relevant information and research from the fields of traffic signal control, Hidden Markov modelling and their combination. Then, Chapter 3 condenses this information into a design problem and solves it by describing the design of a TSC behaviour modelling system. This design is evaluated by performing the experiments described in Chapter 4, which also analyses their results. Chapter 5 concludes the research and presents recommendations for future work.

# 2 Background

This section introduces background information on the state of the art of fields that are relevant for the design of the system described in Section 1.2. First, Section 2.1 presents the most common approaches to control signalized intersections. This information is later used in Section 3.2 to describe which features of TSCs are modelled and how. Second, Section 2.2 summarizes the research on GLOSA-systems and what prevents them from being implemented on a large scale and thereby identifies the research gap that this thesis aims to fill. Section 2.3 then describes which sources of shared traffic data exist and what information about TSC behaviour can be extracted from them. Together, these three sections describe the system that needs to be modelled, what the output of the modelling system is used for and what its input is respectively. Based on this information, Section 3.1 formulates the goals of the modelling system and under which assumptions it is developed.

Section 2.4 explains the basic principles and challenges of modelling with HMMs. Also, the field of information theory is introduced, which is commonly used to compare models during model generation and selection processes. Section 2.5 subsequently describes Hidden semi-Markov Models (HSMMs) and presents the details of a Forward Backward algorithm that is used to learn the parameters of HSMMs.

Finally, Section 2.6 concludes the findings and summarizes the elements that are important for the development of the methodology in Chapter 3.

## 2.1. Signalized intersection control

Traffic signals ensure safe and efficient usage of road sections that are used by multiple traffic streams, by giving cyclic right of way to conflicting streams. The colour of each individual signal is controlled by a central TSC, which can be programmed with various strategies. The most important factor that influences the control strategy is the traffic demand on the individual traffic streams, i.e., how many road-users are entering and exiting the intersection in each direction per unit of time. Additionally, availability of road-user detectors or the funds to install them determines whether a control type that reacts to the current presence of road users can be used. Other factors such as the control of neighbouring intersections, necessity to favour certain traffic streams or to facilitate public transport influence how traffic signal controllers are programmed.

All in all, this means that a bespoke signal controller is designed for each signalized intersection. Therefore, modelling the behaviour of a signal controller whose true control tactics are unknown is not a straightforward task. When this behaviour can only be observed indirectly, partially and irregularly by analysing FCD it becomes even more complex. However, all TSCs must adhere to general rules and most are designed on the same principles, so knowledge of these principles can provide some welcome guidance for this modelling task. This section therefore introduces the most common types of TSCs and describes the basics of their control structure.

Because the terminology differs between countries and researchers, it is first defined how terms are used in this document.

### 2.1.1. Terminology

In literature, some common terms have different meaning from one work to another. To avoid confusion, this section therefore introduces the terminology regarding intersections as it will be used in this research.

**Traffic stream:** a flow of road-users with a particular heading

**Direction:** a traffic stream towards or away from an intersection

**Movement:** a traffic stream that connects two directions on an intersection. Movements are referred to by a number, according to the Dutch standard, which is explained in Appendix A

**Signal group:** one or more traffic signals that control one or more movements. One signal group consists of multiple signals if there are multiple lanes for the same movement. Multiple movements are controlled by one signal if they share a lane. In this case the signal group is numbered by the through going movement as explained in Appendix A

**Realisation:** turning a signal group to green

**Conflict group:** a set of movements of which all members have a conflict with all other members

**No-conflict group:** a set of movements of which none of the members has a conflict with any of the other members

**Maximum conflict group:** the conflict group with the largest number of members for a particular intersection

**Maximal no-conflict group:** a no-conflict group to which no other movement of the intersection can be added without it ceasing to be a no-conflict group

**Phase:** a set of movements that is realised simultaneously. Under the assumption that conflicting directions can not have a green signal simultaneously, this is always a no-conflict group

**Stage:** a maximal no-conflict group that is part of the control structure of the TSC

**TSC-state:** the combination of factors that define the current control output of the TSC. This includes the currently active stage, its past duration, detector activations and their timing

**TSC phasing behaviour:** The collection of phases that can be realised by the TSC and their possible ordering. If multiple phases can follow a certain phase, the transition probability to each of the possible successors is part of the phasing behaviour. All factors that influence which phase is realised are part of the phasing behaviour, e.g., when a phase is realised every other cycle, or is only realised when a particular combination of detectors is activated.

**TSC timing behaviour:** The duration probability function of each phase that can be realised by the TSC. All factors that influence the duration of realisations are part of the duration behaviour

**Clearance time:** time from the moment that a signal controlling a certain movement is set to red until the moment that the signal controlling a conflicting direction is set to green. Clearance times are calculated such that conflict areas are cleared by traffic from one movement before traffic from a different movement can be expected to reach the conflict area

### 2.1.2. Control structure

The control structure of a TSC defines which phases it can realise and in which order, i.e., what the stages of the TSC are [30]. Which phases make up the stages and in which order they are realized depends on the intersection layout and its traffic situation. Generally, maximal no-conflict groups are used as stages, so that as many signals are green as possible during each stage. For most TSCs, the number of stages is equal to the size of the maximum conflict group [18, 30]. This is also the minimum number of stages, because each movement must be part of at least one stage and by definition, the members of the maximum conflict group can not be part of the same stage. In rare situations, it can be better to implement a control structure with more stages, for example if there is an incentive to not realise certain movements as often as others. Figure 2.1 shows an example of a flow diagram of a TSC for an intersection with movements 1 to 12. The figure suggests that there is no overlap between the stages. This can generally be assumed to be true, because conflicting directions can never have a green light simultaneously. However, there is one exception to this, because some countries, such as Germany and Switzerland, allow a negative clearance time [30]. Sometimes it can be expected from the intersection's layout that traffic from one movement will clear a conflict area before traffic from another movement can reasonably be expected to reach that area even if the signal for the first movement is switched to red after the signal for the second movement is switched to green. The two movements can then have a simultaneous green signal for a short period of time. However, this is not common practice and in the Netherlands this is not allowed [30].

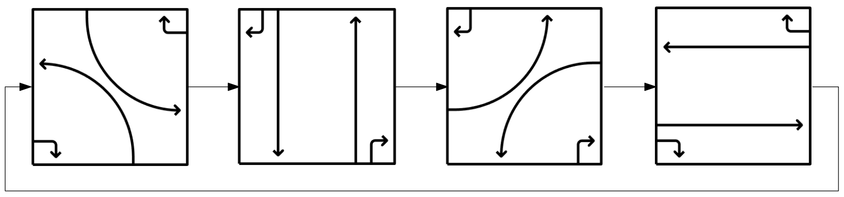


Figure 2.1: Example of a flow diagram of a TSC with four stages, controlling the signals of an intersection with twelve movements. Only movements with a green signal are shown.

### 2.1.3. Types of TSCs

Multiple types of TSCs exist, each with different limitations to their phasing and timing behaviour. This section introduces the types of TSC that make up the majority of those installed. The main focus is on locally controlled intersections, but some well known controllers that optimize the control for road networks as a whole will also be mentioned.

**Fixed-time controllers** The simplest type are Fixed-Time TSCs (FTTSCs), also known as interval control or pre-timed control, which have a control scheme where all stages have a pre-set duration. This type of controller does not require sensors to measure the current traffic situation. However, if the intersection contains traffic signals for active transport modes, these are usually equipped with detectors such that their signals are included in the cycle only if there is a request. Although records are not kept, it can be expected that FTTSCs are the most common type of traffic signal globally, because they are cheap to install and maintain. Another advantage of FTTSCs is that they can easily be coordinated with other traffic signals if the placement of intersections relative to eachother allows it [30]. Urban areas in the USA are particularly suited for coordinating traffic signals, because the streets in these areas are build in a block pattern, with roughly equal spacing between intersections. This layout makes it possible to create green waves in both directions. In situations where intersections are not equally spaced, it can still be possible to coordinate traffic signals in one direction. It is then usually chosen to switch the direction of coordination depending on the time of day, to account for rush hour traffic mostly moving into, or out of the city. A commonly used tool to optimize the control of coordinated traffic signals is TRANSYT [31], which can calculate the optimal fixed-time control strategy offline for a given network and traffic load.

A drawback of FTTSCs is that they are programmed for one particular traffic situation. This means that if the true traffic situation differs from that for which the controller is programmed, road-users will experience unnecessary delays. Especially when there is little traffic it can occur that road-users have to wait for a red signal lasting almost a complete cycle, while no other traffic is using the intersection. This leads to frustration, which in turn can lead to red light running, causing potentially dangerous situations.

**Vehicle-Actuated controllers** Vehicle-Actuated TSCs (VATSCs), also called phase control [30], can overcome the drawbacks of FTTSCs by adapting their control action to the current presence of road users. This requires detectors to be installed at the intersection, which makes VATSCs more expensive to install and maintain. However, these controllers can guarantee efficient use of the intersection under more traffic conditions than FTTSCs can.

Amongst VATSCs, there are two main types. Firstly, semi-actuated is used to control intersections where two or more traffic streams cross, of which one has a considerably higher traffic load [30, 32]. The movements belonging to this traffic stream are set to green by default, which is called waiting in green, and are only set to red when traffic is detected for one of the minor flows. This means that only the minor flows need to be equipped with detectors. Secondly, if all movements are equipped with detectors, the intersection can be controlled using fully-actuated control. With this type of control, waiting in green can also be implemented [18]. Therefore, semi-actuated control can be seen as a sub-type of fully-actuated control, so for remainder of this document, all actuated controllers are referred to as VATSC.

VATSCs can operate with varying levels of adaptivity. Simpler types only vary the duration of stages, but not their order and composition. In Germany, this type is often combined with a fixed cycle time, hence most of the GLOSA research conducted by German researchers is focussed on this type of control [15, 17, 20, 33].

A type of VATSC that is common in North America is the ring and barrier approach [32], an example of which is shown in Figure 2.2. The controller can realise combinations of phases from different rings that are between the same barriers. Once a signal group has been switched to red, it has to wait for the next cycle before it can be realised again [32]. For the example from Figure 2.2, this results in the control structure shown in Figure 2.3.

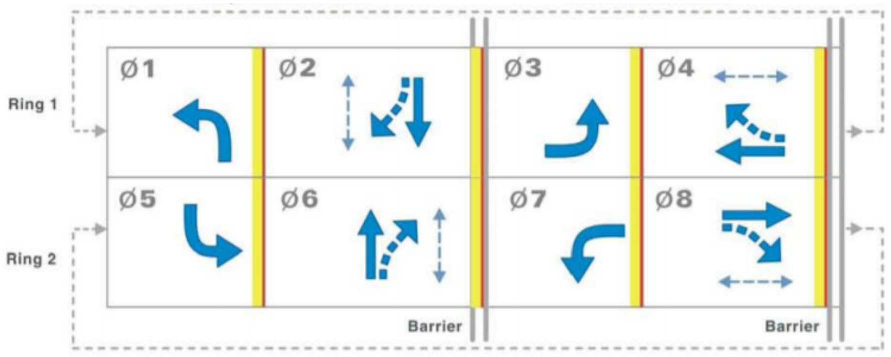


Figure 2.2: Example of a ring and barrier control system [32]. Only movements with a green signal are shown. Permitted movements are indicated with a dashed arrow, which means that they have a conflict with another movement, but that this conflict is permitted. Pedestrian movements are indicated with a double-headed arrow. Between barriers, phases of different rings can be realised simultaneously.

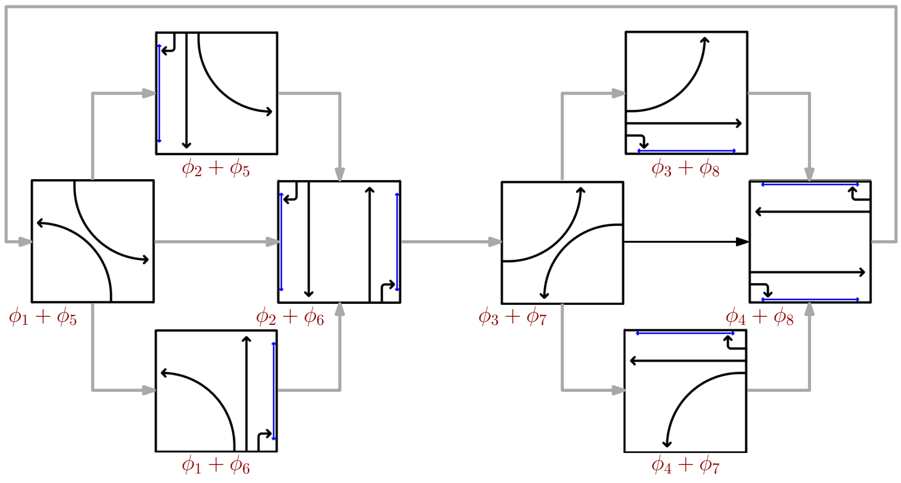


Figure 2.3: Control structure belonging to the ring and barrier controller shown in Figure 2.2. The text underneath the blocks indicates which phases from Figure 2.2 are combined in that block. Blue double-headed arrows indicate pedestrian movements.

VATSCs with even higher adaptivity are common in the Netherlands. These do not have a fixed cycle time and can adapt their control action to the current traffic demand by deviating from their stages. This is called flexibility if movements from the next stage are realised in combination with movements from current stage [18]. If a movement from a stage that is not the next stage is realised with the current stage, this is called an alternative realisation [18]. Flexibility or alternative realisations can take place if some movements from the current stage do not have a request, but a movement from a different stage that does not conflict with the currently realised phase does have a request. To facilitate this, signals of the current stage only get a green signal if they have a request, meaning that the realised phase can differ from the current stage. Figure 2.4 shows a flow diagram of a VATSC with flexibility. As stated in Section 1.1, this type of controller makes up 85% of the TSCs in the Netherlands [18]. When comparing Figure 2.3 and Figure 2.4 it is clear that these types of VATSC are similar for the examples given here. The ring and barrier type can be seen as a sub type of a VATSC with flexibility, because the latter can realise more different combinations of movements.

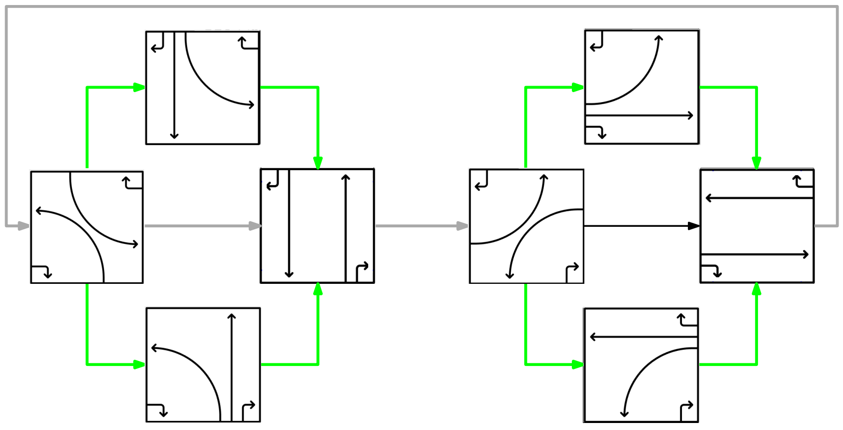


Figure 2.4: Example of a control structure for a TSC with four stages and flexibility. Only movements with a green signal are shown. It can be seen that the controller can deviate from the main stages by realising a flexibility phase, which is a non-conflicting combination of the movements of two consecutive stages. Transitions between stages are indicated by grey arrows. Transitions to and from flexibility phases are indicated by green arrows.

Lastly, VATSCs exist that are structure free, meaning that they do not follow a control scheme with stages, but instead can realise any sequence of phases, based on what best suits the current presence of road users. However, these types of controllers are rare.

**Traffic-Adaptive controllers** Traffic-adaptive TSCs optimize the control of multiple signalized intersections in a network based on the current traffic situation in the network. The most used traffic-adaptive systems are SCATS [34] and SCOOT [35]. Roughly 250 of these systems have been installed worldwide, which covers about 70.000 intersections [36].

These TSCs coordinate the phases of neighbouring intersections. However, the difference with fixed-time approaches, such as TRANSYT, is that the timing of phases can vary from cycle to cycle. This means that the behaviour of signals controlled by these TSCs is similar to that of signals controlled by a VATSC without flexibility. The only difference is that the timing of traffic-adaptive controllers is based on the traffic conditions in the whole network, instead of on the traffic conditions in the direct surroundings of the intersection.

It is important to note here that providing road-users with GLOSA in an area with a network optimized signal control system could disrupt the workings of this system. The optimization algorithms of traffic-adaptive TSCs namely assume that road users drive at roughly the speed limit between intersections, but this is no longer the case when road users follow speed advice. Therefore, it can be expected that the efficiency of the network as a whole decreases, which directly opposes the intention of both traffic-adaptive TSCs and GLOSA.

### 2.1.4. Time variance of control

Because traffic loads vary over the course of the day or week, TSCs are programmed to behave differently at different times. For example, during the morning rush-hour timing and phasing can be different from that during the evening rush-hour. This means that TSCs can be seen as time-variant systems. Due to changing traffic loads, this time-variance also occurs for VATSCs even if their programming does not change over the course of the day. However, if TSCs are observed during a short enough period, the traffic load and TSC behaviour can be seen as time-invariant. When modelling TSC behaviour, it is therefore common practice to create different models for the same intersection. Researchers have investigated how many models are needed to describe all situations [37, 38]. They concluded that traffic conditions and TSC behaviour can be divided into eleven groups of days, as can be seen in Table 2.1. Days that belong to the same group were found to have similar traffic loads at similar times. To account for changing traffic conditions during the course of the day, each type of day is subdivided into fifteen minute long segments. To model a single intersection for each possible traffic situation, it is therefore necessary to construct $11 * \frac{24}{0.25} = 1056$ different models according to these researchers. Although it seems unrealistic to model TSC behaviour if it requires learning this many models, it is important to realize that some of the groups from Table 2.1 represent a much larger fraction of the year than others. Additionally, some time-frames within these groups capture many more traffic movements than others. Therefore, by creating only a couple of models, representing the busiest situations on the most common days, it is possible to model a large fraction of all traffic movements. It can thus be expected that a significant portion of road users can receive GLOSA from only a few models. Additionally, it can be expected that it is easier to create models for periods with higher traffic loads for two reasons. Firstly, a higher traffic load means that more road-users are interacting with the TSC. If it is assumed that data is shared by road-users, this means that there is more data available for busier periods. Secondly, it can be assumed that the behaviour of VATSCs becomes more predictable under higher traffic loads, because it will approach the behaviour of a FTTSC. This can be explained as follows: VATSCs are in fact deterministic systems, but because they adapt to traffic, which behaves in a stochastic way, their behaviour appears stochastic. Each unique combination of current TSC-state and detector activation will lead to a unique control action. Under high traffic loads however, it can be assumed that all detectors are activated continuously, which leads to a continuously repeating cycle of the same control actions.

| | |
|---|---|
| 1: Monday (regular) | 2: Monday (vacation) |
| 3: Tuesday-Thursday (regular) | 4: Tuesday-Thursday (vacation) |
| 5: Friday (regular) | 6: Friday (vacation) |
| 7: Saturday (regular) | 8: Saturday (vacation) |
| 9: Sunday (regular) | 10: Sunday (vacation) |
| 11: Holiday | |

Table 2.1: Types of day with unique traffic loads and TSC behaviour, according to [37, 38]

Researchers have also investigated methods to identify when TSCs change their control action [15, 39]. However, these methods only work for FTTSCs.

## 2.2. Green Light Optimal Speed Advice systems

The main reason to model TSC behaviour is because these models can be used to provide road-users with GLOSA. This section gives an overview of the research that has been done on GLOSA. As stated in Section 1.1, GLOSA systems can greatly reduce vehicle emissions and potentially improve the capacity of existing road networks by improving traffic flow.

The first GLOSA system was developed in 1989 by Volkswagen [40]. It was no immediate success, so the research was put to rest until the late 2000's [2, 20, 41]. However, GLOSA is still not generally available to all road-users on all intersections. This is mainly because only a few research projects have resulted in real-world implementation of a GLOSA system and all these examples only work within a limited area [2, 9, 10, 42]. This is either because the system only works in combination with TSCs that can communicate with road-users, or because the system only works for a particular type of TSC. Additionally, because these systems are often developed in cooperation with vehicle manufacturers, they can only be used by people driving relatively new vehicles from those manufacturers [43]. It was shown that the positive effects of GLOSA are more pronounced if a higher percentage of road-users has access to it and if it is available at more intersections [4]. As stated above, no methods have thus-far been developed that can provide a substantial portion of road-users with GLOSA on a substantial portion of all intersections.

Therefore it is necessary that a method is developed that can cost-effectively provide all road-users that are willing to participate with GLOSA, for as many intersections as possible. This method would have to work with the most common TSCs and without relying on communication with TSCs. Therefore, data would have to come from different sources. The most obvious source is shared GPS-data, also known as FCD. The potential of using this type of data for GLOSA systems has been investigated by several researchers [14–17]. It was shown that even with low penetration rates (<1%), the switching behaviour of FTTSCs could be predicted with accuracy that makes it usable for GLOSA. However, as described in Section 2.1, FTTSCs are uncommon in some parts of the world, so to also be able to predict the behaviour of more common VATSCs, a new method is needed. In current literature, there are two works that attempt to develop such a system, the first of which uses the manual counting of vehicles as a data source [21], which greatly reduces the scalability of the method. Additionally, this method only models phasing behaviour, as will be described in Section 2.2.1. The system developed in the other research only works for VATSCs where the green-durations depend vary within tight limits, while the cycle-time and phasing behaviour are constant [44]. This type of VATSC can be found in Germany, but more dynamic types are common in the Netherlands [18].

In conclusion, no methods exist to model the behaviour of common VATSCs from FCD. To provide a starting point for the research described in this document, the next section highlights relevant aspects of research that has modelled TSC behaviour using some form of Markovian modelling approach.

### 2.2.1. Modelling traffic signal controller behaviour using HMMs

Section 1.1.1 describes which features make HMMs a promising candidate for modelling TSCs. However, at the time of writing, there are only three researches that have used some form of Markov model to represent the behaviour of TSCs.

The earliest example of Markov modelling in the context of predicting TSC behaviour was conducted in 2008 by researchers at Audi AG [20]. They assumed a TSC with a fixed cycle time and communication capabilities. A switching probability matrix is constructed for each second of the control cycle, based on historic data. The probability of switching moment is predicted by multiplying the current state vector with the switching matrix belonging to the current cycle-second. This method requires that the current state is known and therefore is not a true HMM. For the same reason it can not be trained using unlabelled data and can therefore not be used with FCD or any other type of shared data that only contains partial information about the true TSC-state. Also, as described in Section 2.1, not all TSCs have a fixed cycle time, so the method in [20] is limited in its application.

