BACHELOR

Platoon Forming Algorithms for an Intersection with Mixed Traffic

Bouwmeesters, Eline C.M.

Award date:
2022

Link to publication
Platoon Forming
Algorithms for an Intersection with Mixed Traffic

Bachelor Final Project

E.C.M. Bouwmeesters

Supervisors:

R.W. Timmerman
M.A.A. Boon

Eindhoven, Monday 11th April, 2022
## Contents

Contents

1 Introduction 1
2 Literature overview 2
3 Model Description 6
4 Scheduling Algorithm 8
5 Trajectories 14
   5.1 Regular Vehicles 15
   5.2 Automated Vehicles 18
   5.3 closed-form solution for exhaustive policy 19
      5.3.1 MotionBest 20
      5.3.2 MotionSelf 24
      5.3.3 Mixed Traffic 26
6 Results 28
   6.1 Trajectories 29
   6.2 Influence CAVs with fixed-cycle switching process 31
   6.3 Exhaustive scheduling algorithm 34
7 Conclusion and discussion 37

References 40
1 INTRODUCTION

With the increasing number of vehicles used for transport, traffic management has become more complex throughout the years. A higher traffic load can result in excessive queues, which in turn cause economic, social and environmental problems [1, 2]. Efficiently managing intersections could contribute to reducing problems caused by traffic jams. The common occurrence of these problems are reason enough for a plethora of research on this topic. However, while classic traffic is still analyzed, advanced driver assistance systems have emerged with vehicle developments. Although current driver assistance systems aim to automate certain tasks, such as Adaptive Cruise Control (ACC), the final goal is to achieve fully Connected and Automated Vehicles (CAVs) that do not require any human interaction or interference to drive. The discussion is ongoing when CAVs will be on the road, but it is expected that CAVs will take over future traffic. However, between this future concept and the current traffic composition, there will be a transition phase, during which traffic will consist of both CAVs and Regular Vehicles (RVs). Throughout this period with mixed traffic, reducing traffic congestion will still be of importance. Researching how to maximize the efficiency of intersections requires a model to describe the behaviour of the approaching vehicles. Such models exist for exclusively RVs or CAVs, yet they do not capture the interaction between the RVs and CAVs. This paper therefore presents a model for the behaviour of both RVs and CAVs approaching an intersection with mixed traffic. Furthermore, we will propose an algorithm to efficiently manage an intersection with mixed traffic.

Numerous research has been performed to analyze and optimize traffic consisting exclusively of CAVs. CAVs would enable communication between vehicles and traffic systems, which in turn could facilitate vehicle planning in order to optimize traffic intersections. Several studies have shown significant improvement in traffic delay at an intersection using algorithms to schedule CAVs [3, 4]. The communication between CAVs and traffic systems allows CAVs to adapt their driving trajectories in order to cross the intersection at their scheduled times. In addition to reduced traffic congestion, CAVs can contribute to an increase of safety, and a decrease of energy consumption and emission [5]. As opposed to CAVs, RVs do not communicate with traffic systems, but adapt their behaviour to the colour of the traffic light. This difference in behaviour and its interactions, cannot be captured by literature on exclusively RVs or CAVs. Furthermore, algorithms to manage a traffic intersection with RVs, do not use the properties of CAVs to their advantage. On the other hand, algorithms that schedule CAVs at an intersection, assume all vehicles to have CAV-properties. Both types of traffic control systems are therefore not completely suitable to use at an intersection with mixed traffic. Hence, it is important to extend the existing literature on mixed traffic by constructing a model to describe the behaviour of RVs and CAVs, as well as an algorithm to efficiently manage an intersection with such traffic. We will answer the following questions:

- How can we describe the driving trajectories of RVs and CAVs approaching an intersection in mixed traffic?
- What influence will the presence of CAVs have in mixed traffic?
- What is the influence of the proposed scheduling algorithm on the delay of vehicles at an intersection?

In Section 2, a literature overview on the scheduling of CAVs at intersections and their trajectory planning is provided. Particularly, it will summarize the papers of Miculescu and Karaman [6, 7] and Timmerman and Boon [3]. Research considering mixed traffic will also be considered, to establish basic knowledge of existing models with both RVs and CAVs. After the literature overview, Section 3 will establish the mathematical model used throughout this paper. Then in Section 4 a scheduling algorithm for mixed traffic is presented. This algorithm is a continuation on the Platoon Forming Algorithm of Timmerman and Boon [3]. By grouping vehicles together
into platoons that will cross the intersection collectively, this algorithm aims to efficiently use the intersection. The model from Timmerman and Boon has been adapted to take the presence of RVs into account. Next, in Section 5 the trajectories of both RVs and CAVs will be described. The concept for these trajectories is taken from the MotionSynthesize algorithm of Miculescu and Karaman [6, 7], which describes the behaviour with a linear optimization problem. While we can define an “optimal” criterion for trajectories of CAVs, we note that trajectories of RVs will only be dependent on the traffic light. To reduce computational complexity, we will try to construct a closed-form solution for our trajectories of RVs and CAVs. Finally, in Section 6, results of simulations with our proposed algorithms will be discussed, and in Section 7, we will draw conclusion from this.