Another research investigated modelling the behaviour of intersection signal controllers using HMMs [21]. This research created relatively short, but dense observation sequences by counting all vehicles at an intersection during a couple of minutes and denoting their movement. By doing so, the length of the observation sequence used to learn the parameters of the HMM, is determined by the number of vehicles that is counted. The result of this is that the model is not based on true time, but on numbers of vehicles that cross the intersection. TSCs on the other hand use timers, which leads to a discrepancy between model steps and system steps. Additionally, the researchers struggled with fast state switching, because state duration in standard HMMs is only determined by the self-transition probabilities $a_{ii}$. The state duration probability distribution is therefore essentially a geometric distribution:

$$P(d_i) = a_{ii}^{d-1}(1-a_{ii}) \tag{2.1}$$

$P(d_i)$ denotes the probability of state $s_i$ lasting for a duration of $d$ time-steps. Figure 2.5 shows the resulting state duration distributions for different values of self transition probability $a_{ii}$. The authors of [21] noted that using high values of self transition probabilities can prevent fast state switching, but will never lead to state duration distributions that accurately model traffic signal switching behaviour. To clarify why the modelling performance of the models created by the research in [21] does not yield accurate models of traffic signal switching behaviour, we look at a useful feature that HMMs normally have, namely that they are generative models and can thus be used to generate data that is representative of the process that they model. If the HMMs created using the method described in [21] are used to generate a Markov chain $Q$, it is inevitable that the most common state duration is $d = 1$. This is not representative of TSC behaviour if a practical time-step sizes in the range of $0.1 : 1$ are used, because the minimum green time of traffic signals is typically several factors larger than one second. Although this analysis is not made by the authors of [21], they suggest that using a HSMM can provide a better solution to the fast state-switching behaviour that they observed. Section 2.5 explains the differences between standard HMMs and HSMMs and which variant is chosen for this thesis research.

Lastly, one research describes a method to provide cyclists with speed advice that is optimized for the goals of individual cyclists [22]. This method assumes that a state transition model of the TSC behaviour is available, in a format that is similar to the transition matrix of a HMM. The method can cope with the uncertainty that is inherent to predictions that are made using this type of transition model.

The research uses a state description that is a combination of all signals that are green and their

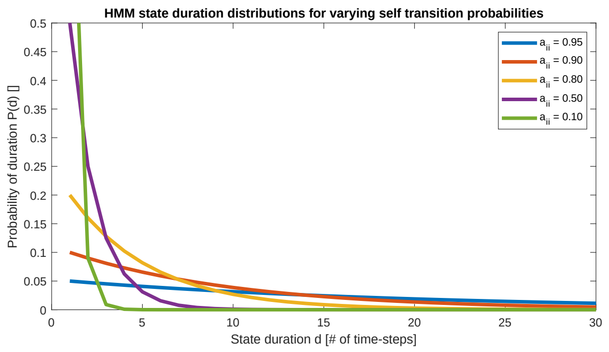


Figure 2.5: State duration distributions for state $s_i$ for different values of the self transition probability $a_{ii}$. It can be seen that a higher self transition probability leads to a higher probability of longer state durations. However, the state duration probability is always maximal for $d = 1$ and decays for higher durations, meaning that it is not a representative duration distribution for stages of traffic signal controllers.

individual elapsed green times. Using this state description, every possible signal output can be modelled. However, the research does not describe how such models can be constructed or learned. The flexibility provided by the state description means that a rather high number of states will be needed to describe the behaviour of TSCs at more complex intersections, which potentially makes the learning of parameters from sparse data impossible. Section 2.5.3 describes Variable Transition HSMMs, which have a similar structure as the models described in [22].

## 2.3. Data sources

Until now, all shared road user data has been referred to as FCD in this document, because this is the term that is most commonly used in research to indicate data of vehicle movements that is measured and shared by road users instead of detectors installed in the road or at the roadside [15, 17, 23, 45–49]. Although FCD, also known as probe data, is most common in literature, other data sources exist that can potentially be used to provide data to the system developed in this research. The data sources that are most commonly mentioned in literature are listed below, together with the sort of data they can produce:

1. Trajectory data collected and shared by smartphone-users, navigation systems or connected vehicles, i.e., true FCD:
    (a) road-user stationary in front of signal: movement has pending request
    (b) road-user passing stop-line: corresponding signal is green
2. V-log data, logged by the TSC [20, 25, 26]:
    (a) full history of the full control action of TSC: colour of all signals over time
    (b) full history of all detector activations: which directions have a pending request or flowing traffic

    Note that accessing this type of data requires permission from road authorities, therefore it is not considered for this research.
3. Vehicle-mounted cameras [50, 51]:
    (a) colour of all signals in field-of-view of camera
    (b) moving road-users for directions in field-of-view of camera

4. Stationary cameras at roadside [52]:

    (a) colour of signals in field-of-view of camera

    (b) traffic is flowing: signal is green

    (c) presence of queue: movement has pending request

From the listed sources, true FCD has the largest potential, because it can be shared without effort by anyone with a smartphone. Given that 1.8 million car drivers use the Flitsmeister application [46], most of whom live in the Netherlands, it can be expected that more than 10% of car drivers is interested in using an app that can save them time. It is therefore not unlikely that a similar percentage of road-users will use an application for GLOSA if that can save them time and fuel. However, the willingness of people to share their location data must be investigated.

## 2.4. Hidden Markov Models

This section describes the fundamentals of HMMs and introduces the terminology as it is used for the scope of this document.

### 2.4.1. Introduction to HMMs

HMMs are used to model systems whose outputs show temporal patterns and for which it is costly or impossible to directly observe the true system states. HMMs model the behaviour by learning the temporal relationships between system states and the relationships between system states and system outputs from observation data. The behaviour of the modelled system is represented by a Markov chain $Q$. $Q$ is a sequence of states which is generated by a Markov process. The process can be in any one of $N$ states from the set $S = \{s_1, s_2, ..., s_N\}$. The state at time $t$ is denoted by $q_t$. Each state has a certain probability of transitioning to another state. These transition probabilities are defined in transition matrix **A** as shown by (2.2). Element $a_{ij}$ represents the probability of transitioning from $q_t = s_i$ to $q_{t+1} = s_j$. The Markov assumption says that only the elements in **A** determine the state transitions, meaning that the system has a memory of one time-step. HMMs are called hidden because the states of the modelled system can not be directly observed, instead, at each time-step the system produces one of $M$ possible observations from the set $V = \{v_1, v_2, ..., v_M\}$. The observation at time-step $t$ is denoted by $o_t$ and the full observation sequence of length $T$ is $O = o_1, o_2, ..., o_T$. Each of these observations can be emitted by any of the states with a certain probability. These emission probabilities are represented in emission matrix **B** as shown by (2.3). Element $b_{ik}$ represents the probability that state $s_i$ emits observation $v_k$. To determine the most likely state sequence from an observation sequence, a probability distribution for the first state of the sequence is needed. The probability of state $s_i$ being the first state, $q_1$ is $\pi_i$. Vector $\mathbf{\Pi}$ contains the probabilities for all states of being state $q_1$. Together, **A**, **B** and $\mathbf{\Pi}$ make up all the parameters of a HMM. The symbol $\mathbf{\Lambda}$ is used to denote all parameters together: $\mathbf{\Lambda}=\{\mathbf{A},\mathbf{B},\mathbf{\Pi}\}$.

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NN} \end{bmatrix} \tag{2.2}$$

$$\mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1M} \\ b_{21} & b_{22} & \cdots & b_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ b_{N1} & b_{N2} & \cdots & b_{NM} \end{bmatrix} \tag{2.3}$$

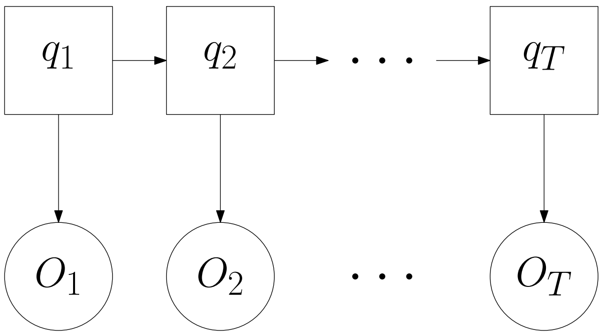


Figure 2.6: Overview of the HMM.

HMMs are doubly stochastic models, because both the underlying state transition process and the emission of observations are stochastic processes. This doubly stochastic nature fits the modelling of traffic signals using shared data, because the switching behaviour of traffic signals is dependent on the stochastic traffic supply and the production of observations is dependent on both the traffic supply and the distribution of data-sharing agents.

Three fundamental problems for HMMs exist:

1. The evaluation problem: evaluating the likelihood of a certain observation sequence given the HMM parameters $\lambda$, i.e. calculating $P[O|\lambda]$.
2. The decoding problem: inferring the state sequence of the underlying process given an observation sequence and the HMM parameters $\lambda$.
3. The learning problem: learning the model parameters $\lambda$ to maximize the likelihood of an observation sequence.

The algorithms that are commonly used to solve these three problems for the standard HMM are not discussed here, but provided as reference in Section B.1. A variant of HMMs that requires different algorithms will be used for this research, which is described in Section 2.5.

### 2.4.2. Challenges of modelling with HMMs

HMMs can be generated in several ways, but it depends on the application which approaches are possible and which of those is best. In some situations, the states of the system are obvious and its transition and emission behaviour is known from historical data. In such cases, the topology of the model, i.e., which parameters are initialized with a non-zero value, can be set before learning the final parameter values from an observation sequence. However, more often it is not directly clear what the order, i.e., the number of states $N$, should be or which observations will be emitted by which state. In these situations, a model selection process is needed to identify which model topology best suits the situation that is being modelled. In general the aim of this process is to identify the model with the highest parsimony, meaning that it should be as simple as possible while possessing good predictive powers, thereby striking the right balance between under- and overfitting [53]. Although this goal sounds reasonable, its precise mathematical meaning is vague and more importantly, there is no consensus on which measure is best at comparing the parsimony of competing models [54–56]. Often an information criterion is used, of which the Bayesian (BIC) and Aikake information criteria (AIC) are most used. All information criteria are used to score a model by weighing its goodness of fit against its complexity. The first is represented by the likelihood of the observation data for the given model and the latter by the number of parameters in the model. In this way, these criteria aim to identify the model that maximizes the information per parameter and thus has the highest predictive power possible relative to the number of parameters.

The criteria differ in how they weigh the importance of both measures and therefore they serve different purposes. Most importantly, the BIC was proven to always select the correct model if it is part of the candidate models, whereas the AIC does not have this property. Instead, the AIC assumes that none of the candidates is correct and selects the one that resembles the true situation most closely [57]. This can lead to the AIC selecting models that are more complex than necessary.

On a conceptual level, parsimonious models are generally obtained by minimizing the amount of model parameters, while pushing their values to the extremes. This means it must be looked for models that have few parameters, which ideally are either 1 or 0, meaning that there is no alternative for the event that the parameter represents or that the event will never happen. In a mathematical sense, this can be evaluated using the concept of entropy. The entropy of a probability mass function can be calculated using (2.4):

$$H(X) = -\sum_{i=1}^{n} P(x_i)\log(P(x_i)) \tag{2.4}$$

where X is a discrete random variable with $n$ possible outcomes $x_1, \dots, x_n$. It can be clearly seen that the entropy is zero for an element with the value zero, because the first term goes to zero. In the same manner, the entropy goes to zero for an element with a value close to one. A distribution thus has a high predictive power if its entropy is low, hence a logical goal when designing a model generation system for HMMs is to construct models with low entropy. Note that a model with many parameters can have a lower entropy than a model with fewer parameters if a large portion of its parameters is zero. However, if a model has more parameters, it can be expected that there are more local optima, so finding the correct, global optimum requires more data. Following this reasoning, there is a trade-off between the learnability of models, i.e., with how little data or effort the intended models can be learned, and model complexity, i.e., how many parameters a model has and how many situations it can therefore model.

**Parameter initialization and learning** The parameters of a HMM are iteratively updated for an observation sequence $O$ until the likelihood of the model having produced $O$ no longer increases. The most common method to update the parameters is the Baum-Welch algorithm [19], which is an Expectation Maximization method. Expectation Maximization algorithms converge to the nearest local optimum, but whether this is the intended global optimum depends on the initialization of the model parameters. Once the parameters of a HMM are learned, the likelihood of $O$ can be evaluated for the resulting model and directly compared to that of models with the same amount of parameters. Usually, multiple models with the same number of parameters are learned with the same $O$ from different random initializations. The model that yields the highest likelihood is selected and potentially compared to the best models with different amounts of parameters using an information criterion. How many different random initializations are needed depends on the system that is modelled and it can not be guaranteed that the correct model is found, since the amount of possible different initializations is infinite [56].

An example of a situation that is typically hard to find using random initializations is presented in [56] and shown in Figure 2.7.

$A$ shows the signal that is modelled and it can be seen that it roughly takes on three distinguishable values. Most initializations will lead to the model shown in $B$, because once $S_3$ has the highest emission probability for values around the middle, all observations of these values will be assigned to $S_3$. However, it can be clearly seen that there is a pattern in the signal that is not captured by $B$, namely that once the signal goes from the low value to the middle value, it will always continue to the high value and not back to the low value. The same is true when moving from the top value to the middle value. $C$ captures this pattern by splitting $S_3$ into $S_3$ and $S_4$, so that each state can only transit to one other state. (2.5) and (2.6) show the transition matrix and its entropy of $B$ and $C$ re-

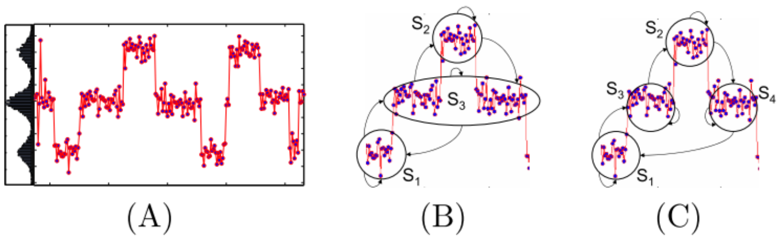


Figure 2.7: Example of a model that is hard to find using random initialization. *A* shows the signal that is modelled. *B* shows the model that is found from most initializations and *C* shows the model with the lowest entropy [56].

spectively, under the assumption that there are no self transitions. It can be clearly seen that *C* has the lowest entropy and can therefore be used to make predictions with higher certainty.

Unfortunately, states $S_3$ and $S_4$ have the exact same parameters in **B**, which means that the most initializations will start favouring either one of them and therefore lead to *B*. This example shows that the best model is not always the model with the least parameters and can not always be found easily.

$$A_B = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0.5 & 0.5 & 0 \end{bmatrix}, \quad H_B = 0.3466 \tag{2.5}$$

$$A_C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \qquad H_C = 0 \tag{2.6}$$

## 2.5. Hidden semi-Markov Models

As described in Section 2.2.1, the state duration modelling possibilities of standard HMMs are limited and insufficient to model TSC behaviour. This section shows in what way HSMMs overcome this shortcoming of HMMs. HSMMs release the Markov assumption by assuming that the next state is not only dependent on the current state, but also on the duration of the current and/or next state. Multiple variants with varying applications exist [28], but the most common type of HSMMs is the Explicit Duration HSMM, which is described in Section 2.5.1. Another type of HSMM is the Variable Transition HSMM, which offers more flexibility, as is explained in Section 2.5.3.

### 2.5.1. Explicit Duration HSMM

Explicit Duration HSMM theory expands standard HMM theory, by adding a duration probability distribution for each state [28]. Additionally, it is assumed that states have zero probability of self transmission, so all values on the diagonal of the transition matrix become zero. Instead, the probability that the system resides in a certain state for a certain duration is modelled by the additional duration probability matrix **D**, which can be seen in (2.7). In this matrix, $p_{i,d}$ indicates the probability that state $s_i$ lasts for $d$ time-steps. $D$ indicates the maximum state duration, which must be defined before learning. Following that each row of **D** represents a probability distribution, the elements on each row of **D** must add up to one.

$$\mathbf{D} = \begin{bmatrix} 0 & p_{12} & \cdots & p_{1D} \\ p_{21} & 0 & \cdots & p_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ p_{N1} & p_{N2} & \cdots & 0 \end{bmatrix} \tag{2.7}$$

The transition probability from state $s_i$ to state $s_j$ can now be described using (2.8), from which it becomes clear how the duration of $s_i$ influences the transition probability. Also, it can be seen that the parameters of **D** are used to scale the transition probabilities to each state equally, which means that the transition probabilities to each state stay the same relative to each-other with changing state duration.

$$P(q_{t+1} = s_j | q_t = s_i, \tau_t = d) = a_{ij} p_{id} \tag{2.8}$$

Because TSCs usually have both a minimum and maximum green time, this means that Explicit Duration HSMMs can potentially model TSC behaviour more accurately than HMMs.

By defining a matrix such as **D**, the duration probability distribution for each state becomes discrete and therefore is a probability mass function. Alternatively, state duration distributions can be modelled as parametric functions [28]. The duration distributions then become continuous and can be described using fewer parameters than when one parameter is used for each possible state duration. The downside of using parametric distributions is that the type of distribution has to be determined beforehand. Selecting the most suitable type of parametric distribution is not a straightforward task. Therefore this research limits to using probability mass functions in the form of **D** to describe state duration distributions. The set of parameters for a Explicit Duration HSMM are denoted as $\mathbf{\Lambda}$, where $\mathbf{\Lambda} = \{\mathbf{\Pi}, \mathbf{A}, \mathbf{B}, \mathbf{D}\}$. It can be easily deducted that the number of parameters of a Explicit Duration HSMM can be calculated using (2.9). Recall that $N$ represents the number of model states, $M$ the number of unique observations and $D$ the maximum state duration.

$$n_{par,EDHSMM} = N^2 + ND + NM + N \tag{2.9}$$

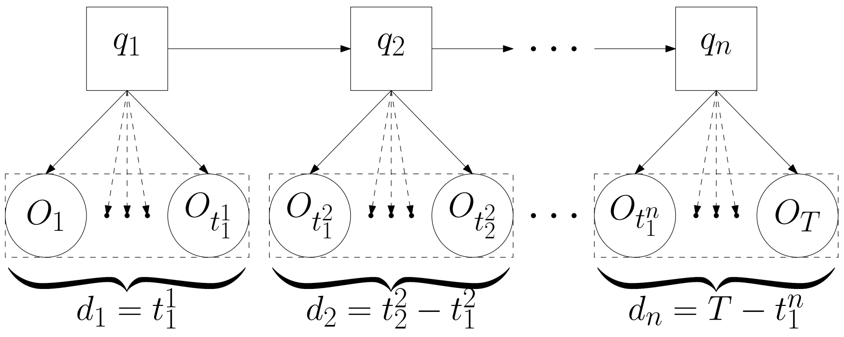


Figure 2.8: Overview of a Markov chain of a HSMM. The states in the Markov chain, $q_1, q_2, \ldots, q_n$, have durations $d_1, d_2, \ldots, d_n$ respectively. The decoding algorithm uses the parameters of **D** to determine which state sequence, with which durations have the highest likelihood for the given observation sequence $O$.

### 2.5.2. Forward-Backward algorithm implementation for Explicit duration HSMM

The three basic problems from Section 2.4 also apply to Explicit Duration HSMMs. Yu describes a Forward-Backward algorithm that is used to solve these problems. This algorithm iterates over the observation sequence $O$ twice. During the Forward pass, the evaluation problem is solved by iterating over $O$ from $t = 1$ to $t = T$. The Backward pass is then performed by iterating over $O$ from $t = T$ to $t = 1$, to determine the most likely state sequence $Q$ for the current model parameters $\mathbf{\Lambda}$ and to subsequently re-estimate $\mathbf{\Lambda}$.

Compared with standard HMMs, the learning algorithm not only has to evaluate potential transitions between states, but also all possible durations of each state. This is done by evaluating the likelihood of all possible partial observation sequences of lengths $1:D$ for each state, as can be seen in. For models with a long maximum state duration in combination with a large number of states, this can lead to numerical underflow, because the product of a large number of fractions has to be calculated. Numerical underflow occurs when values of variables become so small that the numerical precision of the machine on which they are calculated is insufficient to store the true value and therefore sets the value to zero. Yu therefore developed an implementation method that does not suffer from numerical underflow [58].

The following variables are defined to perform the forward pass of the Forward-Backward algorithm, using the definitions as previously described for general HMMs.

1. The forward variable $\alpha_t(i,d)$ represents the probability that $q_t$ is $s_i$ and has been the state for the $d$ time steps up to $t$, given the observation sequence up to time-step $t$:

$$\alpha_t(i,d) = P(q_t = s_i, \tau_t = d | o_1^t) = S_{t-1}(i)p(i,d) + b_i^*(o_{t-1})\alpha_{t-1}(i,d+1) \tag{2.10}$$

   Where $\tau_t$ represents the elapsed duration of $q_t$.

2. $r_t$ is a scaling factor that is used to avoid numerical underflow and is calculated by summing $\alpha_{t-1}(i,d)$ over time and space, multiplied by the probability of observing $o_t$:

$$r_t = P(o_t|o_1^{t-1}) = \sum_{d=1}^{D}\sum_{i=1}^{N} \alpha_{t-1}(i,d)b_i(o_t) \tag{2.11}$$

   if $\log r_t$ is summed over time, the log-likelihood of $\mathbf{\Lambda}$ producing $O$ is found:

$$LL(O) = \sum_{t=1}^{T} r_t \tag{2.12}$$

3. $b_i^*$ is scaled by $r_t$ and represents the probability that $o_t$ was emitted from state $s_i$:

$$b_i^*(o_t) = \frac{b_i(o_t)}{r_t} \tag{2.13}$$

4. $E_t(i)$ is defined for convenience of notation and represents the probability that state $s_i$ ends at time-step $t$, given the observation sequence $o_1^t$, i.e. the probability of a state transition from state $s_i$ at time-step $t$:

$$E_t(i) = P(q_t = s_i, \tau_t = 1 | o_1^t) = \alpha_{t-1}(i,1)b_i^*(o_t) \tag{2.14}$$

5. $S_t(i)$ is also defined for convenience of notation and represents the probability of state $s_i$ starting at time-step $t+1$, given the observation sequence $o_1^t$i.e. the probability of a state transition to state $s_i$ at time-step $t$:

$$S_t(i) = P(q_{t+1} = s_i, \tau_t = 1 | o_1^t) = \sum_{j=1}^{N} E_t(j)a(ji) \tag{2.15}$$

**Algorithm 1:** Forward pass of the Forward-Backward algorithm

```
initialize $\alpha_1(i,d)$ and $b_i^*(o_1)$
for t=2:T do
    calculate;
    $\alpha_t$ using (2.10),
    $b_i^*(o_t)$ using (2.13),
    $E_t$ using (2.14) $S_t$ using (2.15)
end
```

The scaled values of $b_i^*$ are used in the Backward pass of the Forward-Backward algorithm along with the following variables:

1. $\beta_t(i,d)$ is the backward variable and describes the probability that state $s_i$ is the state at time step $t$ and has been the state for the $d$ time steps up to $t$, given the observation sequence from $T$ to $t$, $o_t^T$. $\beta_t(i,d)$ is calculated for $i \in 1,\dots,N$ and $d \in 1,\dots,D$.

$$\beta_t(i,d) = \frac{P(o_t^T|q_t = s_i, \tau_t = d)}{P(o_t^T|o_1^{t-1})} = \begin{cases} S_{t+1}^*(i)b_i^*(o_t) & \text{d=1} \\ \beta_{t+1}(i,d-1)b_i^*(o_t) & \text{d>1} \end{cases} \tag{2.16}$$

2. $E_t^*(i)$ is defined for convenience and represents the probability of observation sequence $o_t^T$ given that $q_t$ is $s_i$ and that $s_i$ has started at time-step $t$, divided by the probability of $o_t^T$ given $o_1^{t-1}$:

$$E_t^*(i) = \frac{P(q_t = s_i, \tau_{t-1} = 1|o_t^T)}{P(o_t^T|o_1^{t-1})} = \sum_{d=1}^{D} p(i,d)\beta_t(i,d) \tag{2.17}$$

3. $S_t^*(i)$ is also defined for convenience of notation and represents the probability of $o_t^T$ given that $q_{t-1}$ is $s_i$ and that $s_i$ has started at time-step $t$, divided by the probability of $o_t^T$ given $o_1^{t-1}$:

$$S_t^*(i) = \frac{P(o_t^T|q_{t-1} = s_i, \tau_{t-1} = 1)}{P(o_t^T|o_1^{t-1})} = \sum_{j=1}^{N} a_{ij}E_t(i) \tag{2.18}$$

**Algorithm 2:** Backward pass of the Forward-Backward algorithm

```
initialize $\beta_t(i,d)$
**for** t=T-1:1 **do**
    calculate;
    $\beta_t$ using (2.16),
    $E_t^*$ using (2.17),
    $S_t^*$ using (2.18)
**end**
```

### 2.5.3. Variable Transition HSMM

Variable Transition HSMMs are similar to general HMMs in the sense that they consist of a transition matrix **A**, an emission probability matrix **B** and an initial state distribution **Π** [28]. However, the **A** matrix is different, because it takes over the function of the state duration matrix **D** used in Explicit Duration HSMMs. Each state in a Variable Transition HSMM consists of a tuple that describes both that meaning of the state and the elapsed duration of that state. This is realised by adding a dimension to the transition matrix, which models how the transition probabilities change for increasing state duration, as shown in (2.19).

$$a_{ij}(d) = P(q_{t+1} = s_j|q_{t-d+1:t} = s_i) \tag{2.19}$$

By defining state transition probabilities as shown in (2.19), a Variable Transition HSMM can represent situations where transitions from one state to another become more likely with increasing state duration, while a transition to a different state becomes less likely.

Although Variable Transition HSMMs are more flexible, they do have more parameters than Explicit Duration HSMMs, as can be seen when comparing (2.9) to (2.20). Recall that $N$ represents the number of model states, $M$ the number of unique observations and $D$ the maximum state duration. A Variable Transition HSMM only has less parameters for models with few states or short maximum state durations. This generally means that Variable Transition HSMMs require more data to be learned without overfitting [28].

$$n_{par,VTHSMM} = N^2 D + NM + N \tag{2.20}$$

## 2.6. Conclusion

From the findings presented in this chapter, it can be concluded that most TSCs adhere to a control structure consisting of a a number of stages equal to the maximum conflict group size. TSCs that use a similar structure with flexibility between subsequent stages are also common.

Also, it was found that GLOSA systems have a higher impact when more road users use it and that many road-users are willing to share their location using their smartphone if they can benefit from that.

Lastly, no previous research has developed a system to generate HSMMs of TSC behaviour, even though these models have features that make them suited for such a system, because they can model both phasing and timing behaviour from unlabelled data. However, several challenges related to modelling with HMMs were identified that also apply to HSMMs.

Chapter 3 describes the research problem for this thesis, based on the background information from this Chapter. Subsequently, the methodology is presented to solve this problem.

# 3 Methodology

This chapter presents the research problem and the proposed methodology to solve it. Section 3.2 performs the former by translating the findings from Chapter 2 to a concrete formulation of the problem that this thesis research aims to solve and under which assumptions a solution is developed. Section 3.2 describes the design of this solution in the form of a new modelling system.

## 3.1. Problem formulation

Figure 1.1 already showed a high-level overview of the system that this research aims to develop. Based on the literature presented in Chapter 2, this section formulates the goals and assumptions for which this system is developed.

### 3.1.1. Design goals

As stated in Section 1.2, the overarching goal of this research is to develop a system to generate models of TSC behaviour from 'FCD data, so that the resulting models can be used to generate GLOSA. Additionally, it was indicated that it should be strived to develop a system that can provide as many road users with GLOSA as possible. This was motivated by the fact that the proven potential of GLOSA systems to make a substantial contribution to the reduction of $CO_2$-emissions remains unused due to the limited applicability of existing systems. The review of literature from Chapter 2 has shown that the effects of GLOSA become more pronounced when more road-users have access to it and when it works for more intersections. However, it was found that most existing GLOSA systems only work in combination with TSCs that can broadcast Signal Phase and Timing messages, while most TSCs are not able to do so. Additionally, although systems exist that solely rely on shared data, none of them works with VATSCs with flexibility, while this type of TSC is common or even predominant in some countries, such as the Netherlands. The approaches that do use shared data make reserved assumptions on penetration rates, rarely exceeding 1%, while it can reasonably be expected that shared traffic data will become a more abundant resource in the foreseeable future due to the growing connectivity of vehicles and people in general. The combination of these factors explains why GLOSA systems have not had a serious impact on road traffic yet.

This research takes a new approach to the problem of modelling TSC behaviour by developing a solution that is aimed at being widely applicable. This wide applicability must be embedded in the design of the solution in two ways. Firstly, it should be able to model the most common types of TSCs under the circumstances where most road-users can benefit from GLOSA. More concretely, this means that the behaviour of FTTSCs, VATSCs and VATSCs with flexibility are modelled for the busier periods of the day. Secondly, the system has to be able to produce useful models from FCD, i.e., data that does not express the complete, true TSC-state at all times. However, no explicit bounds should be imposed on the level of sparsity. Instead, it must be investigated how much data is needed to generate useful models. By doing so, it can be estimated which future developments are necessary before large scale deployment is possible. A model qualifies as useful when it can be used to generate a strategy that improves the performance of a road-user for his specific goals compared to when he would have no knowledge about the TSC behaviour. Because the generation of such strategies is not part of this research, these can not be used to evaluate the performance of models. Instead, the following goals are defined for the design of the system:

**Goal 1** The developed system can generate models of:

- a: fixed-time TSCs;
- b: vehicle-actuated TSCs with fixed phasing;
- c: vehicle-actuated TSCs with flexibility.

**Goal 2** The developed system generates models whose states:

- a: cover all possible controller outputs;
- b: each represent one TSC state.

**Goal 3** The developed system generates models that represent phase transition behaviour as predictably as possible, i.e., models should have transition matrices with low entropy.

**Goal 4** The developed system generates models that explicitly model the duration probability distributions of all elements of the control structure of the modelled TSC.

**Goal 5** When provided with enough data, the developed system generates models whose state duration probability distributions converge to the true duration probability distributions of the TSC state that they represent.

**Goal 6** The developed system can generate models from data that is representative of FCD, i.e., data that:

- a: is sampled irregularly;
- b: only represents a part of the true TSC control output.

### 3.1.2. Assumptions

The following assumptions are made regarding the modelled situation, i.e. the intersection and its TSC:

1. The signal groups of the modelled intersection and the movements that they control are known in the form of a conflict-matrix. This assumption can be made because this information can be extracted from generally available sources [59].
2. The type of TSC at the modelled intersection is not known, because it can not be relied on road authorities to share this information and it is not generally available.
3. The number of stages in a TSC control scheme is equal to the size of the maximum conflict group of the intersection, as is the case for most TSCs [18].
4. Clearance times are never smaller than zero, meaning that two conflicting movements can never have a green signal simultaneously, as is the case in most countries [30].

5. Transitions between phases happen simultaneously for all signals belonging to the consecutive phases. In true situations, signals can transition a couple of seconds sooner or later than others, due to a difference in clearance times. For simplicity, these differences are ignored in this research.

6. Traffic conditions and controller parameters are stationary, meaning that it is investigated how a model of TSC behaviour at a certain time of a certain type of day can be generated. Field testing is necessary to determine how many models are needed per intersection, but this lies outside the scope of the research presented here.

7. the behaviour of TSCs is ergodic, meaning that all possible behaviour is exerted if the TSC is observed long enough. Because of this assumption, a sufficiently large sample of the TSC behaviour is representative of its true behaviour.

The following assumptions are made regarding the data that is used to learn the parameters and evaluate the learned models:

1. Data is available in the form of time-stamped data-points. A data-point indicates the colour of one signal at a specific moment in time. The process of translating shared data to this type of observations is not considered. Because of this assumption, the development of an additional system is needed before the developed modelling system can be used with real data. Several methods to perform this translation are described in literature [15, 24, 50].

2. Road-users strictly obey red signals. Research has shown that red-light-running occurs in only 0.1 to 0.5% of intersection crossings [60, 61]. The effects of red-light-running are therefore assumed to be negligible. Future research will have to investigate the true effects of drivers disobeying red signals.

3. Road-users can cross during a yellow signal, therefore, yellow signals are regarded as green signals.

4. Data-points are produced without error. It can be expected that true shared data contains errors, so the effects of this will have to be investigated in future research.

## 3.2. Design overview

HMMs can be used to model a wide array of processes, but there is no off-the-shelf solution for each modelling problem. Instead, it has to be assessed how HMMs can best represent the specific process that needs modelling and how these models can be constructed from the available data. This section describes how the goals from Section 3.1.1 are translated into a concrete design of the modelling system shown in Figure 1.1. For the structuring of this document, the system is divided into three elements as shown Figure 3.1. The elements are discussed in the order indicated by the white circles.

①: Section 3.3 explains that state duration distributions can only be explicitly modelled from FCD with HSMMs if observation sequences are constructed that contain empty observations or observations with multiple data-points. It is argued that observations with multiple data points can best be interpreted independently to lower the number of model parameters and thus increase the learnability of models. The consequences of this choice for the learning algorithm are discussed in Section 3.6.

②: Section 3.4 and Section 3.5 describe how the timing and phasing behaviour of TSCs is represented respectively by learned models to meet the design goals. It is explained why Explicit Duration HSMMs are chosen over Variable Transition HSMMs and that model states do not represent which signals are actually green, but which could be green. Also, it is chosen that TSC states that yield the realisation of the same signals, but represent a different state in the control scheme, are modelled with separate model states. Although this approach yields models with more states than strictly necessary to represent all possible outputs of the TSC, it lowers the entropy of the transition matrix of these models and is therefore preferred. It was shown in Chapter 2 that this choice results in models that are harder to find from random initializations, so Section 3.6 explains how this problem is circumvented using knowledge of the intersection layout.

③: Section 3.6 describes how the intended models (②) can be generated from observation sequences (①) and data of the intersection layout. It is first presented how the Forward-Backward learning algorithm by [58] is adapted to accept observation sequences that contain empty observations or multiple observations per time step. Then it is explained how the four steps of the model generation process shown in Figure 3.1 work, which can be summarized as follows: First, it is calculated which combinations of stages can create a valid control structure from the intersection data. Next, a model is initialized and learned for each possible set of stages. The model that yields the highest likelihood is then analysed to determine the control type. During learning, the most likely order of the stages is automatically found. Thirdly, a model with all states and their transition behaviour, as defined in Section 3.5, is initialized from the ordered set of stages and identified control type. Lastly the duration distributions of all states are learned from the observation sequence using the adapted learning algorithm.

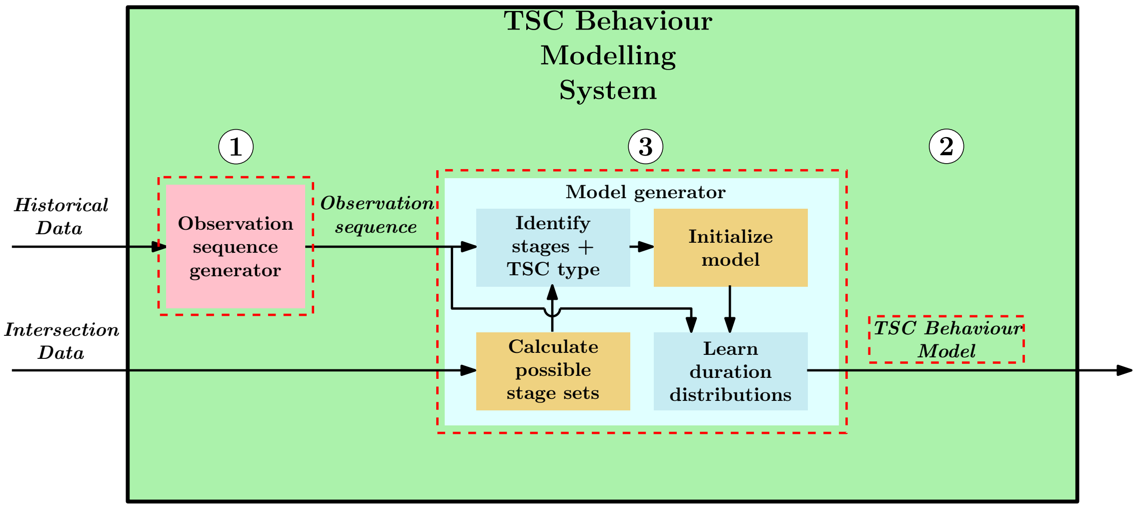

TSC Behaviour Modelling System
①
③
②
Historical Data
Observation sequence generator
Observation sequence
Model generator
Identify stages + TSC type
Initialize model
Intersection Data
Calculate possible stage sets
Learn duration distributions
TSC Behaviour Model

Figure 3.1: Elements of the TSC behaviour modelling system developed in this research. The input of the "Model generator" is described first in Section 3.3 (①) so that it is clear from which information the models must be extracted. Then, Section 3.4 and Section 3.5 (②) describe how the resulting models represent TSC timing and phasing behaviour. In this way, the boundaries and goals of the model generation process are clear before it is described in Section 3.6 (③). The blocks within

## 3.3. FCD to observation sequences

This section describes how observation sequences are created from time-stamped data points, so that **Goal 4** and **Goal 6** can be met:

**Goal 4** The developed system generates models that explicitly model the duration probability distributions of all elements of the control structure of the modelled TSC.

**Goal 6** The developed system can generate models from data that is representative of FCD, i.e., data that:

a: is sampled irregularly;
b: only represents a part of the true TSC control output.

Additionally, it is explained that data-points can either indicate that a signal is green, or that a signal can not be green.

### 3.3.1. Model time-steps

When modelling with HMMs, it is important to realize that the time between states in the Markov chain is not necessarily equal. A property of any type of HMM is namely that each observation is assumed to be emitted by a state, so a state is inferred and added to the Markov chain for each observation in the observation sequence $O$. This means that if observations are not equally spaced over time, i.e. are made at irregular intervals, the states in the Markov chain are also irregularly spaced over time. Usually a time step can be chosen by sampling data at a fixed interval. However, given **Goal 6**, it can not be assumed that data points are produced at fixed intervals. Instead, whether or not a data point is produced at any given time depends on whether a road user makes an observation at that time and is able to share it. It can be expected that there are periods during which no observations are made, so the modelling system should be able to cope with this. The remainder of this section explains what consequences this has for the construction of observation sequences by means of an example.

Figure 3.2 shows a collection of four data points, placed on a time axis to indicate at what time they were produced. The data points are given a colour to visualize how they are represented by the observation sequences from Figure 3.3.

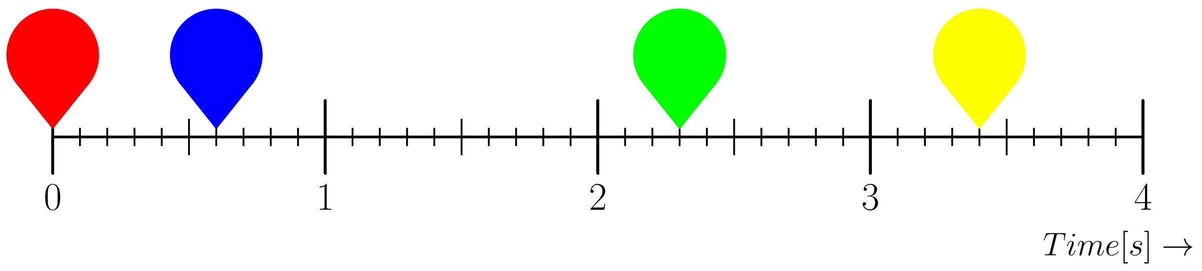
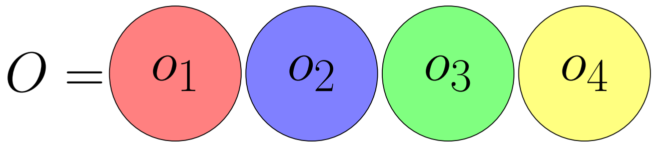
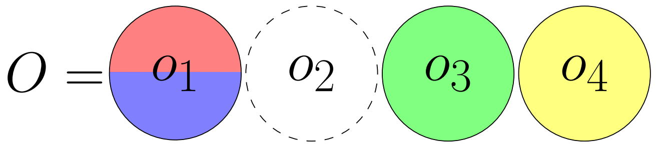


Figure 3.2: An example of four data points that are produced at irregular time intervals. Each data point is given a colour to visualize how it is mapped to the observation sequences in Figure 3.3.

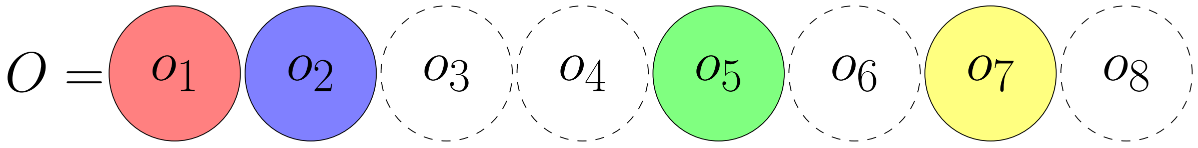
Figure 3.3 shows three different observation sequences that all result from the four data points shown in Figure 3.2. The observation sequences differ because they result from different choices regarding the structuring of the data points. The observation sequence in Figure 3.3a is the result of choosing that each data point yields one observation in $O$ and that the observations are ordered based on their time of production. This is a common approach and was used by the researchers in [21]. However, the result of this choice is that a HMM whose parameters are learned from such an $O$ has no direct relation to time. Instead, a state is inferred and added to the Markov chain for each observation, i.e. data point. The duration of a given state is thus based on the number of observations that is produced by that state, so if these observations are produced at irregular time-intervals, such as those from Figure 3.2, there is no explicit connection between a model step and time, which violates **Goal 4**. Because TSCs use a clock to determine their control action, for example to measure whether the minimum green time of the current stage has passed, it makes sense to construct models where state duration represents a duration in time and not a number of data points. Luckily, models with this property automatically result from observation sequences with an explicitly defined duration between observations. This duration is the time-step size of the model and can be chosen by the designer of the HMM. Figure 3.3b shows an observation sequence with a time-step size of 1 second, meaning that all data points that fall in the same 1 second time-interval are mapped to the same observation. It can be seen that both the red and the blue data point lie in the interval $0 \leq t < 1$ and are therefore both mapped to $o_1$. Because there are no data points in the second interval, $o_2$ is an empty observation, indicated by the dotted white circle. Section 3.6 explains how observations with multiple or zero data points can be processed by the learning algorithm. If it is chosen that the learned model should have a higher temporal resolution, the time-step size must reduced. Figure 3.3c for example shows an observation sequence resulting from a time-step size of 0.5 seconds.



(a) Resulting observation sequence for a model without an explicit time-step.



(b) Resulting observation sequence for a model with an explicit time-step size of 1 second. $o_1$ contains all data points that fall in the time interval $0 \leq t < 1$, $o_2$ contains those from $1 \leq t < 2$, etc.



(c) Resulting observation sequence for a model with an explicit time-step size of 0.5 seconds. The data points are assigned to the

Figure 3.3: The data points from Figure 3.2 can be mapped to different observation sequences $O$, depending on the type of model. For models where $O$ is of the type shown in Figure 3.3a, each data point adds one element to the observation sequence and thus to the Markov chain. For models with an explicit time-step size, the observation sequence consists of one observation per time-step. each data point is mapped to its closest observation. It can occur that more than one data point is mapped to one observation, an example of which is represented by $o_1$ in Figure 3.3b. Here, $o_1$ is both red and blue, because both the red and the blue data point lie closest to $t = 0$. It can also occur that no data points are mapped to a particular observation, because no data points were collected during the corresponding time-interval. These elements are indicated by the dotted white circles. Section 3.6 describes how these cases can be handled by the parameter learning and decoding algorithms.

**Temporal resolution** It has been established that, in order to fulfil **Goal 4**, observation sequences should have a fixed time step, so that resulting models explicit model timing behaviour. The time-step size is a modelling parameter that has to be defined before observation sequences can be produced. Figure 3.3b and Figure 3.3c show the effect of choosing a different time-step. It can be seen that the observation sequence from Figure 3.3c has a higher resolution and therefore is able to capture the true durations between the data points in Figure 3.3 more accurately. However, it can also be seen that this observation sequence is twice as long as that from Figure 3.3b and contains more empty observations.

The question must be asked what the most appropriate time-step size is for the problem at hand. So far it has been established that models with a small time-step size are more flexible and therefore theoretically able to model timing behaviour more precisely than models with a larger time-step size. However, a smaller time-step size also leads to a model with more parameters, and therefore requires more training data to learn parameters without over-fitting. On the other hand, a lower temporal resolution, i.e., a larger time-step size, can lead to prediction errors that make it impossible to generate useful GLOSA. Additionally, this increases the amount of contradictory data points being mapped to the same observation. The concept of contradictory data points is illustrated using an example; let's assume that the red data point from Figure 3.2 indicates that a certain traffic signal is red and the blue data point indicates that the same signal is green. It can be seen from Figure 3.3b that, if the model time-step size is 1 second, this leads to an observation where the same signal is both red and green simultaneously. This is an example of a contradictory observation and if these are not filtered out, the learning algorithm will learn that certain signals can be red and green simultaneously, which does not represent any true situation and is therefore unwanted. Further on, it will be explained how contradictory observations are handled in the construction of observation sequences, but first it is described which two situations can be indicated by data points

### 3.3.2. Types of data points

Data consists of data points that have both a meaning and a time stamp. For the scope of this research, a data point can be produced by any of the sources listed in Section 2.3 and its meaning is one of two conclusions about the state of a traffic signal. Firstly, it can be concluded that a traffic signal is green, for example when it is measured that a vehicle passes the stop-line of that traffic signal. Secondly, a data point can indicate that a road user is waiting to make a particular movement. This can lead to the conclusions that the signal is red and, if the TSC at the intersection is vehicle-actuated, that a request has been placed for it to turn green. The second type of data point must not be confused with measuring that the signal for a particular direction is red, because a red signal without waiting traffic does not indicate that the signal could not have been green at that moment in time. As explained in Section 2.1, VATSCs generally do not turn signals to green if there is no pending request, with the exception of signals that can be realised together with the signals that are set to green without causing a potential delay for traffic on another movement in the case that this movement would receive a request. For this reason, it can occur that a direction that belongs to the active stage of the TSC has a red signal. Using observations of this type of red signal does not contribute to the decoding of the state-sequence, because no conclusions about the TSC-stage can be drawn from it, whereas an observation of a green signal can lead to the conclusion that current TSC-stage is one to which this signal belongs. Similarly, observations of waiting traffic can lead to the conclusions that the active stage is not one that contains the movements for which traffic is waiting. These conclusions are drawn by the decoding algorithm based on the parameters of **B**. However, before explaining what the parameters of **B** represent, it must be decided how to handle observations that consist of multiple data points, such as $o_1$ from Figure 3.3b.