2 Literature overview

At traditional intersections, conflicting flows are directed across the intersection with a switching process. This switching process alternatively gives batches of vehicles on conflicting lanes access to the intersection by changing the colours of the traffic light. Typically, there are three general approaches to the switching process: fixed-time, actuated and adaptive. Where fixed-time is based on timers, regardless of traffic flow, actuated and adaptive react to the real-time traffic situation with the aid of sensors. A commonly used, but often inaccurate roadway sensor is the inductive-loop detector [8]. Therefore, information from sensor data is unreliable, whereas the introduction of CAVs could give access to more and exact data (location, velocity, direction, etc.) about the car. Moreover, this information would be available further away from the intersection, allowing time for adaptation of the switching process, as well as the CAVs trajectories, assuming autonomous vehicles can communicate with each other and adapt their driving [9, 10].

When switching from RVs to CAVs, both the traditional switching process, as well as the vehicles’ trajectories, can be adapted to improve efficiency of the intersection. Hence, there are multiple aspects to consider when aiming to efficiently manage an intersection. Several studies have investigated traffic control strategies using the information from CAVs and adapting their trajectories. Some approaches exhibit good performance, but a mathematically rigorous analysis of performance is not available in these frameworks due to the complexity [11, 12, 13]. The results solely rely on simulation experiments. Furthermore, due to the complexity of the scheduling problem (NP-hard [14]), a trade-off between computational complexity and optimality arises. There are algorithms to direct the switching process that cannot be used in real time because of the long calculation times [15], and other algorithms even result in suboptimal or infeasible solutions [16].

To construct a model that could be used in real-time, and could possibly be mathematically analyzed, we will consider articles that have formulated different scheduling strategies for intersections with CAVs. These scheduling algorithms find their foundation in queueing theory, and we will expand on how principles from queueing theory can be used to model traffic intersections. Later, we will apply these principles to our model for mixed traffic. Furthermore, we will describe how trajectories for CAVs have been derived in these papers. Lastly, we will focus on articles that have modeled mixed traffic, and we will conclude with the contributions of our proposed model to existing literature on mixed traffic intersections.

Using queueing models for traffic intersections Queueing theory has long been important in the modeling and performance analysis of signalized and unsignalized intersections [17, 18]. When viewing each lane as a queue, and the green light as the server, it seems straight-forward to use queueing theory to model an intersection. The challenging part in developing a queueing model for a traffic intersection lies in its specific dynamics: there are multiple queues, served in a cyclic manner by a single server, where effects from acceleration and deceleration influence the dynamics of vehicles. In the existing queueing literature, polling models have been developed to capture a part of these queueing dynamics. Polling models are queueing system where a single
server visits multiple queues, often in a cyclic order. In general, a polling model has \( n \) queues, each having a distinct arrival process with parameter \( \lambda_i \), \( 0 \leq i \leq n \). It is assumed these arrival processes are independent from each other. Each queue has a generally distributed service time, which are assumed to be independent. There is a single server that visits each of the \( n \) queues in a particular order to serve customers. The server spends a certain period at a queue, before switching to the next one. Switching can be followed by a switchover and/or setup time, depending on the model assumptions.

Miculescu and Karaman, as well as Timmerman and Boon, have argued that polling models are particularly suitable to model traffic intersections for CAVs [6, 7, 3]. To describe a traffic intersection with a polling model, each lane at the intersection can be considered a queue, where the green light acts as the “server”. The server will only serve one lane at a time, which means that while one queue has a green light, the other queues have a red light. It is assumed switching between queues takes zero time, but once switched to a non-empty queue, a setup time is required. This setup time corresponds to the required time between two green lights, in order to make the crossing empty and safe. Once in turn to be served, the service time resembles the time between two consecutive vehicles. When the server has finished serving a vehicle, it is immediately ready to serve the next vehicle of that lane. It is important to note that polling models consider a vertical queue, where all customers stop exactly at the stop line, occupying no space at all. Translating this to a traffic intersection, would mean that once a vehicle has finished service, the next vehicle is immediately ready to be served. In other words, the next vehicle would instantly be at the stop line, and there would be no effects from accelerating and decelerating. However, Timmerman and Boon[3] argued that there is a one-to-one relation between the vertical polling model, and their algorithm. By planning the trajectories of CAVs, it can be ensured that all vehicles cross the intersection at maximum velocity. As a consequence, effects from acceleration and deceleration on the service time are avoided.