### 3.3.3. Types of observations

Above, the subject of a single observation containing contradictory data points was introduced. If this happens, the observation becomes conflicting, meaning that it either suggests that a signal is both green and red at the same time, or that two movements that can not have a simultaneous green signal according to the control structure are realised together. Either one of the conflicting data points can be removed from the observation sequence, or the individual data points can be assigned to the most logical subsequent observations. If either of these steps would not be taken, the learning algorithm would learn that certain states can produce conflicting observations, because it has no way of detecting that this can not occur in the process it models. In line with **Goal 2**, model states have to have an unambiguous interpretation and represent the true TSC control structure. Therefore, it is necessary that conflicting observations are removed from the observation sequence before learning. The process of translating true data points to conflict-free observation sequences lies outside the scope of this research. A situation can also occur where multiple data-points with the same meaning are sampled during the same time step. In this case, the meaning of these data-point is included once in the observation. A result of having conflict and duplicate-free observations is that each signal group can produce either one or no data point at each time-step. This means that the maximum number of data points in a single observation is equal to the number of signal groups on the intersection.

Given that multiple data points can be produced for a single time-step, it must be decided how observations consisting of multiple data points are interpreted. In order to do this, it must be defined what the elements in the observation space $V$ represent. There are two main approaches that can be taken. Firstly, each unique combination of data points can be regarded as a unique observation $v_k$, for $k \in 1 \dots M$. For a simple, unrealistic intersection with two conflicting movements, this leads to the observations shown in Figure 3.4. It is clear that signal-groups 2 and 5 can not be green simultaneously, so $V$ has eight elements instead of nine. If the observation sequences from Figure 3.3 are considered, the empty observations, such as $o_2$ from Figure 3.3b, would be labelled as $o_2 = v_1$. It is not hard to imagine that for an intersection with more signal-groups, the number of possible observations increases rapidly. In fact, if conflicting observations are also counted, the size of $V$, $M$, for an intersection with $n_{sig}$ signals can be calculated using (3.1):

$$M = 3^{n_{sig}} \tag{3.1}$$

from which it becomes clear that $M$ grows exponentially with $n_{sig}$. The base is 3, because each signal can be observed to be green or have waiting traffic, or is not observed. It can be imagined that this way of modelling observations becomes unfeasible for intersections with more signal-groups, even if conflicting observations are discarded. For example, an intersection with signal-groups 2, 5, 8 and 11 would have an observation space consisting of 55 elements. Section B.2 provides a visual representation of the resulting observation space. However, this is still an intersection that is relatively simple. Intersections with signal-groups 1 to 12 are not uncommon in the real world, but if such an intersection would be modelled using combined observations, an observation space of size $M = 94,463$ needed. This number is calculated by constructing each possible combination of data-points, which yields $M = 3^{12} = 531,441$ combinations, and removing all those that are conflicting, which can be checked using a conflict matrix of the signal groups at the intersection. Although experiments would have to show to what extend such a large observation space influences parameter learning, it can be expected that it yields an unfeasible number of parameters to learn with realistic amounts of data. Therefore, a different way of interpreting parameters is desired.

The second approach is to not combine data points into unique combinations, but instead assume that data points are independent of each-other. This means that each type unique data point represents an element of $V$. Inspiration for this approach comes from the research performed in [25]. In this research, V-log data is used to construct models of VATSC behaviour under the assumption that measurements made by different detectors are statistically independent from each-other, because one vehicle can only activate one detector. For the scope of this thesis research, data is not generated by detectors installed at the intersection, but instead is shared by road users. However, which data source is used for the modelling system does not influence the true TSC behaviour and the observations used in this research are a sparse sub set of those used in [25]. Therefore it is assumed that data points are independent for this research as well.

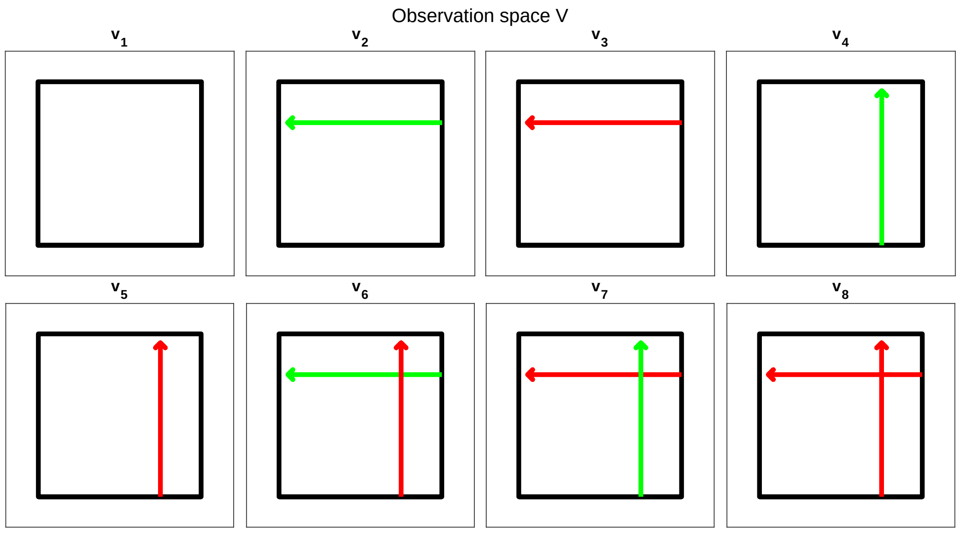


Figure 3.4: All possible combined observations for an example intersection with signal-groups 2 and 5. A green arrow indicates a green signal and a red signal is denoted by a red arrow. Note that the conflicting observations that both signal-groups are green is not part of the observation space.

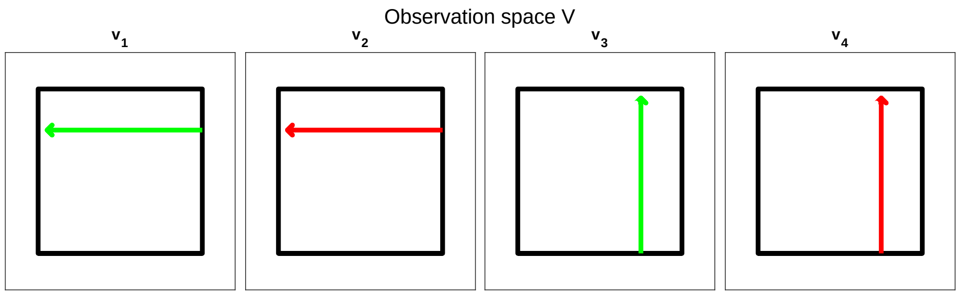


Figure 3.5: All possible independent observations for an example intersection with signal-groups 2 and 5. A green arrow indicates a green signal and a red signal is denoted by a red arrow. Note that there is no empty observation in this example.

Under this assumption, for the same example intersection as before, with only directions 2 and 5, $V$ becomes as shown in figure Figure 3.5. It is clear that the size of the observation space is reduced, because it is now equal to the number of unique data points and can thus be calculated using (3.2):

$$M = 2 * n_{sig} \tag{3.2}$$

After all, each signal can produce two types of data points, as explained above.

When looking at Figure 3.5, note that there is no separate observation for the case that there are no data points. Such an observation could be added to $V$, in which case it would indicate which state is most likely at a time-step for which no data points are available. Each state would then have a parameter in **B** to indicate the probability that no data points are available when being in that state, which intuitively can help with inferring the correct state when there is little context in terms of data, by relying on statistics about which state the system is in the most. However, it is expected that it is better to let the state duration distributions determine the probability of being in a certain state when there is no data available, because otherwise a positive feedback loop could be created during parameter learning with sparse data. This can be explained as follows: Expectation Maximization algorithms decode the observation sequence based on the current parameters and consequently update the parameters based on the resulting state sequence. If a state is inferred more often, more of the empty observations are assigned to it, so its probability of emitting an empty observation is increased when updating the parameters. During the next iteration, even more empty observations are therefore assigned to this state, leading to a further increase of the probability that this state emits empty observations. One could think that this reasoning applies to the other observations as well, but the difference lies in the fact that any other observation implies something about the true TSC-state and therefore there are limits to the number of consecutive observations of a single type, whereas a sequence of empty observations can be infinitely long.

In conclusion, the choice is made to treat data points as independent observations and to not have a separate observation to indicate a time-step without data points. This means that the observation sequence can contain time steps at which no or multiple observations are made. Section 3.6 explains how the learning algorithm is be adapted to cope with this.

## 3.4. Modelling timing behaviour

To comply with **Goal 4** and **Goal 5**, it must be defined how timing behaviour is captured by learned models.

**Goal 4** The developed system generates models that explicitly model the duration probability distributions of all elements of the control structure of the modelled TSC.

**Goal 5** When provided with enough data, the developed system generates models whose state duration probability distributions converge to the true duration probability distributions of the TSC state that they represent.

An important aspect of modelling TSC behaviour for GLOSA is that the timing behaviour of the TSC is represented accurately, otherwise it will be impossible to use resulting models to predict the switching moments such that speed advice can be generated. The ability of a model to do so is determined by the combination of its temporal resolution and the way that TSC-state duration is represented by its parameters. It was previously described that regular HMMs, as used in [21], are not suited to accurately represent TSC timing behaviour, but that HSMMs are developed specifically to model state duration behaviour. This section therefore compares two HSMM approaches that can be used to model the duration of states. It is analysed which of these approaches can best be used to reach the goals described in Section 3.1.1.

### 3.4.1. Variable Transition HSMMS or Explicit Duration HSMMS

Section 2.2.1 describes how the modelling approach developed in [21] is limited in its ability to model the timing behaviour of TSCs. The authors suggest using HSMMs to increase this ability, but it was found that only the research in [22] uses an approach resembling a HSMM, which can best be interpreted as a Variable Transition HSMM. However, this research did not investigate the learning of such models, so all factors related to learning are not discussed. This section compares the characteristics of Explicit Duration HSMMs and Variable Transition HSMMs and argues which can best be used to model and learn the timing behaviour of TSCs from sparse data.

As explained in Chapter 2, the main difference between Explicit Duration HSMMs and Variable Transition HSMMs is how the elapsed state duration influences the state transition probability. In a Explicit Duration HSMM, the probability that state $s_i$ transits to state $s_j$ after a duration of $d$ time-steps is calculated using (3.3), where $\tau_t$ indicates the past duration of state $q_t$:

$$P(q_{t+1} = s_j | q_t = s_i, \tau_t = d) = a_{ij} p_{id} \tag{3.3}$$

The transition matrix **A** of an Explicit Duration HSMM is stationary, meaning that the values of its parameters are independent over time. As a result of this, an Explicit Duration HSMM can not model a system where the transition to a certain state becomes less likely with increasing duration of the current state, while transition to another state becomes more likely. This is because all transition probabilities are scaled with the same parameter from **D**, meaning that only the probability that a transition happens changes over time, but not which transition that is. In a Variable Transition HSMM on the other hand, there is an individual parameter for the probability of each state transition for each elapsed state duration. This is achieved by adding a dimension, whose size is the maximum state duration $D$, to **A**. The consequence of this added flexibility is that the number of parameters of a Variable Transition HSMM is higher under most circumstances. (3.4) shows how the number of parameters of an Explicit Duration HSMM can be calculated and (3.5) shows the same for a Variable Transition HSMM. Remember that $N$ indicates the number of model states, $D$ the maximum state duration and $M$ the size of the observation space.

$$n_{par,EDHSMM} = N^2 + ND + NM + N \tag{3.4}$$

$$n_{par,VTHSMM} = N^2 D + NM + N \tag{3.5}$$

These calculations are the result of summing the number of elements in the matrices that define each of the models. A Explicit Duration HSMM is defined by the matrices **A**, **D**, **B** and **Π**, whereas a Variable Transition HSMM does not have a **D** matrix. However, because **A** has an extra dimension in a Variable Transition HSMM, the dominant term from (3.5) is scaled with $D$, compared to (3.4). Figure 3.6 visualises what consequences this has for the total number of parameters of a Explicit Duration HSMM compared to a Variable Transition HSMM. The example in the figure neglects the contribution of the size of the state space, $M$, because it is the same for both methods.

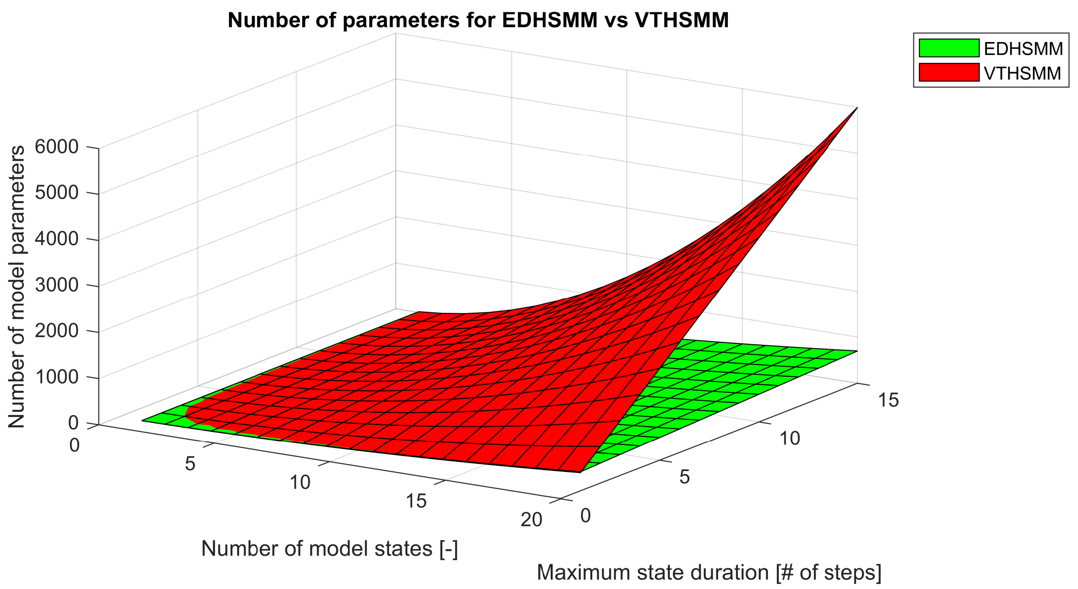


Figure 3.6: Number of model parameters for a Explicit Duration HSMM compared to a Variable Transition HSMM for varying model order, N, and maximum state duration, D. The number of possible observations, $M$, is neglected in this example.

Figure 3.6 clearly shows that the number of parameters of a Variable Transition HSMM exceeds that of a Explicit Duration HSMM by several factors, even for models of a relatively modest order. To be able to learn the parameters of models, it is therefore decided to use Explicit Duration HSMMs to model the behaviour of TSCs in this research. This means that it is assumed that the phasing behaviour of TSCs is independent of their timing behaviour. Whether this assumption holds can be investigated by analysing whether some transitions occur more often than others after a certain elapsed state duration, while the opposite is true before this elapsed duration. If this is found to be the case for a state, it can be split into two new states that represent the situation before and after the elapsed duration respectively. With this targeted state-splitting approach, other situations that are observed to correlate with a change in TSC behaviour can also be captured by the model, with the aim to improve the prediction quality of the model. However, the aim of this research is to lay the basis of a new approach for the modelling of TSC behaviour and therefore state-splitting does not lie within the scope of this research.

Section 3.6 explains how the Forward-Backward algorithm presented by [58] is used to learn the parameters of the intended Explicit Duration HSMMs.

### 3.4.2. Maximum state duration

When modelling with Explicit Duration HSMMs, an important modelling parameter is the maximum state duration, $D$. This parameter determines the maximum number of consecutive steps that the same state can be inferred. Setting this parameter too low results in a situation where the actual process is in a certain state longer than the model can represent, which leads to erratic behaviour of the model. On the other hand, setting $D$ too high can lead to different problems. To begin with, the number of parameters increases with $D$, although this is not the dominant term, so this only a problem if there are strict storage limitations. More importantly, a $D$ that is set too high can lead to wrong inference behaviour and thus to the incorrect learning of model parameters when little data is available. In a situation where there is a system with two alternating states, if one of them happens to stay unobserved for one cycle, that state could be left out of the inferred state sequence, because the decoding algorithm can find a solution where only the observed state is inferred for a long duration, overarching the unobserved state. If the maximum state duration would have been limited, the decoding algorithm would be forced to infer the unobserved state as well. Based on this example, it can be stated that $D$ can best be kept smaller than the cycle time. 120 seconds is commonly used as the maximum cycle time [30], so this should be regarded as the upper limit of $D$. It must be investigated which value of $D$ yields the best performance and whether this number can be set for all TSCs, or that this needs to be determined for each TSC individually.

## 3.5. Modelling phasing behaviour

This section explains how TSC states are represented by model states to comply with the goals that were set for the system developed in this research. It is described how a known TSC control structure should be represented by a learned model. Timing behaviour is not regarded in this section, instead it is assumed that each state lasts one time step and that self-transition probabilities are zero. The latter assumption is in accordance with the definition of Explicit Duration HSMMs, so the transition matrices described in this section are compatible with Explicit Duration HSMMs. **Goal 1**, **Goal 2** and **Goal 3** relate to the modelling of phasing behaviour:

**Goal 1** The developed system can generate models of:

- a: fixed-time TSCs;
- b: vehicle-actuated TSCs with fixed phasing;
- c: vehicle-actuated TSCs with flexibility.

**Goal 2** The developed system generates models whose states

- a: cover all possible controller outputs
- b: each represent one TSC state

**Goal 3** The developed system generates models that represent phase transition behaviour as predictably as possible.

To comply with **Goal 3**, the state parameters of the state transition matrix should have a low entropy. In Chapter 2, an example was given based on the research in [56], which showed that a model with states that have similar values in **B**, i.e., states that are inferred based on the same observations, can have a lower entropy if these states have different transition behaviour, i.e., can transit to and from different states. Therefore, TSC states that have a different place in the control structure, but that result in the realisation of the same signal groups, should be modelled using different model states. An example of such TSC states can be seen in Figure 3.7, which represents the control structure of a TSC with controller flexibility. It can be seen that the flexibility states, numbered 1.1, 2.1 and 3.1 yield the realisation of the same signals, but have a different meaning.

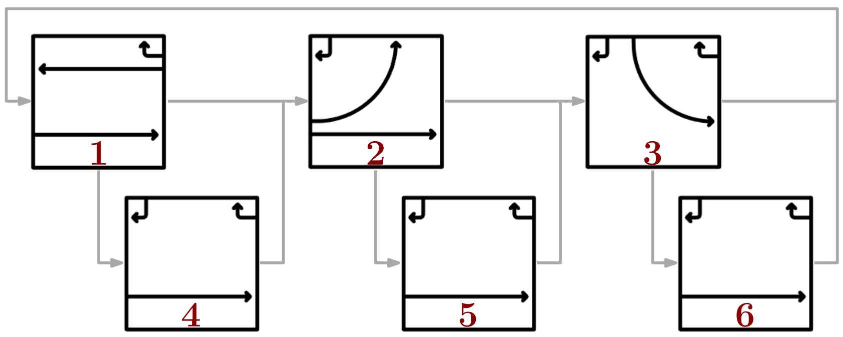


Figure 3.7: Example of a flow diagram for a TSC with three stages and flexibility. The blocks that represent the stages are numbered with 1 to 3 and blocks that represent flexibility phases are numbered 4 to 6. Only movements with a green signal are shown, all other signals are red.

If it is assumed that states that can be succeeded by multiple states have an equal probability of transitioning to any of the subsequent states, the transition matrix of the control structure in Figure 3.7 looks as shown in (3.6):

$$A_{separate} = \begin{bmatrix} 0 & 0.5 & 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0 & 0.5 & 0 \\ 0.5 & 0 & 0 & 0 & 0 & 0.5 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \qquad H_{separate} = 2.0794 \tag{3.6}$$

where $H_{separate}$ indicates the entropy of the transition matrix. Figure 3.8 shows a transition model where all TSC states that yield the realisation of the same signals are modelled using one state. As explained in Section 2.4, Expectation Maximization algorithms usually get trapped by the local optimum that such a model represents, because it assigns all similar observations to the same state.

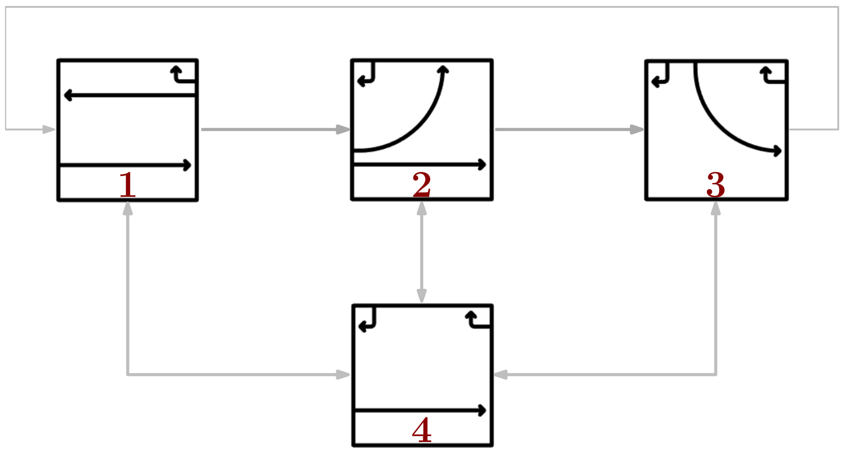


Figure 3.8: Example of a transition diagram of the control structure shown in Figure 4.2, when all three flexibility phases are modelled by the same state. States 1, 2 and 3 represent the stages and state 4 represents the flexibility phases. Only movements with a green signal are shown, all other signals are red.

The transition matrix belonging to the model in Figure 3.8 looks as shown in (3.7):

$$A_{combined} = \begin{bmatrix} 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0.5 & 0.5 \\ 0.5 & 0 & 0 & 0.5 \\ 0.33 & 0.33 & 0.33 & 0 \end{bmatrix}, \qquad H_{combined} = 3.1770 \tag{3.7}$$

where it can be seen that the entropy of this transition matrix, indicated by $H_combined$ is higher than that of the transition matrix of the true TSC control structure and therefore has lower predictive powers. For this reason, it is chosen that learned Explicit Duration HSMMs of TSC behaviour should resemble the control structure of the modelled TSC, even though it is known that these models can probably not be found in one step from a randomly initialized model. This choice is in accordance with **Goal 2b**. To also meet **Goal 2a**, it has to be defined how transitions between the elements of the control structure of a TSC are modelled.

### 3.5.1. Modelling signal transitions

When a TSC has decided that it must change from one stage to the next, or that there is an opportunity to realise a flexibility phase, it initiates a transition process. This process consists of three steps. First, all signals that are currently green, but have to become red are turned to yellow. After a predetermined yellow duration has passed, the yellow signals turn red. As explained in Section 2.1, the signals that have to become green do not turn green instantly. Instead, the TSC waits for a predetermined clearance time to pass, to allow road-users that passed under the yellow signal to clear the conflict area. Once the clearance time has passed, the signals of the new phase are turned to green. Due to differences in clearance time between sets of conflicting movements, one signal can turn green before another, with a difference of at most a few seconds. This difference is neglected here for simplicity, but it should be researched what effects this has on the usefulness of learned models under real-life conditions. This section analyses how this transition process can best be modelled by the model states. Two approaches are described and it is explained why the last one is chosen.

Firstly, it can be chosen that no states should be added to the model to represent TSC-transitions. In this case, a solution must be described to handle observations that can only indicate that a transition process is taking place. The most obvious example would be to observe that all signal groups are red simultaneously. This situation can occur when the phase before and after the transition do not contain the same movements and the transition requires a positive clearance time. The simplest way of handling such observations is to discard them by removing them from the observation sequence. However, this leads to the destruction of valuable information, because the timing of transitions is of particular interest when generating GLOSA, since transitions mark the moments when signals turn green and thus indicate when a road-user should arrive at the intersection. Another option is to change the parameters of **B**, so that observations of transitions can be assigned to existing states. However, this requires that the parameters of **B** of a single state become contradictory, because they suggest that a signal can be either red or green when being in that model state. In combination with independent observations, as described in Section 3.3, this means that states can be expected to be incorrectly inferred. It is clear that this is not desired, so a different way of modelling transitions is required.

The second approach adds states to explicitly model transitions. As described in Section 2.4, there are benefits to adding a state for each unique situation, even if this means that there are multiple states with seemingly similar interpretation in terms of which observations are associated with them. In the context of transitions, this means that each transition between a stages or flexibility phases is represented by one model state. If this is applied to the TSC shown in Figure 3.7, the state space of the resulting model is shown in Figure 3.9. Each state transition with a non-zero probability is indicated by grey arrows, to show that there is a limited amount of transition possibilities from each state. It can also be seen that there are now 15 states, but the entropy of the resulting transition matrix is the same as for the model shown in Figure 3.7, because all transitions have only one possible preceding and succeeding state and therefore only add transition parameters with value 1 or 0 to the transition matrix. It was explained in Section 2.4 that the entropy is zero of a parameter that has value 1 or 0. Therefore, transition states can be included in learned models without losing any predictive power.

Additionally, by defining models in this way, every possible controller output is represented by one of the model states, as required for **Goal 2a**.

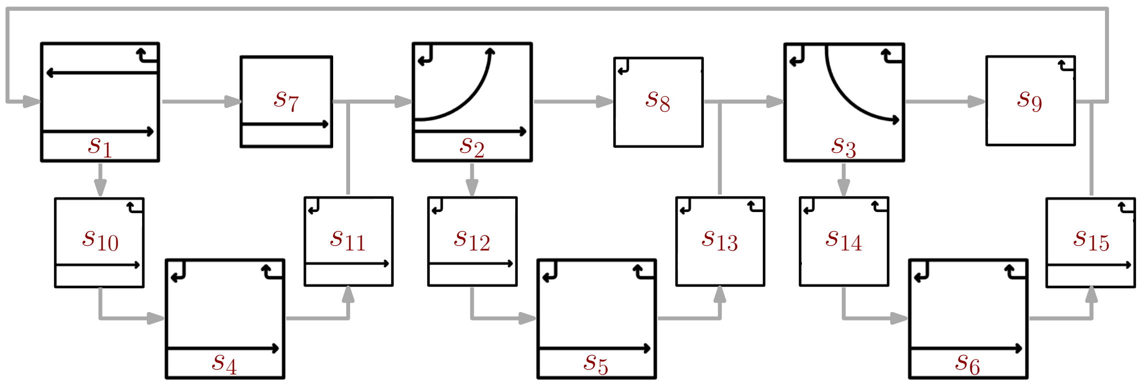
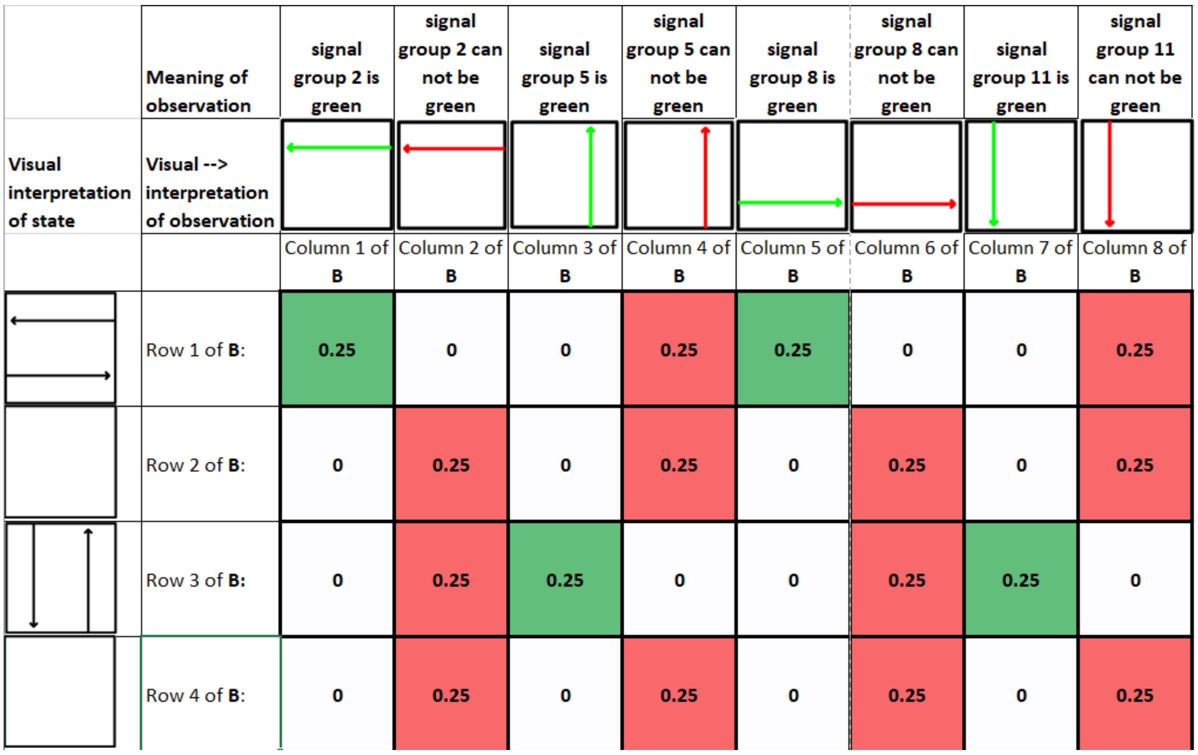


Figure 3.9: State space with transition states. If there is no grey line between states, then a transition is not possible.

### 3.5.2. Interpretation of parameters

So far, Section 3.4 has explained why the Explicit Duration HSMM framework is chosen, Section 3.3 has described why independent observations will be used and Section 3.5 defined the high level relationship between model-states and TSC-states. This section describes how these choices influence the model on a low level, by explaining how each parameter of a learned model should be interpreted if it is constructed according to the design choices made before. Only the parameters of **B** are discussed, because the meaning of the other model parameters is clearly defined in Section 2.4 and Section 2.4 and does not depend on the modelling of TSCs, whereas the parameters of **B** directly show which combination of green signal-groups is represented by each state of the HMM.

Given that each signal group can produce two observations, the observation space is of size $M = 2n_{sig}$. For a model with $N$ states, this means that **B** has size $N \times 2n_{sig}$. The signal groups of the intersection are sorted according to their numbering. The odd indices of the observation space represent observations of a green signal and the even indices indicate observations of red signals, such that observations of the signal group with the lowest number are represented by observations "1" and "2". Figure 3.10 shows an example of an observation emission probability matrix, **B**, for a TSC controlling signal groups 2, 5, 8 and 11. The matrix has been constructed under the assumption that each observation has an equal probability of being emitted.



| | Meaning of observation | signal group 2 is green | signal group 2 can not be green | signal group 5 is green | signal group 5 can not be green | signal group 8 is green | signal group 8 can not be green | signal group 11 is green | signal group 11 can not be green |
|---|---|---|---|---|---|---|---|---|---|
| Visual interpretation of state | Visual --> interpretation of observation | | | | | | | | |
| | | Column 1 of B | Column 2 of B | Column 3 of B | Column 4 of B | Column 5 of B | Column 6 of B | Column 7 of B | Column 8 of B |
| | Row 1 of **B**: | 0.25 | 0 | 0 | 0.25 | 0.25 | 0 | 0 | 0.25 |
| | Row 2 of **B**: | 0 | 0.25 | 0 | 0.25 | 0 | 0.25 | 0 | 0.25 |
| | Row 3 of **B**: | 0 | 0.25 | 0.25 | 0 | 0 | 0.25 | 0.25 | 0 |
| | Row 4 of **B**: | 0 | 0.25 | 0 | 0.25 | 0 | 0.25 | 0 | 0.25 |

Figure 3.10: Interpretation of the parameters of **B** of a learned Explicit Duration HSMM of a TSC with two stages.

As explained in Section 2.1, VATSCs only turn those signals of the current stage to green for which there are requests. This means that the realised phase can be a subset of the TSC state. It is chosen that full stages are used as model states, which means that model states do not represent which signals are actually green, but which could be green. Because it was defined in Section 3.3 that there are no observations of signals that are red without waiting traffic, this choice will not lead to the inference of incorrect states. If a signal is namely observed to have waiting traffic, it is certain that the TSC is not in a state where it can turn that signal to green, whereas an observation of a red signal without waiting traffic could not lead to the conclusion that the signal could not have been green if there would have a request from a road user.

## 3.6. Generating models

The previous sections in this chapter describe how observation sequences are constructed and what learned models should look like to comply with the design goals described in Section 3.1.1. This section describes an approach that is developed to translate these observation sequences into the intended models. First, it is explained how the HSMM learning algorithm from [58] is adapted to accept the observations described in Section 3.3.3. Then, an analysis of the problems related to finding the intended models is performed and an approach is presented that solves these problems for the modelling of TSC behaviour.

### 3.6.1. Handling multiple or empty observations

In Section 3.3, the choice was made to treat each data point as an independent observation. This was done to drastically reduce the number of model parameters, of which the expected consequence is that models can be learned from less data, because models with less parameters are usually less prone to overfitting. However, it was shown that a consequence of the independence assumption is that observation sequences can contain multiple observations at a single time step, with an upper limit defined by the number of signal groups installed at the intersection. The standard Explicit Duration HSMM learning algorithm described in Section 2.5, has to be adapted to this, because it expects exactly one observation per time step [58]. This is incorporated in the learning algorithm by calculating the average emission probability of all observations at a given time step, for each state, by changing (2.11) from:

$$r_t = \sum_{d=1}^{D} \sum_{i=1}^{N} \alpha_{t-1}(i,d) b_i(o_t) \tag{2.11}$$

to:

$$r_t = \sum_{d=1}^{D} \sum_{i=1}^{N} \alpha_{t-1}(i,d) \prod_{x=1}^{n_{obs_t}} b_i(o_{t,x}) M \tag{3.8}$$

and (2.13) from:

$$b_i^*(o_t) = \frac{b_i(o_t)}{r_t}, \quad \text{for } 1 \le i \le N \tag{2.13}$$

to:

$$b_i^*(o_t) = \frac{\prod_{x=1}^{n_{obs_t}} b_i(o_{t,x}) M}{r_t}, \quad \text{for } 1 \le i \le N \tag{3.9}$$

where $n_{obs_t}$ denotes the number of observations at time-step $t$. Note that the size of the observation space, $M$, is used to normalize the values of $r_t$ and $b_i^*(o_t)$, to assure that the likelihood of the observation sequence is independent of the number of observations. In the normal learning algorithm, the average value of $b_i(o_t)$ is $1/M$, because this is the average value of the parameters in **B**, given that each row in **B** has $M$ elements that sum to 1. The values of $r_t$ and $b_i^*(o_t)$ can thus be normalized by multiplying each observation emission probability by the inverse of this average value: $\frac{1}{\frac{1}{M}} = M$

An additional change must be made to the learning algorithm, because it can be expected that there are time steps at which there are no observations due to the sparse nature of FCD, as explained in Section 3.3. In these cases, i.e., when $n_{obs_t} = 0$, all states are assumed to have an equal probability of having produced the observation at this time-step, by defining that:

$$\prod_{x=1}^{n_{obs_t}} b_i(o_{t,x}) = \frac{1}{M}, \quad \text{for } 1 \le i \le N \tag{3.10}$$

This means that for time steps without observations, only the values of $\alpha_{t-1}$ determine which state is most likely. Again, it can be seen that $M$ is used to scale the outcome of (3.10) to be equal to the average value of the parameters in **B**.

With the presented adaptations to the Explicit Duration HSMM Forward-Backward algorithms, these can be used to learn parameters from observation sequences with no, one or multiple observations at a time-step.

### 3.6.2. Finding the global optimum

Expectation Maximization algorithms, like the Forward-Backward algorithm for Explicit Duration HSMMs [62], converge to an optimum, but it depends on the initialization of parameters which optimum that is [19]. The intention is always to find the global optimum, which is defined as the model that has the highest likelihood of having produced the observation sequence that is used for learning. However, many local optima can exist, depending on the modelled system and the used model dimensions. Generally, model parameters are initialized randomly, unless it is desired that prior knowledge of the modelled system is already present in the model before learning [19]. For example, it could be assumed that all transition-states have a maximum duration of 10 seconds, in which case all parameters in **D** that represent the probability of a longer duration of a transition state are initialized as zero. It is important to note that a parameter that has a value of zero can never be changed to any other value by the learning algorithm. Initializing parameters with a value of zero hence limits the flexibility of the model and this should therefore only be done when it is absolutely certain that a parameter should be zero. A common approach to finding the model that represents the global optimum is to randomly initialize several models and to update their parameters with the same observation sequence. The model with the highest likelihood is then chosen as the final model. However, it is not possible to determine beforehand how many different models must be initialized before the correct one is found, neither is there a way to know for sure that the found best model represents the global optimum. One situation where learning algorithms typically struggle to find the global optimum is when this optimum contains multiple states that have similar values in **B** [56]. For example, if the intended model looks like the one shown in Figure 3.9, where states $S_4$, $S_5$ and $S_6$ obviously have similar observation parameters, this must be forced by the parameter initialization. However, for the scope of this research, it was assumed that no prior knowledge of the TSC type or control structure is known. Luckily, the types of TSCs considered in this research have several features that can be exploited to provide a starting point for learning. A model generation system that is tailored to finding the intended models of TSC behaviour from limited quantities of data will now be described. An overview of this system, consisting of a four-step process, can be seen in Figure 3.11.

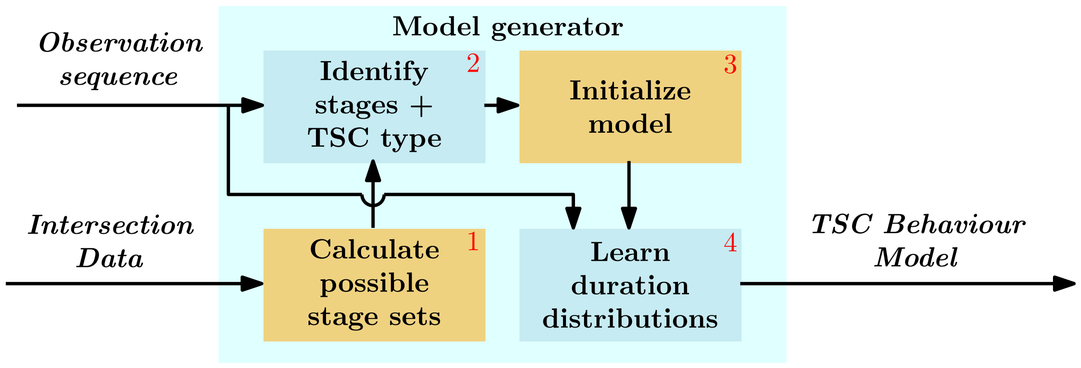


Figure 3.11: Overview of the steps involved in the model generation process

### 3.6.3. Step 1: calculating all possible stage sets

All TSC types considered in this research have a control structure that consists of a set of stages, which is potentially expanded with flexibility phases. The stages are maximal non-conflicting groups and are activated in a fixed order. The number of stages in the control structure is equal to the maximum conflict-group size and each signal group at the intersection is part of at least one stage. Based on this knowledge, all possible valid sets of stages can be calculated from the available intersection layout data. Figure 3.12 shows all possible sets for an intersection with signal groups 1-12.

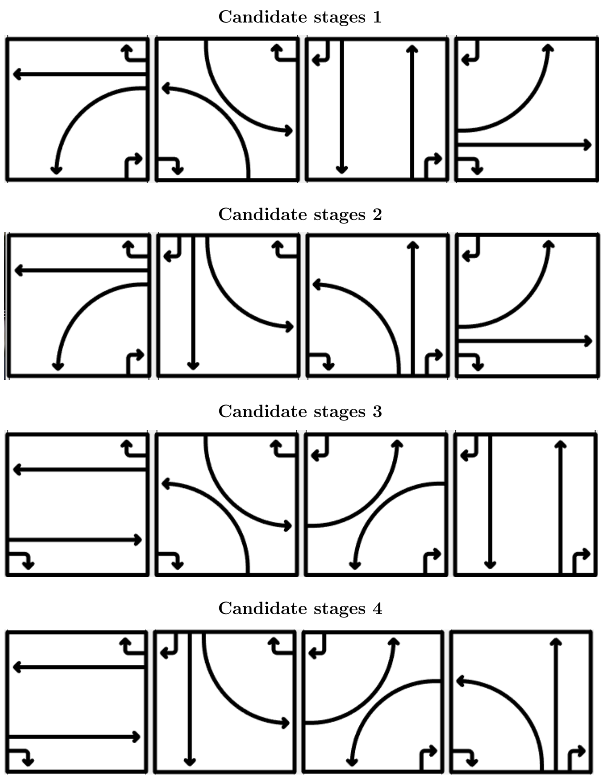


Figure 3.12: All possible sets of stages for an intersection with directions 1 to 12. The stages in each set can be ordered in any way to form a control structure. Note that the third set of stages shown here is equal to the true stages of Figure 4.1.

### 3.6.4. Step 2: Identifying the correct stages and type of TSC

This step aims to find three basic features of the TSC:

1. Which stages are used
2. The ordering of the stages
3. Whether the TSC uses controller flexibility

The features are found by initializing and learning an Explicit Duration HSMM for each set of stages found in Step 1. Within each model, each stage is represented by one state. The parameters of **A** and **D** are initialized uniformly and those of **B** are initialized to represent the correct combination of red and green signals for each stage, as shown by the example in Figure 3.10. Then, the parameters of all initialized models are learned from the available observation sequence, using the Forward-Backward algorithm from [58], with the adaptations described earlier in this section.

When the parameters of all models have been learned, the model that yields the highest likelihood is selected, under the expectation that this model represents the true set of stages. From the learned **A** of this model, the ordering of the stages can easily be deducted. Additionally, it is evaluated at what percentage of the time steps the observations can not be fully attributed to any one of the stages, or any transition between stages. If this percentage is higher than a certain threshold, it is concluded that the TSC has flexibility, because it is then likely that some controller outputs are not represented by the full stages, but instead are a combination of stages.

### 3.6.5. Step 3: Initialize the model structure

From the information found by Step 2, a model can be initialized that has the correct structure. If it was found that the modelled TSC has no flexibility, the model is constructed by adding transition states to the model selected in Step 2, for the transitions that can be expected from a control structure with the order and stages that were identified. The parameters of **D** are initialized uniformly and those of **B** are again initialized to represent the correct combinations of observations. For a TSC with four stages, a model is thus created with 8 states: 4 states to represent the stages, and 4 to represent the transitions between stages.

However, if it was found in Step 2 that the TSC uses controller flexibility, it is calculated for each stage which phases can be expected to be realised by the TSC as their successor. Remember that a TSC with flexibility can realise any phase that is a non-conflicting combination of signal groups of successive stages. For each of these phases, a state is added to the model and the parameters of **A** are initialized so that states that represent stages can only transit to their flexibility states or the next stage. Flexibility states transit to the state that represents the stage that follows the stage from which they were reached. If a state can transition to multiple other states, the transition probability parameters are uniformly initialized. Finally, for each non-zero parameter in **A**, a transition state is added to the model, so that a model representative of, for example, the one shown in Figure 3.9 is constructed.

### 3.6.6. Step 4: Learning the final distributions

As a final step, the model initialized in Step 3 is learned on the available observation sequence to find the parameter distributions that yield the highest likelihood. If enough data is available and the correct model structure was found, the correct transition and duration distributions are learned.

## 3.7. Conclusion

This chapter described the goals for the TSC behaviour modelling system developed in this research and presented the details of the design of this system. Next, Chapter 4 will describe, perform and discuss the results of experiments to evaluate to what extend the goals were met and how much data is needed to find the intended models.

# 4 Experiments

This chapter describes experiments to analyse to what extent the design goals stated in Section 3.1.1 are met by the system described in Section 3.2 and to evaluate whether the chosen design outperforms other, more conventional approaches. Given that no methods have previously been developed to model the behaviour of VATSCs with controller flexibility, it is not possible to directly compare the performance of the system described in this research with a benchmark. Neither is it possible to evaluate the usefulness of the proposed method for every imaginable situation. Instead, the experiments are performed for two simulated intersections with simple, but realistic behaviour. Section 4.1 explains how these intersections are simulated in MATLAB®and how data is generated. The subsequent sections describe three experiments and presents and discusses their results.

Firstly, the proposed model generation system shown in Figure 3.11 is compared to learning the parameters of randomly initialized models directly. This is done to investigate the benefits of the proposed system over a more conventional approach for varying levels of data sparsity.

Secondly, it is investigated how well models with correctly initialized states can learn duration distributions for varying penetration rates and observation durations.

Then, it is investigated how well learned models can be used to estimate TSC-states once learned, to indicate their usefulness under real-time conditions.

Lastly, the entropy of models resulting from different design choices is compared to verify the choices made in Section 3.2.

## 4.1. Experimenting environment

Before the experiments themselves are described, this section explains in which environment the experiments take place and how data is generated.

Because no previous attempts have been made to use HSMMs for the modelling of TSC behaviour, the main goals of this research are to find the requirements to construct meaningful models and identify limitations of the presented modelling approach, as described in Section 1.2. Using simulated data makes it easier to find these requirements and limitations, because changes to the data source can be made at will. However, to assure that the results of the experiments with simulated data give insight into the possibilities of using the proposed system in real-world applications, it is imperative that the observations resemble observations that can be produced by available data sources.

### 4.1.1. Simulating TSC behaviour

For the scope of this research, data that is used for the experiments is generated from simulations rather than collected from a real-world environment. This is done to have full control over the both the quality and the quantity of data that is used to generate models from, so that the effect of changes to either data quality or quantity can be investigated with as few external disturbances as possible.

TSC behaviour is simulated in MATLAB®. The behaviour is not influenced by actual measurements of vehicles, instead a Explicit Duration HSMM whose states represent the stages of a predefined TSC is used to generate a state-sequence, from which an observation sequence can be sampled.

The experiments described in this chapter will be performed for two types of TSC:

1. Vehicle-actuated TSCs without controller flexibility
2. Vehicle-actuated TSCs with controller flexibility

There is no separate model of a FTTSC, because the phasing behaviour of such TSCs is the same as that of the model without flexibility. Therefore it is expected that the results of the experiments performed with this model are also representative for FTTSCs.

A schematic overview of the simulated TSC without flexibility can be seen in Figure 4.1. The simulated TSC with flexibility is shown in Figure 4.2.

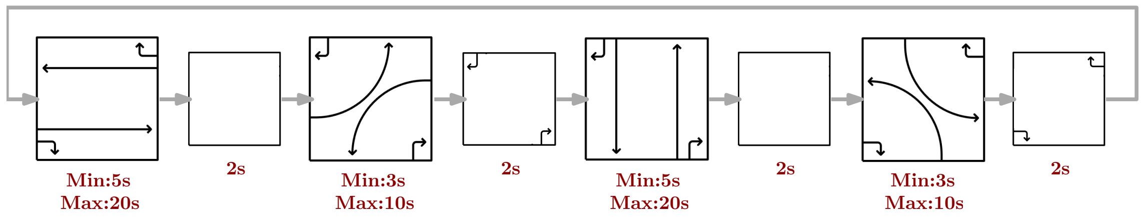


Figure 4.1: Source model for the data generated for all experiments without flexibility. The text below each block indicates its duration distribution. Each stage has a uniform distribution between the two noted extremes, whereas the transition phases have a set duration of 2 seconds.

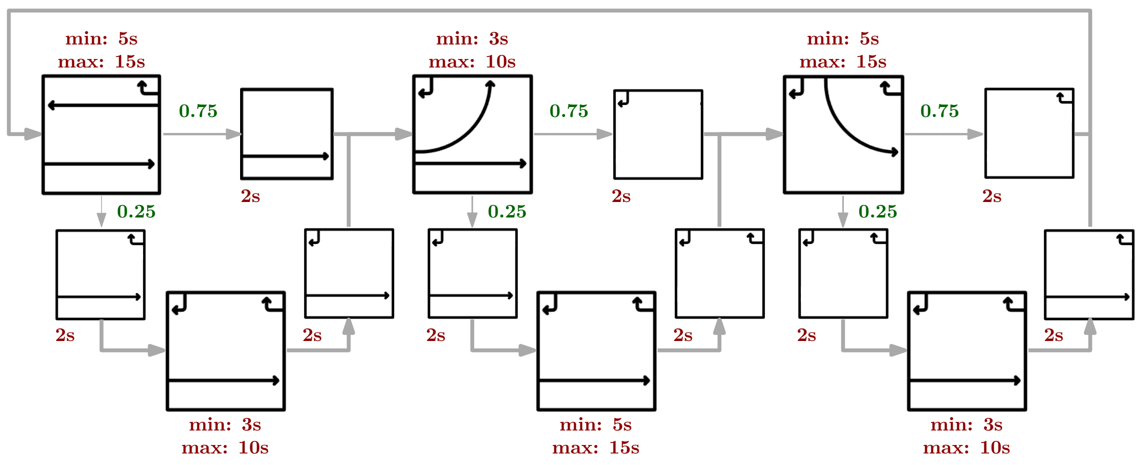


Figure 4.2: Source model for the data generated for all experiments with flexibility. The text in red next to each block indicates its duration distribution. Each stage has a uniform distribution between the two noted extremes, whereas the transition phases have a set duration of 2 seconds. If there are multiple transition possibilities from one state, the probability of each transition is noted in green.

The simulation models are used to generate state-sequences that represent the sequences of phases that are realised by true TSC. When the state-sequence has been sampled, the timing behaviour of the TSC is simulated by drawing state durations from distributions that are predefined for each state. It is chosen that the time-step-size is equal to 1 second. For the experiments in this section, all duration distributions are uniformly distributed between a minimum and a maximum value, which are stated in Figure 4.1 and Figure 4.2. States that represent a transition are assigned a fixed duration of two steps, equalling two seconds. It can be expected that transitions always have a fixed duration, because their sole purpose is to allow traffic from one movement to clear a conflict area before traffic from another movement can reach it. The time that this process takes only depends on the intersection layout, which is fixed, and is therefore calculated and set to a constant value by the programmer of the TSC.

True VATSCs with flexibility have the freedom to add or remove any signal group from the current or next stage while the current stage is active, as long as clearance and minimum-green times are respected. Therefore, it can be argued that the true behaviour of this type of TSC is more dynamic than that of the simulated TSC described above, because in the simulation, it is assumed that the phase remains the same during the whole stage, or until a flexibility phase becomes active. However, because it is defined that the intended HSMM that models the TSC behaviour has states that represent the full stages, it should not make a difference whether or not the combination of green signal-groups changes during the stage, as long as they remain a subset of the stage. Therefore, it is chosen to simulate the behaviour as described above.

To guarantee that the results of the experiments are reproducible even though random numbers are used to simulate TSC behaviour and initialize parameters, a random number generator is initialized in MATLAB®.

### 4.1.2. Simulating shared data

Given that the behaviour of TSCs is simulated without microscopically simulating road-users, the generation of shared data also has to be simulated. The main characteristics of shared data are the sampling rate and the penetration rate. For the experiments described in this chapter, the combination of both is simulated as follows: it is assumed that there is a traffic flow $q = 1000\frac{\text{veh}}{\text{h}\times\text{dir}}$ for each traffic direction, i.e., each leg of the intersection. The penetration rate $pr$, i.e., the probability that a given vehicle is connected is varied per experiment. Additionally, it is assumed that each vehicle on average spends $t = 10\frac{\text{s}}{\text{veh}}$ at the intersection, of combined waiting and traversing time. Also, connected vehicles are estimated to have a sampling frequency of 1Hz, i.e, $f = 1\frac{\text{obs}}{\text{s}}$, which means that each connected vehicle produces an observation for every second that it spends interacting with the intersection. Lastly, a time-step size of $\Delta_\text{t} = 1\text{s}$ is chosen for all experiments conducted for this research. The expected number of observations per time step can then be calculated using (4.1):

$$n_{obs} = pr \times q \times n \times \Delta_\text{t} \times t \times f \tag{4.1}$$

where $n_\text{dir}$ indicates the number of legs of the intersection. Using (4.1), the number of observations per timestep for the simulated TSC without flexibility shown in Figure 4.1, which controls an intersection with four legs, becomes:

$$n_{obs4.1} = pr \times \frac{1000}{3600}\left[\frac{\text{veh}}{\text{s}\times\text{dir}}\right] \times 4\,[\text{dir}] \times 1\left[\frac{\text{s}}{\Delta_\text{t}}\right] \times 10\left[\frac{\text{obs}}{\text{veh}}\right] \times 1\left[\frac{\text{obs}}{\text{s}}\right] = 11.11 \times pr\left[\frac{\text{obs}}{\Delta_\text{t}}\right] \tag{4.2}$$

Similarly, for the simulated TSC with flexibility shown in Figure 4.2, which controls an intersection with three legs, the number of observations per time step is calculated using (4.3)

$$n_{obs4.2} = 8.33 \times pr\left[\frac{\text{obs}}{\Delta_\text{t}}\right] \tag{4.3}$$

Remember that a data-point can either indicate that a signal is green or that there is traffic waiting for that signal. This means that the full observation-sequence contains one data-point for each second for each signal-group that is green or has waiting traffic at that second. The observation sequences used for the experiments are generated by randomly selecting a number of observations at

| Penetration rate | 1% | 2% | 5% | 10% | 25% | 50% | 75% | 100% |
|---|---|---|---|---|---|---|---|---|
| Average number of models initialized before finding correct model | 1000+ | 400 | 94 | 7.4 | 7.3 | 2.4 | 2.3 | 2.8 |
| Standard deviation over 10 tries | - | 360 | 150 | 6.4 | 3.7 | 1.9 | 1.3 | 2.7 |

Table 4.1: Number of models that had to be randomly initialized before a model equal to the ground truth model was found. The ground truth model was generated by training a model with the correct stages for the TSC without flexibility shown in Figure 4.1. The randomly initialized models were learned from the same sparse observation sequences. For each observation sequence, random initializations were produced until the correct model was found 10 times. For the sparsest observation sequence, with only 1% of the observations remaining, none of the 10x1000 initializations led to the correct model.

each time step from the full observation sequence, in accordance with (4.2) and (4.3) respectively. Note that this means that it is assumed that every movement is equally likely to produce an observation, while this is not true for most intersections, because some movements process a smaller percentage of the traffic flow than others. Additionally, observations are selected without replacement, meaning that each signal group can produce a maximum of one observation per time step to simulate that connected vehicles are distributed equally over the intersection.

## 4.2. Learning phasing behaviour

As stated in Section 3.1, one of the goals of this research is that the developed system can generate models without prior knowledge of the behaviour of the TSC. This means that it is unknown which stages the TSC uses in which ordering and whether it has flexibility. A 4 step approach was described in Section 3.6 of which the first step is to extract these features by initializing a model for each possible set of stages and learning all resulting models on the available observation sequence. The model that yields the highest likelihood is then selected and its transition matrix is analysed to identify the correct stage sequence. For both simulated TSCs, it is evaluated from which penetration rates the correct set of stages and its correct ordering are found.

This approach is compared with a more conventional approach of randomly initializing models with a number of states equal to the number of stages, which can be calculated by calculating the maximum conflict-group size. It is evaluated how many different random models must be initialized and learned before the correct stages are found. Only the parameters of **A** and **B** are randomly initialized. Those of **D** are uniformly initialized.

### 4.2.1. Results and discussion

The following results were acquired when trying to find the correct stages for the TSC without flexibility using random initializations. It can be seen in Table 4.1 that it takes an increasing amount of models to find the correct stages when observation sequences become sparser. For an observation sequence with only 1% data remaining, the correct model was not found after learning 10,000 randomly initialized models. Figure 4.3 shows the same results represented by a box-plot. Here it can be seen that it becomes harder to estimate how many models have to be initialized to be sure that the correct stages are found when sparser data is used. Given the mean and standard deviation from Table 4.1, it can be calculated that 1238 models would have to be initialized to achieve a 99% certainty that the correct stages are found if it is known that 2% of the data is available. However, as explained in Chapter 4, it is not possible to directly translate this to a penetration rate, nor does this particular example represent every possible real-life situation.

For the method described in Section 3.6, it can be seen in Figure 4.4 that the correct stages, which are in the third set of candidate stages, is identified correctly for higher data quantities. It also becomes clear that this approach can not distinguish the correct stages for lower data quantities without doubt. However, from 2% onwards, multiple sets of stages produce a model with clear transition behaviour, indicated by the low entropy. This means that each state has a significantly higher transition probability for just one of the other state. Using this information, new models

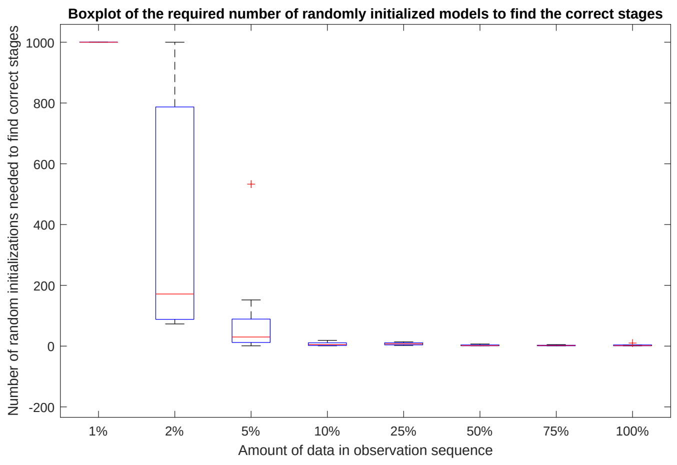


Figure 4.3: Boxplot of the values of Table 4.1

including transition states can be constructed and learned using the rest of the model generation system described in Section 3.6.

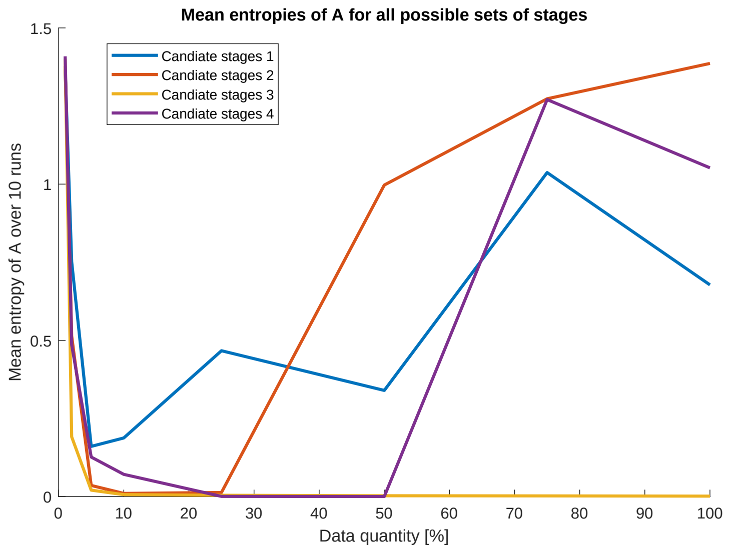


Figure 4.4: Mean entropies of learned **A** for all candidate sets of stages for the TSC without flexibility shown in Figure 4.1.

## 4.3. Learning timing behaviour

This section describes experiments to investigate under which circumstances the timing behaviour of a TSC can be learned by the proposed system. For these experiments, it is assumed that the phasing behaviour has been correctly identified, meaning that the parameters of **B** are as shown in Figure 3.10 and the state transitions correspond to the values shown in Figure 4.1 and Figure 4.2, where a grey arrow without value indicates a transition probability of 1. The parameters of **D** are uniformly initialized for all states over the full possible duration interval, which has size $D = 30$. This value is chosen for the maximum state duration because it is assumed that this forms a safe upper bound for the maximum duration of phases under the busy conditions that are modelled here.

Under the assumption that transition processes from one stage to another can be realistically expected to last only a couple of seconds, the choice could be made to initialize the duration distributions of states that represent a transition with a lower maximum duration. This would drastically reduce the number of non-zero parameters before learning and therefore improve learning performance. However, it is chosen to only investigate what happens if all states are initialized equally, because the risk of wrongly initializing parameters as zero should be avoided where possible. This experiment therefore aims to identify whether this can be avoided for the initialization of state duration distributions. More importantly, the effects of observation sequence length and data sparsity are investigated to get an indication of how much data is needed to learn the timing behaviour of each type of TSC. This information can be used to estimate whether it can realistically be expected that models of the behaviour of a given TSC during a certain part of the day can be learned.

The experiments for this section consist of four elements: both TSCs from Figure 4.1 and Figure 4.2 are evaluated twice. First, they are trained on increasingly long observation sequences. The shortest sequence is 900 seconds long, representing fifteen minutes of collected data. The sequence length is increased in steps of 900 seconds to a maximum length of 10,800 seconds, or three hours. It is expected that this is the maximum time during which data can realistically be collected for one model, given that it is expected that a separate model is needed for every quarter of an hour of each type of day. This means that three hours represents 12 weeks of data collection if a time frame is modelled that only occurs once per week. If data is collected for longer periods, seasonal influences start to play a role, meaning that models might no longer be representative. For this experiment, a fixed penetration rate of 10 % is used.

Secondly, the influence of the penetration rate on the learning of duration distributions is investigated. This is done by reducing the observation sequences in several steps to simulate the penetration and sampling rates, as described in Section 4.1.2. Again, both intersections are

Each sequence is used to learn the duration parameters of a model that has the correct phasing parameters defined, meaning that **A** and **B** are correctly initialized for the corresponding model. To ensure that the results have statistical meaning, each situation is simulated 25 times. For each simulation, a new behaviour is sampled and the resulting observation sequence is randomly reduced. The resulting duration distributions are compared per state with the true distributions by means of a Kullback-Leibler divergence, which provides a way to measure the difference between two distributions. It does so by comparing the entropy of both distributions. Not the actual value of this measure is of interest, but the relationship between penetration rate and change of the divergence. Note that the duration distributions of the sampled behaviour might differ from the true duration distribution of the underlying process, because the sample is of limited size.

### 4.3.1. Results and discussion

The results for the experiments investigating the relationship between learning the duration distributions of models and the time during which the modelled system is observed are discussed first. Figure 4.7 shows the results for the intersection without flexibility from Figure 4.1, while Figure 4.8

shows the results for the intersection with flexibility from Figure 4.2. Both experiments were performed with a penetration rate of 10%.

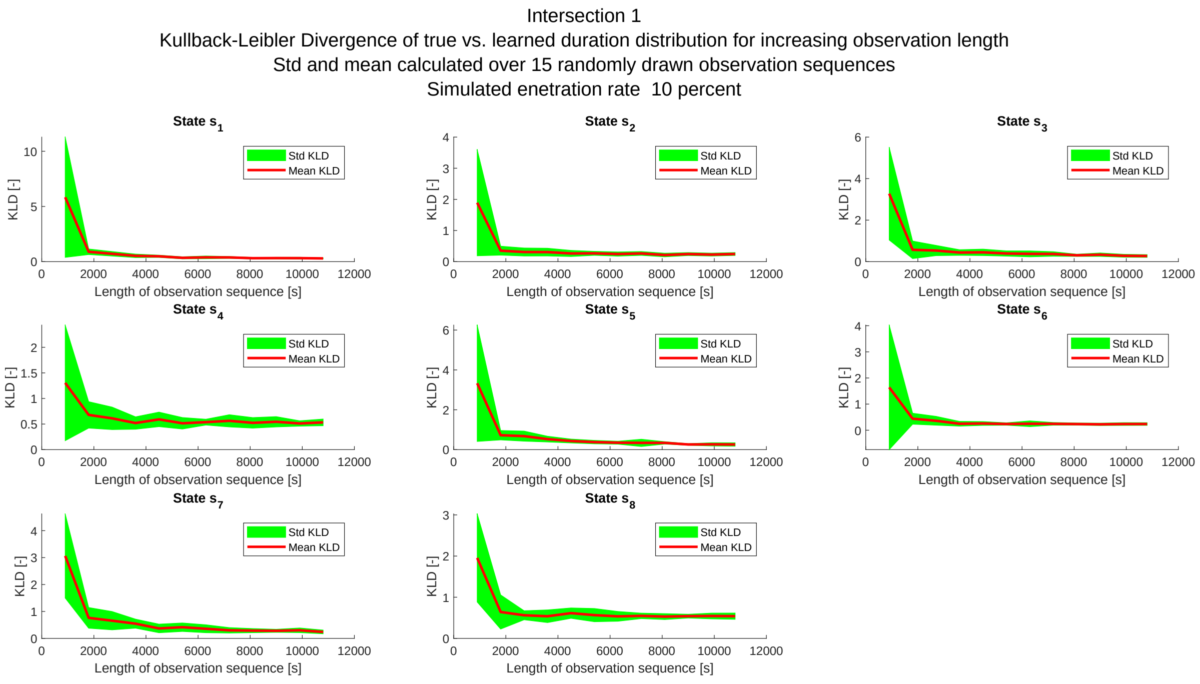


Figure 4.5: Kullback-Leibler divergence between the true and learned duration distributions for each state of example intersection 1 (Figure 4.1), for varying observation duration.

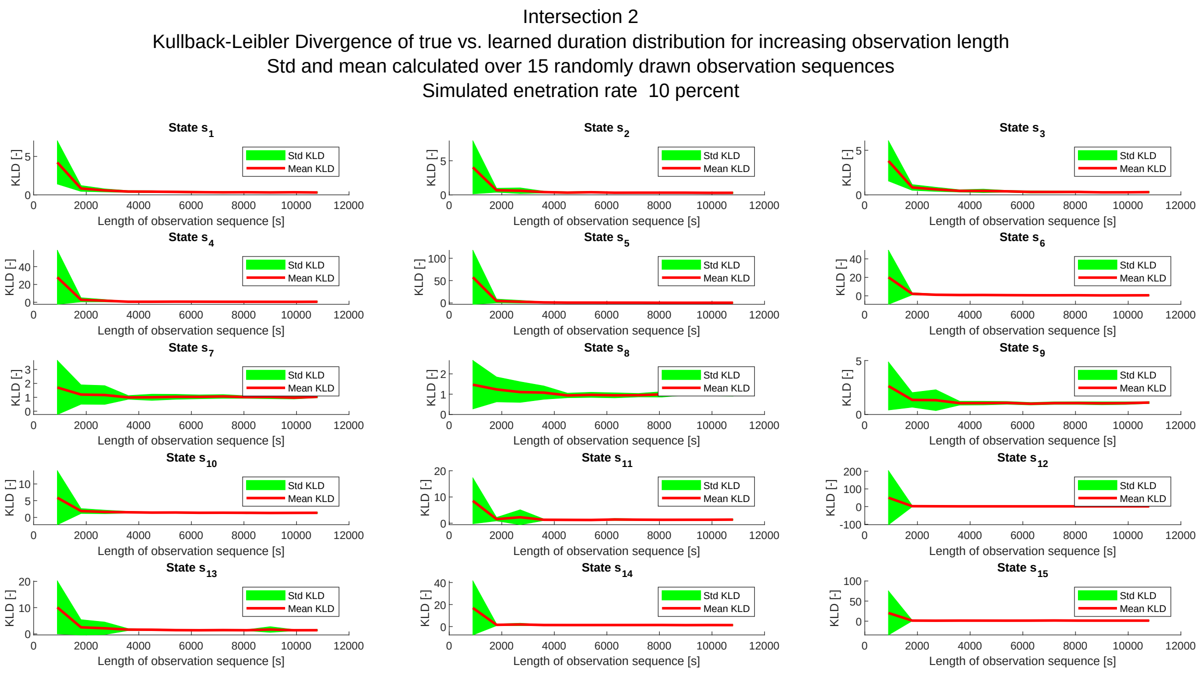


Figure 4.6: Kullback-Leibler divergence between the true and learned duration distributions for each state of example intersection 2 (Figure 4.2), for varying observation time.

It can be seen that models of both intersections converge to the true situation when the intersection is observed long enough. The turning point seems to lie at roughly one hour of observations, which seems like a duration that is short enough to allow for different models to be learned for one intersection to represent different traffic loads.

**Varying penetration rates** The results for the experiments investigating the relationship between learning the duration distributions of models and the density of the learning data are shown here. Figure 4.7 shows the results for the intersection without flexibility from Figure 4.1, while Figure 4.8 shows the results for the intersection with flexibility from Figure 4.2.

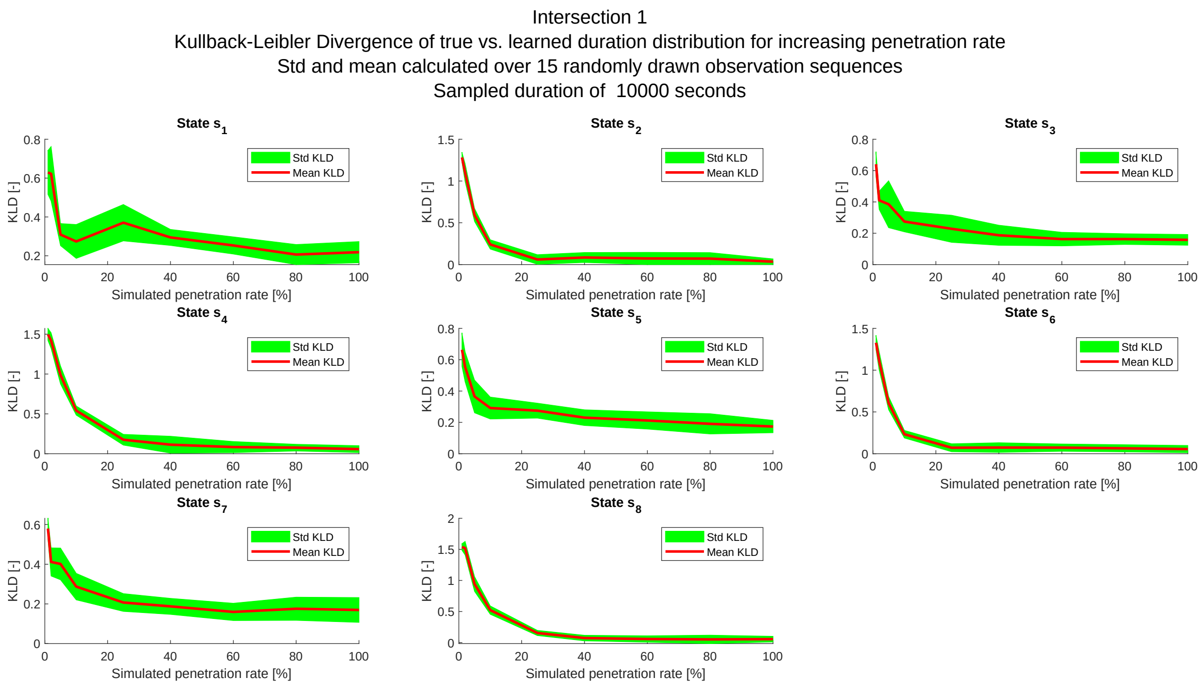


Figure 4.7: Kullback-Leibler divergence between the true and learned duration distributions for each state of example intersection 1 (Figure 4.2), for varying data densities. The densities indicate how much of the full observation sequence is used to learn the model parameters.

Figure 4.8 shows that for TSCs with flexibility, the duration distributions for less frequently visited states need more data to be learned. The clearest example of this can be given by comparing the transition states $s_7$ and $s_{10}$, who both origin from the same state $s_1$, but because the transition to $s_7$ is three times as likely, its duration distribution is learned more accurately for the same data density.

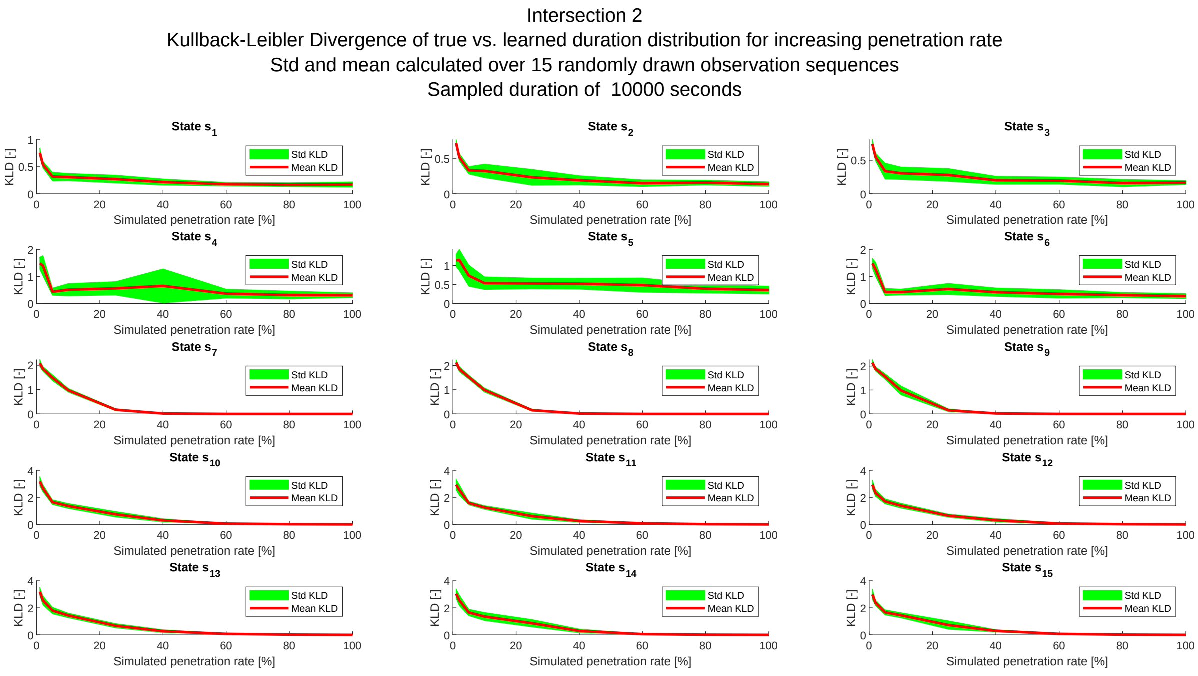


Figure 4.8: Kullback-Leibler divergence between the true and learned duration distributions for each state of example intersection 2 (Figure 4.2), for varying data densities. The densities indicate how much of the full observation sequence is used to learn the model parameters.

## 4.4. Usefulness of learned models

Given that the modelling method developed in this research intends to be used for GLOSA, it is important to get an indication of whether the models designed in Section 3.2 can be used to this end. This is determined by two properties of the models. Firstly, it is important that the resulting models can be used to estimate the current state of the TSC, because this determines under which assumptions a prediction about the future behaviour of the TSC is made and hence on what basis a speed advice is generated. Secondly, the capability of learned models to predict future behaviour is of interest. This is evaluated by comparing the entropy of correctly learned models with that of alternative models that were discarded during the design process described in Section 3.2.

## 4.5. State estimation

An advantage of the proposed modelling methodology is that the resulting models can estimate the current TSC-state when the true state of the TSC is only observed partially at infrequent intervals, as is the case when using shared data. Because HSMMs are generative, it is even possible estimate the state of the TSC if no data is available at all, but it highly depends on the predictability of the TSC to what extend the correct state sequence can be estimated. A correctly learned model of a FTTSC can predict the correct state infinitely far into the future without real-time data, assuming that there is no drift between the clock of the TSC and the clock of the model. As described in Section 2.1, the behaviour of a VATSC can approach that of a FTTSC during busy periods, because its detectors are constantly activated, meaning that its control actions will approach a constant cycle. At the other side of the spectrum, one can find that the behaviour of a VATSC during a period with little traffic can be almost impossible to predict, because of the irregular arrival of road-users, to which the TSC can immediately respond, leading to highly dynamic behaviour. This section investigates how well the state of a VATSC with flexibility can be estimated using a learned model of that TSC. The learning of the model is not considered for this experiment.

Two experiments are carried out to evaluate the ability of learned models to infer the correct states. The first experiment evaluates how well a full state sequence can be decoded for varying data densities. This can indicate whether the models can be used to label data. Labelled observation sequences, or correctly decoded state sequences could potentially be used to find correlations between, for example, the durations of subsequent phases. This information can be used construct more deterministic models of the behaviour of the TSC.

Secondly, it is investigated how well learned models can be used to estimate the state during run-time. This is an important aspect of generating GLOSA, because calculating probability of every possible next transition and transition moment depends on knowledge of the current state. This performance is evaluated by inferring the states for an observation sequence of growing length. The last state is compared with the true state at after each observation is added. The correctness of the state is defined by whether the correct phase is inferred and how far the past duration of that state is estimated.

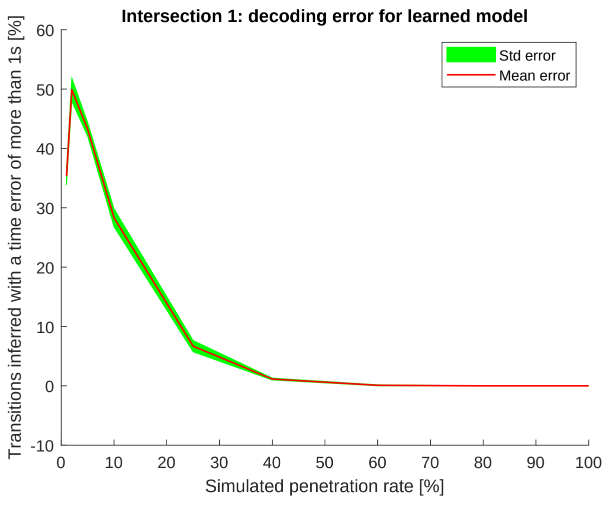


Figure 4.9: Decoding performance for a correctly learned model of the TSC without flexibility shown in Figure 4.1.

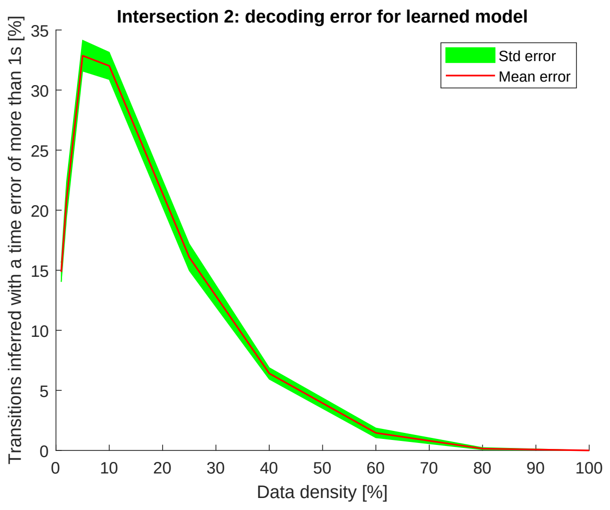


Figure 4.10: Decoding performance for a correctly learned model of the TSC wit flexibility shown in Figure 4.2.

Assume stable situation: 1 cycle past, start with first 50 elements of O, see how last state is estimated for growing O to represent real-time estimation. count errors for varying amounts of data. Also: how certain was estimation: likelihood.

## 4.6. Conclusion

This chapter described experiments to evaluate to what extend the designed system meets the design goals stated in Section 3.1.1. The experiments were performed in a simulate environment in MATLAB®and it was shown that for the specific setup, the phasing behaviour of TSCs could be identified under low penetration rates, even with random initializations. Additionally, it was shown that the learning of timing behaviour converged when data was collected for a long enough period of roughly one hour. However, it was shown that for VATSCs with flexibility, a penetration rate of 25% is needed for convergence, even when a long observation period is used.

# 5 Conclusions and recommendations

This research described the design of a system to model the behaviour of TSCs from FCD using Explicit Duration HSMMs. The following goals were set for this design:

**Goal 1** The developed system can generate models of:

- a: fixed-time TSCs;
- b: vehicle-actuated TSCs with fixed phasing;
- c: vehicle-actuated TSCs with flexibility.

**Goal 2** The developed system generates models whose states:

- a: cover all possible controller outputs;
- b: each represent one TSC state.

**Goal 3** The developed system generates models that represent phase transition behaviour as predictably as possible.

**Goal 4** The developed system generates models that explicitly model the duration probability distributions of all elements of the control structure of the modelled TSC.

**Goal 5** When provided with enough data, the developed system generates models whose state duration probability distributions converge to the true duration probability distributions of the TSC state that they represent.

**Goal 6** The developed system can generate models from data that is representative of FCD, i.e., data that:

- a: is sampled irregularly;
- b: only represents a part of the true TSC control output.

To achieve these goals, it was chosen to design a system that generates Explicit Duration HSMMs. An existing Forward-Backward parameter learning algorithm was adapted to be able to treat observations independently and allow time-steps without observations, which can realistically be expected when modelling from FCD. It was described how all possible outputs of the most common types of TSC can be represented by model states, but that this yields model states that are hard to find using conventional random initializations. Therefore, a model generation system was described that first identifies the control structure of the TSC and then adds states to represent its possible transitions and flexibility phases. This system was tested using simulated data to estimate under which conditions models converge to the true situation.

## 5.1. Conclusions

Experiments have shown that the proposed model generation method can learn the phasing and timing behaviour of simulated TSCs with and without flexibility. However, the results indicated that although the phasing behaviour can be found with certainty for relatively low penetration rates (10<%), the timing behaviour requires higher penetration rates (>25%) to be learned correctly. Still, it can be expected that the proposed methodology can be used to generate models that can be used to give GLOSA to road users under realistic conditions, since it was shown that the Kullback-Leibler divergence converges to a minimum after presenting data gathered over roughly one hour, which is an observation time that is not unrealistically large, for a penetration rate of ten percent, which is not unrealistically high. This means that the designed system shows that it has the potential to generate useful TSC behaviour models under realistic boundary conditions. However, it must be investigated how the assumptions under which this research was conducted limit the applicability of the system. To provide a starting point for this investigation, the next section will indicate which steps should be taken.

The main motivation for this research was to pave the way for large scale implementation of GLOSA systems. Although this goal was reckoned with throughout the design process, the combination with a speed advice algorithm has not yet been made. The GLOSA system in [22] is developed to explicitly cope with the inherent uncertainty of HMMs, therefore it can be expected that the models generated by the system developed in this thesis research can be used in combination with this GLOSA system. However, the models used in [22] are potentially more flexible, so it must be investigated how this influences the usefulness of the generated speed advice.

## 5.2. Recommendations

As explained in the previous section, there are three main hurdles to take from a development viewpoint before the model generation system can be used in practice. After which it can be ensured that the system can be used to generate reliable GLOSA if enough road users are willing to participate by sharing their data.

The first step would be to change the testing environment to a microscopic simulator to be able to test with data that is generated on a road user level, instead of on a macroscopic level. This step should also involve the investigation of the effects of the assumptions that were made in this research regarding data accuracy, i.e., the effects of noisy data on learning convergence must be investigated. Additionally, the influence of traffic with mixed modes and public transport pre-emption should be researched. This allows to investigate the effects of an uneven distribution of connected road users, which is necessary because it is likely that in real situations, certain movements have a higher penetration rate than others, because they are used by different kinds of traffic, both in terms of modality as in travel goal. It is imperative that many different situations and TSCs are simulated and modelled in this step, to ensure that situation-specific issues are found before the system is implemented in a real world environment. A system that works reliably from the start will namely attract more users and can therefore be expected to become even more reliable, since this means that more data is available to generate models from.

Secondly, the system should be tested in combination with the GLOSA system described in [22]. This would involve evaluating the quality of the generated speed advice and to determine whether it is sufficiently accurate.

Lastly, if it was shown that the system can reliably produce useful GLOSA under circumstances that accurately represent true road situations, the system can be tested in a real-world environment. This step requires the addition of a method to automatically switch to the static model that best represents the current traffic situation. A method to do so using HSMMs already exists for semistatic TSCs [39].

Alternatively, a three layer Hierarchical HMM could be used in the following way: The highest level describes the traffic situation (which is correlated to the time and type of day), the middle layer describes the active controller stages and the lowest level describes the actual combination of directions that can have a green signal. The reason for discriminating between the lower two levels is to make sure that each stage has a minimum duration and that observations that could belong to different stages are seen as flexibility or alternative realisations and not as a different stage. Additionally, this could help to keep track of the cycle-time, for which a separate duration distribution could be learned. The highest level allows there to be one model for the entire day, which can switch to the most representative sub-model depending on the real-time traffic situation and identified control actions, even if the traffic situation is somehow different from other days at a similar time, for example when there is a big yearly event.

If at any step it is found that the system is unable to meet the demands, there are some additions and changes that can be made to potentially increase the accuracy and thus usefulness of the resulting models:

1. In this research, data was not simulated to accurately represent the data sharing process. For example, a vehicle waiting for a red signal would realistically generate a sequence of observations of that signal being red, before a couple of observations that the signal is green (depending on the distance of the vehicle to the stopline and the temporal resolution of the model). It could be that this allows the model to learn a dependency between the time that a vehicle is or is not present and the moment of switching. Previous research [25, 26] claims that for VATSCs, this dependency exists and influences the behaviour. It should be investigated how this dependency can be captured using the described modelling method or whether a useful GLOSA system can be developed without it. Maximum Entropy Markov Models [63] could prove useful to this end, although these require ground truth data to be known. Potentially, ground truth data can be approached by a state sequence generated using the method proposed in this research.

2. Investigate whether parametric duration distributions can improve learning duration behaviour for low data densities. It can be expected that this is the case, because the number of parameters will be greatly reduced. Real data must be analysed in order to identify which parametric distribution is best. For transition states use the mean duration, because it can be expected that transition states have an almost fixed duration, meaning they would need only one parameter to describe their duration distribution.

3. Investigate how state definition refinement can be used to lower the entropy of resulting models. The goal is to maximize the information in state duration distributions, without creating too many states. This is done in literature using state splitting [56]. If a state's duration distribution has two peaks for example, split it up into two states with both one peak. Various measures can be used to base a state split on, such as: elapsed state duration, previous state, previous state duration, cycle duration, pending request for one of the other movements, behaviour of nearby intersections. Methods to find the best splits exist in literature [56].

4. investigate usefulness of defining a different Hierarchical HMM , where each movements has its own sub-model. This makes it possible to define a guaranteed green time/maximum green time per movement. However, using states that contain multiple directions is still the easiest way to define some causality between certain signals being green and others being green if observations are sparse.

5. Several researches mention that installed loop detectors commonly fail to register vehicles or produce ghost observations [25, 26], this can lead to a discrepancy between the inferred state

and the TSC-state, because the inferred state might be based on the observation of waiting traffic for a certain signal-group, while the TSC has not measured the presence of this traffic and therefore has not adapted its control action to it, whereas the model assumes the traffic is waiting because other conflicting signal-groups have a green signal. This

# A
# Traffic signal control

## A.1. Intersection movement numbering

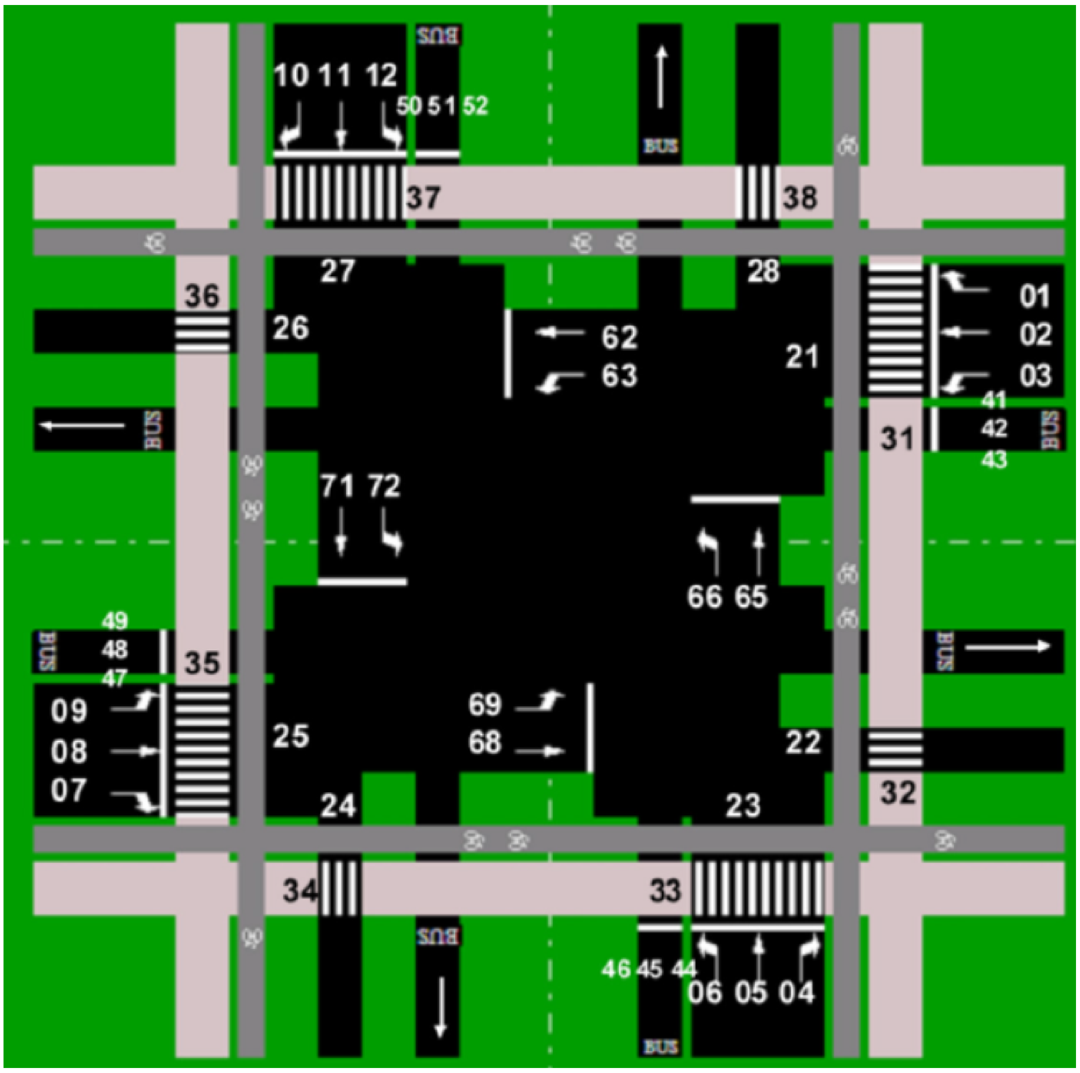


Figure A.1: Numbering of movements according to the Dutch standard [30]. The top of the figure is oriented towards the North.

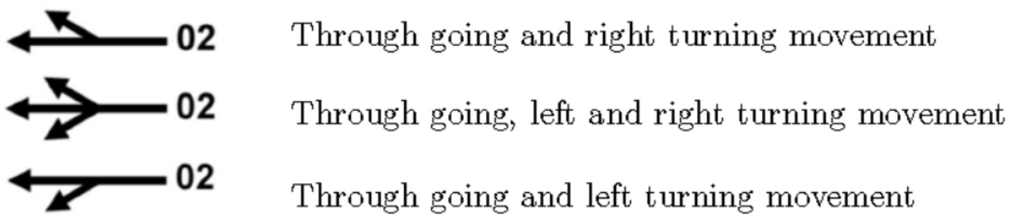

02
Through going and right turning movement
02
Through going, left and right turning movement
02
Through going and left turning movement

Figure A.2: Numbering of combined movements.[30]

# B Hidden Markov Modelling

## B.1. Solutions to the three basic problems of HMMs

This section shows the solutions to the three basic problems commonly described for HMMs. All theory in this section is based on the paper by Rabiner [19].

### B.1.1. The evaluation problem

Problem 1 is solved using the Forward algorithm. The forward variable $\alpha_t(i)$ represents the likelihood of partial observation sequence $o_1 o_2 ... o_t$ up to time $t$ and state $S_i$ at time $t$ for model parameters $\lambda$

$$\alpha_t(i) = P[o_1 o_2 ... o_t, q_t = s_i | \lambda] \tag{B.1}$$

This variable is used in three steps to solve problem 1 by calculating $P[O|\lambda]$:

**Step 1** Initialize $\alpha_1(i)$ as the probability of observing $o_1$ from any of the $N$ states:

$$\alpha_1(i) = \pi_i b_i(o_1), \quad 1 \leq i \leq N \tag{B.2}$$

**Step 2** For all remaining time-steps, calculate $\alpha_{t+1}(j)$ for each possible next state $s_j$ as the joint probability of making observation $o_t$ from $s_j$ and the sum over the probabilities of transitioning to state $s_j$ from each state $s_i$ multiplied by $\alpha_t(j)$:

$$\alpha_{t+1}(j) = \left[ \sum_{i=1}^{N} \alpha_i(t) a_{ij} \right] b_j(o_{t+1}), \quad \begin{matrix} 1 \leq t \leq T-1 \\ 1 \leq j \leq N \end{matrix} \tag{B.3}$$

**Step 3** Calculate the final $P[O|\lambda]$ by summing over all forward variables found for $t = T$:

$$P[O|\lambda] = \sum_{i=1}^{N} \alpha_T(i) \tag{B.4}$$

A close look at the Forward algorithm learns that it has a time complexity of $\mathcal{O}(TN^2)$ per calculation of $P[O|\lambda]$. The output can be used to compare different models, the model that shows the highest likelihood of $O$ best suits the observation sequence. It is important to note that the model with the highest likelihood is not necessarily the best model if that model has an extensive amount of parameters. Models with more parameters are more flexible and therefore better able to represent a dataset, but also more prone to overfitting. **??** describes model selection methods that make a trade-off between likelihood and flexibility.

### B.1.2. The decoding problem

The Viterbi algorithm is used to solve problem 2. Other algorithms may be used depending on how the "best" state sequence is defined. The Viterbi algorithm maximizes the probability of observing the full state sequence for a given model $\lambda$ and $O$, so that both the observation emission and the state transition probabilities are accounted for. This equals maximizing $P[Q|O,\lambda]$, or $P[Q,O|\lambda]$. The Viterbi algorithm is similar in structure to the forward algorithm and therefore also has a time complexity of $\mathcal{O}(TN^2)$. The algorithm consists of the following steps:

1. Initialize the variables $\delta$ and $\phi$:

$$\delta_1(i) = \pi_i b_i(O_1), \quad 1 \le i \le N \tag{B.5}$$

$$\phi_1(i) = 0, \quad 1 \le i \le N \tag{B.6}$$

2. Recursively update $\delta_t$ for each $t$ by finding the most probable 2-state sequence and store the argument of this sequence in $\phi$:

$$\delta_{t+1}(j) = \max_{1\le i\le N}[\delta_t(i)a_{ij}]b_j(O_{t-1}), \quad \begin{matrix} 1 \le t \le T-1 \\ 1 \le j \le N \end{matrix} \tag{B.7}$$

$$\phi_{t+1}(j) = \operatorname*{argmax}_{1\le i\le N}[\delta_t(i)a_{ij}], \quad \begin{matrix} 1 \le t \le T-1 \\ 1 \le j \le N \end{matrix} \tag{B.8}$$

3. Define the probability of the most likely sequence as $P^*$ and the most likely final state as $q_T^*$:

$$P^* = \max_{1\le i\le N}[\delta_T(i)] \tag{B.9}$$

$$q_T^* = \operatorname*{argmax}_{1\le j\le N}[\delta_T(i)] \tag{B.10}$$

4. Find the complete state sequence by backtracking from $q_T^*$ to $q_1^*$:

$$q_t^* = \phi_{t+1}(q_{t+1}^*), \quad t = T-1, T-2, ..., 1 \tag{B.11}$$

### B.1.3. The learning problem

Finally, the most common method to solve problem 3 is the Baum-Welch algorithm. No analytical methods exist to find the optimal parameters $\lambda$, so the Baum-Welch algorithm is an iterative method. The Baum-Welch algorithm is essentially an expectation maximization (EM) algorithm and it has been shown that after each iteration, the new parameters $\lambda$ have at least the same probability of producing $O$, thus the algorithm converges to an optimum. Three new variables need to be defined to implement the algorithm.

1. The backward variable $\beta$ is defined as:

$$\beta_t(i) = P[O_{t+1}O_{t+2}...O_T|q_t = S_i, \lambda] \tag{B.12}$$

and is calculated in a similar manner to the forward variable:

$$\beta_T(i) = 1, \quad 1 \le i \le N \tag{B.13}$$

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij}b_j(O_{t+1})\beta_{t+1}(j), \quad \begin{matrix} t = T-1, T-2, ..., 1 \\ 1 \le i \le N \end{matrix} \tag{B.14}$$

2. $\gamma$ gives the probability of each individual state at time $t$ for given $O$ and $\lambda$ and is defined as:

$$\gamma_t(i) = P[q_t = S_i | O, \lambda] \tag{B.15}$$

and is calculated as a normalized variable, using the forward and backward variables as:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} \tag{B.16}$$

3. $\xi$ gives the probability of being in state $S_i$ at $t$ and being in state $S_j$ at $t+1$:

$$\xi_t(ij) = P[q_t = S_i, q_{t+1} = S_j | O, \lambda] \tag{B.17}$$

and is calculated using the forward and backward variables as:

$$\xi_t(ij) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)} \tag{B.18}$$

The updated parameters $\bar{\lambda} = \{\bar{\pi}, \bar{A}, \bar{B}\})$ can then be interpreted and calculated as follows:

$$\bar{\pi}_i = \textbf{expected probability of starting in state } S_i = \gamma_1(i) \tag{B.19}$$

$$\bar{a}_{ij} = \frac{\textbf{expected number of transitions from } S_i \textbf{ to } S_j}{\textbf{expected total number of transitions from state } S_i} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{B.20}$$

$$\begin{aligned} \bar{b}_j(k) &= \frac{\textbf{expected number of times in state } S_j \textbf{ and observing } v_k}{\textbf{expected number of times in state } S_j} \\ &= \frac{\sum_{t=1}^{T} \gamma_t(j), \text{ s.t. } O_t = v_k}{\sum_{t=1}^{T} \gamma_t(j)} \end{aligned} \tag{B.21}$$

## B.2. Dependent observations

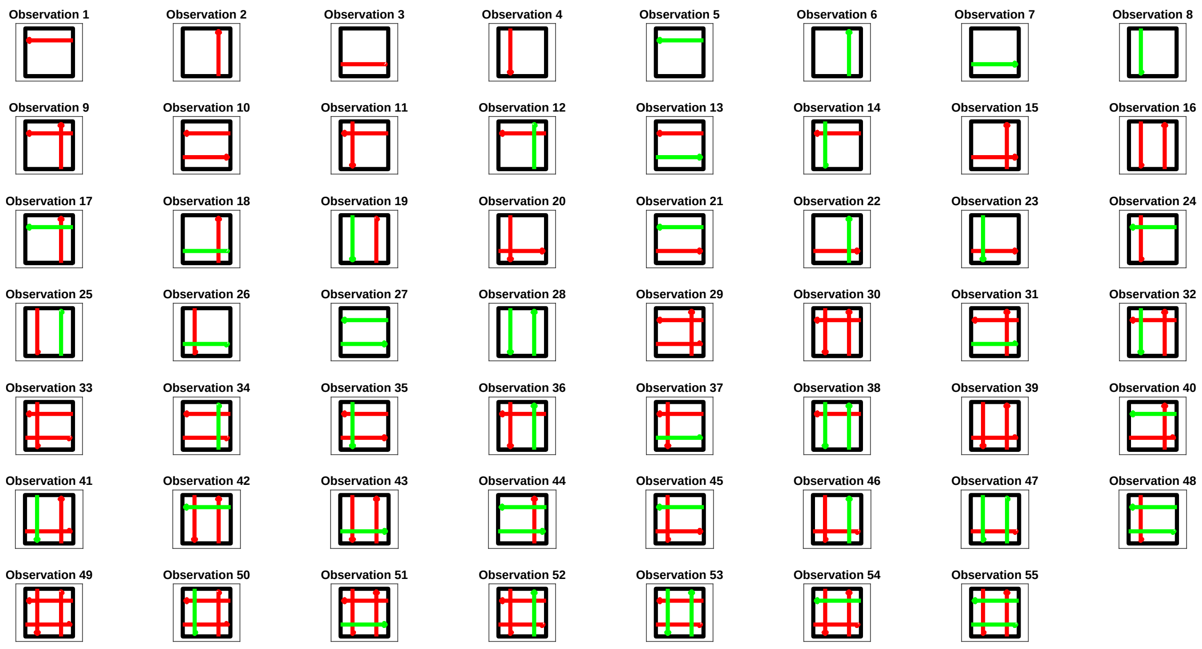


Figure B.1: All observations for an intersection with movements 2, 5, 8 and 11.

# Bibliography

[1] K. Yuan, V. L. Knoop, and S. P. Hoogendoorn, “Capacity Drop: Relationship between Speed in Congestion and the Queue Discharge Rate,” Transportation Research Record: Journal of the Transportation Research Board, vol. 2491, pp. 72–80, jan 2015.

[2] I. Iglesias, L. Isasi, M. Larburu, V. Martinez, and B. Molinete, “I2V communication driving assistance system: On-board Traffic Light Assistant,” in IEEE Vehicular Technology Conference, 2008.

[3] T. Tielert, M. Killat, H. Hartenstein, R. Luz, S. Hausberger, and T. Benz, “The impact of traffic-light-to-vehicle communication on fuel consumption and emissions,” in 2010 Internet of Things, IoT 2010, 2010.

[4] K. Katsaros, R. Kernchen, M. Dianati, and D. Rieck, “Performance study of a Green Light Optimized Speed Advisory (GLOSA) application using an integrated cooperative ITS simulation platform,” in IWCMC 2011 - 7th International Wireless Communications and Mobile Computing Conference, pp. 918–923, 2011.

[5] C. Raubitschek, N. Schütze, E. Kozlov, and B. Bäker, “Predictive driving strategies under urban conditions for reducing fuel consumption based on vehicle environment information,” in 2011 IEEE Forum on Integrated and Sustainable Transportation Systems, FISTS 2011, pp. 13–19, 2011.

[6] H. Xia, K. Boriboonsomsin, F. Schweizer, A. Winckler, K. Zhou, W. B. Zhang, and M. Barth, “Field operational testing of ECO-approach technology at a fixed-time signalized intersection,” in IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC, pp. 188–193, 2012.

[7] D. Eckhoff, B. Halmos, and R. German, “Potentials and limitations of Green Light Optimal Speed Advisory systems,” in IEEE Vehicular Networking Conference, VNC, pp. 103–110, 2013.

[8] M. Treiber and A. Kesting, “Automatic and efficient driving strategies while approaching a traffic light,” in 17th International IEEE Conference on Intelligent Transportation Systems (ITSC), 2014.

[9] R. Bodenheimer, A. Brauer, D. Eckhoff, and R. German, “Enabling GLOSA for adaptive traffic lights,” in IEEE Vehicular Networking Conference, VNC, vol. 2015-Janua, pp. 167–174, IEEE Computer Society, jan 2015.

[10] V. Sokolov, D. W. Etherington, C. Schmid, D. Karbowski, A. Rousseau, and M. Imran, “Effects of Predictive Real-Time Traffic Signal Information,” 2018 21st International Conference on Intelligent Transportation Systems (ITSC), jul 2018.

[11] S. Ibrahim, D. Kalathil, R. O. Sanchez, and P. Varaiya, “Estimating Phase Duration for SPaT Messages,” IEEE Transactions on Intelligent Transportation Systems, vol. 20, pp. 2668–2676, jul 2019.

[12] AASHTO, “National Connected Vehicle Field Infrastructure Footprint Analysis Final Report,” tech. rep., jun 2014.

[13] Siemens Mobility, “Actieve VRI’s in Nederland,” 2021.

[14] S. A. Fayazi, A. Vahidi, G. Mahler, and A. Winckler, “Traffic signal phase and timing estimation from low-frequency transit bus data,” IEEE Transactions on Intelligent Transportation Systems, vol. 16, pp. 19–28, feb 2015.

[15] S. Axer and B. Friedrich, “Signal timing estimation based on low frequency floating car data,” in Transportation Research Procedia, vol. 25, pp. 1645–1661, Elsevier B.V., jan 2017.

[16] J. Yu and P. Lu, “Learning traffic signal phase and timing information from low-sampling rate taxi GPS trajectories,” Knowledge-Based Systems, vol. 110, pp. 275–292, oct 2016.

[17] M. Kerper, C. Wewetzer, A. Sasse, and M. Mauve, “Learning traffic light phase schedules from velocity profiles in the cloud,” in 2012 5th International Conference on New Technologies, Mobility and Security - Proceedings of NTMS 2012 Conference and Workshops, 2012.

[18] A. Wilson, R. Hornman, V. Bronkhorst, R. de Bruin, G. van der Burgt, G. van Dijck, E. Greweldinger, M. Kant, P. J. Kleevens, E. van de Laak, B. de Leeuw, W. Mak, and M. Salomons, Handboek verkeerslichtenregelingen 2014. CROW, 2014.

[19] L. R. Rabiner, “A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition,” Proceedings of the IEEE, vol. 77, no. 2, pp. 257–286, 1989.

[20] C. Menig, R. Hildebrandt, and R. Braun, “Der informierte Fahrer - Optimierung des Verkehrsablaufs durch LSA-Fahrzeugkommunikation,” in HEUREKA ’08 - Optimierung in Verkehr und Transport, Köln: FGSV Verlag, 2008.

[21] M. Reisi Gahrooei and D. B. Work, “Inferring traffic signal phases from turning movement counters using hidden Markov models,” IEEE Transactions on Intelligent Transportation Systems, vol. 16, pp. 91–101, feb 2015.

[22] A. Dabiri, A. Hegyi, and B. Goñi-Ros, “Optimal Speed Advice for Cyclists using a Roadside Sign at Signalized Intersections with Uncertainty in Traffic Light Timing,” Transportation Research Record: Journal of the Transportation Research Board, vol. 2673, pp. 239–247, jul 2019.

[23] C. Wang and S. Jiang, “Traffic signal phases’ estimation by floating car data,” in 2012 12th International Conference on ITS Telecommunications, ITST 2012, pp. 568–573, 2012.

[24] Y. Zhao, Y. Zhang, T. Yu, T. Liu, X. Wang, and X. Liu, “CityDrive: A Map-Generating and Speed-Optimizing Driving System,” in IEEE INFOCOM 2014 - IEEE Conference on Computer Communications, 2014.

[25] R. Bodenheimer, Entwicklung und Analyse eines Prognoseverfahrens für verkehrs-adaptive Lichtsignalanlagen (Development and Analysis of a Prediction for Traffic-Adaptive Intersection Controllers). PhD thesis.

[26] R. Scheepjens, Algorithm Design for Traffic Signal Timings Predictions of Vehicle-Actuated Intersections using Support Vector Regression. PhD thesis, 2016.

[27] Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, pp. 436–444, may 2015.

[28] S. Z. Yu, Hidden Semi-Markov Models: theory, algorithms and applications. Elsevier, 1st ed., 2015.

[29] H. H. Avilés-Arriaga, L. E. Sucar, C. E. Mendoza, and B. Vargas, “Visual recognition of gestures using dynamic naive Bayesian classifiers,” Proceedings - IEEE International Workshop on Robot and Human Interactive Communication, no. January, pp. 133–138, 2003.

[30] V. L. Knoop, A. Hegyi, M. Salomons, H. van Lint, Y. Yuan, and R. Landman, “Lecture Notes: Traffic Flow Modelling and Control,” 2019.

[31] D. Robertson, “TRANSYT: a traffic network study tool,” tech. rep., TRRL, Crawthorne, 1969.

[32] T. Urbanik, A. Tanaka, B. Lozner, E. Lindstrom, K. Lee, S. Quayle, S. Beaird, S. Tsoi, P. Ryus, D. Gettman, S. Sunkari, K. Balke, and D. Bullock, Signal Timing Manual - Second Edition. Transportation Research Board, 2nd ed., sep 2015.

[33] M. Barthauer and B. Friedrich, “Evaluation of a signal state prediction algorithm for car to infrastructure applications,” in Transportation Research Procedia, vol. 3, pp. 982–991, Elsevier, jan 2014.

[34] A. G. Sims and K. W. Dobinson, “The Sydney Coordinated Adaptive Traffic (SCAT) System Philosophy and Benefits,” IEEE Transactions on Vehicular Technology, vol. 29, no. 2, pp. 130–137, 1980.

[35] D. I. Robertson, “Research on the TRANSYT and SCOOT Methods of Signal Coordination,” ITE Journal, vol. 56, no. 1, pp. 36–40, 1986.

[36] Y. Zhao and Z. Tian, “An Overview of the Usage of Adaptive Signal Control System in the United States of America,” Applied Mechanics and Materials, vol. 178, no. 1, pp. 2591–2598, 2012.

[37] C. Dittrich, J. Lüssmann, F. Busch, R. Mänz, R. Stahlmann, and C. Menig, “Greener Roads by Talking Traffic Lights: Knowledge about Queue Length and Upcoming Traffic Light Signal,” in 2nd International Conference on Sustainable Automotive Technologies (ICSAT 2010), no. February 2010, (Chieming, Germany), 2010.

[38] C. Dittrich, R. Stahlmann, and F. Busch, “Refinement of Urban Traffic State Estimation by Using Queue Length Information,” in 5th International Symposium FOVUS - Networks for Modbility, (Stuttgart, Germany), 2010.

[39] E. Ozatay, U. Ozguner, D. Filev, and J. Michelini, “Bayesian Traffic Light Parameter Tracking Based on Semi-Hidden Markov Models,” IEEE Transactions on Intelligent Transportation Systems, vol. 17, pp. 2998–3008, nov 2016.

[40] G. Hoffmann, “Geschwindigkeitsempfehlungen im Kraftfahrzeug - Ein Beitrag zur Kraftstoffeinsparung durch das Informationssystem "Wolfsburger Welle",” Strassenverkehrstechnik, vol. 35, no. 5, 1989.

[41] C. Duivenvoorden, “Roadside versus in-car speed support for a green wave : a driving simulator study,” tech. rep., Universiteit Twente, Enschede, 2007.

[42] G. Kent and O. Choudhry, “EcoDrive II: I2V connected vehicle pilot project,” tech. rep., City of Ottowa, Ottowa, may 2020.

[43] M. Zweck and M. Schuch, “Traffic light assistant: Applying cooperative ITS in European cities and vehicles,” in 2013 International Conference on Connected Vehicles and Expo, ICCVE 2013 - Proceedings, pp. 509–513, IEEE Computer Society, 2013.

[44] V. Protschky, K. Wiesner, and S. Feit, “Adaptive traffic light prediction via Kalman filtering,” in IEEE Intelligent Vehicles Symposium, Proceedings, pp. 151–157, Institute of Electrical and Electronics Engineers Inc., 2014.

[45] O. Altintasi, H. Tuydes-Yaman, and K. Tuncay, “Detection of urban traffic patterns from Floating Car Data (FCD),” in Transportation Research Procedia, vol. 22, pp. 382–391, Elsevier B.V., jan 2017.

[46] Flitsmeister, “Flitsmeister.nl - Floating Car Data via Flitsmeister,” 2021.

[47] M. Rahmani and H. N. Koutsopoulos, “Path inference from sparse floating car data for urban networks,” Transportation Research Part C: Emerging Technologies, vol. 30, pp. 41–54, may 2013.

[48] P. Ranacher, R. Brunauer, S. Van Der Spek, and S. Reich, “What is an appropriate temporal sampling rate to record floating car data with a GPS?,” ISPRS International Journal of Geo-Information, vol. 5, jan 2016.

[49] S. Wang, X. Zhang, F. Li, P. S. Yu, and Z. Huang, “Efficient Traffic Estimation with Multi-Sourced Data by Parallel Coupled Hidden Markov Model,” in IEEE Transactions on Intelligent Transportation Systems, vol. 20, pp. 3010–3023, Institute of Electrical and Electronics Engineers Inc., aug 2019.

[50] J. Campbell, H. B. Amor, M. H. Ang, and G. Fainekos, “Traffic light status detection using movement patterns of vehicles,” in IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC, pp. 283–288, Institute of Electrical and Electronics Engineers Inc., dec 2016.

[51] E. Koukoumidis, L. S. Peh, and M. R. Martonosi, “SignalGuru: Leveraging mobile phones for collaborative traffic signal schedule advisory,” in MobiSys’11 - Compilation Proceedings of the 9th International Conference on Mobile Systems, Applications and Services and Co-located Workshops, (New York, New York, USA), pp. 127–140, ACM Press, 2011.

[52] Traffic Technology Today, “EXCLUSIVE: How New York City traffic-signal data is being harvested with no direct DOT input,” 2018.

[53] J. Ding, V. Tarokh, and Y. Yang, “Model Selection Techniques: An Overview,” IEEE Signal Processing Magazine, vol. 35, pp. 16–34, nov 2018.

[54] S. A. Mulaik, L. R. James, J. Van Alstine, N. Bennett, S. Lind, and C. D. Stilwell, “Evaluation of Goodness-of-Fit Indices for Structural Equation Models,” Psychological Bulletin, vol. 105, no. 3, pp. 430–445, 1989.

[55] M. Costa and L. D. Angelis, “Model selection in hidden Markov models : a simulation study,” Quaderni di Dipartimento, 2010.

[56] S. M. Siddiqi, G. J. Gordon, and A. W. Moore, “Fast State Discovery for HMM Model Selection and Learning,” tech. rep., mar 2007.

[57] S. I. Vrieze, “Model selection and psychological theory: A discussion of the differences between the Akaike information criterion (AIC) and the Bayesian information criterion (BIC),” Psychological Methods, vol. 17, pp. 228–243, jun 2012.

[58] S. Z. Yu and H. Kobayashi, “Practical implementation of an efficient forward-backward algorithm for an explicit-duration hidden Markov model,” IEEE Transactions on Signal Processing, vol. 54, pp. 1947–1951, may 2006.

[59] TomTom, “HD Map with RoadDNA: High definition map with sensor-agnostic localization,” 2019.

[60] T. Mäkinen and O. Hway-Liem, “Automatic enforcement of speed and red light violations.” 1992.

[61] K. Shaaban and A. Pande, “Evaluation of red-light camera enforcement using traffic violations,” Journal of Traffic and Transportation Engineering (English Edition), vol. 5, pp. 66–72, feb 2018.

[62] S. Z. Yu, “Hidden semi-Markov models,” Artificial Intelligence, vol. 174, pp. 215–243, feb 2010.

[63] A. Mccallum, D. Freitag, and F. Pereira, “Maximum Entropy Markov Models for Information Extraction and Segmentation,” tech. rep.