**Scheduling CAVs** With the model of their intersection based on a polling model, Miculescu and Karaman propose an algorithm to simulate different scheduling policies that originate from polling models [6, 7]. In Figure 1, the general setting of the intersection they consider, is displayed. At this intersection, there are two conflicting lanes where only CAVs will arrive. The presence of a central controller with a certain control radius is assumed, where each vehicle that enters the control region, is registered by the central controller. Once inside the control region, the central controller governs the velocities of all vehicles. When a vehicles arrives in lane \( k \), a corresponding customer in the polling model is added to the end of queue \( k \). Their procedure `Simulate` simulates the polling system with its current customers and returns the scheduled service times of each vehicle.
customer. This schedule is determined by the polling policy, the service time, the setup time, the order and arrivals of the vehicles, and the time the server started serving. Scheduled times may be altered upon a later simulation of the polling system, due to the arrival of a new vehicle. Next to the connection between polling systems and an autonomous intersection, Miculescu and Karaman also provide a coordination algorithm for the trajectories of the CAVs. These trajectories are based on the MotionSynthesize procedure. MotionSynthesize generates an optimal trajectory for a scheduled vehicle. Here, an optimal trajectory is defined by minimizing the average distance to the intersection, while keeping a safe distance from their predecessor and crossing the intersection at maximum speed. To implement the procedure, a linear optimization on the discretization of the distance to the intersection was constructed. Note that crossing the intersection at maximum speed, ensures that the shared resource, the intersection region, is used as efficiently as possible. Moreover, with all cars crossing the intersection at the same speed, this simplifies the service time in the polling model. Each car occupies the intersection equally long, which makes the service time in the underlying polling model a constant.

Continuing on this model, Timmerman and Boon have constructed Platoon Forming Algorithms to schedule arriving vehicles, after which the MotionSynthesize procedure is used to determine the vehicles’ trajectories [3]. The proposed Platoon Forming Algorithms in this paper more explicitly connect to policies from polling models. Also, an explicit rule is provided to determine the construction of platoons: vehicles that can get within a specified distance of their predecessor, are allowed to join their platoon. Furthermore, in this paper, a closed-form solution to the MotionSynthesize procedure for a particular polling policy was constructed. After proving the correctness of the closed-form, the performance of the model was analyzed. Due to the explicit link between their model and polling models, and the simplification of the service times of vehicles, the mean delay could be approximated using formulae from queueing theory. The approximations of the delay of three different polling policies were compared to the delay of traditional traffic lights. All three policies showed a significant improvement in average delay as compared to a fixed-cycle traffic light.

Another article that has done research on this topic, is from Tachet et al. [4]. In this paper, the BATCH procedure is proposed, which behaves similar to the Platoon Forming Algorithm of Timmerman and Boon [3]. Vehicles are scheduled in a First Come First Served (FCFS) manner, until one vehicle $V$ is expected to experience a delay of $\Delta$. All vehicles, up to a maximum, that have arrived at most a time $\Delta$ after $V$ are then considered for rescheduling. The vehicles are grouped together per lane, and each group is scheduled to cross the intersection together. In other words, in each lane a platoon of vehicles is created that will cross the intersection. Tatchet et al. later compare the service rates of a traditional fixed-cycle traffic light and the BATCH algorithm, and show that service rates of the BATCH procedure are superior, even asymptotically doubling service capacity.

To actually implement each of the previously mentioned algorithms, the resulting schedules must be safe such that collisions are prevented, as well as feasible such that vehicles adhere to traffic regulations. With each of the previously mentioned algorithms, cars may be rescheduled multiple times once they have entered the control region. In order to ensure feasible solutions, and important condition these scheduling algorithms have to satisfy is regularity.

**Definition 2.1** (Miculescu and Karaman 2014, 2019). A polling policy is regular if vehicles within each queue are serviced in a First Come First Served manner, and an arrival in a queue does not change the order of service of all currently present vehicles, i.e. the new arrival is inserted somewhere in the order of service of all waiting vehicles.

On top of a regular polling policy, to ensure safe conditions at the intersection, a sufficiently large control region is required. These assumptions are necessary with respect to the rescheduling of vehicles, in order to ensure prevent collisions. Furthermore, it is assumed arriving vehicles can only enter the control region when a safe and feasible trajectory is possible. In total, these
assumptions will ensure that the proposed scheduling algorithm, as well as the trajectory planning do not cause any collisions.

**Mixed traffic models** The previously mentioned studies have only considered intersections where autonomous cars drive exclusively. The proposed algorithms are not suitable to schedule mixed traffic, since they use characteristics of CAVs that RVs do not posess. Although limited in numbers, various papers have been published on mixed traffic. Not all proposed algorithms can be mathematically analyzed, nor are fit for real-time traffic control due to the computational intensity [19]. To broaden our knowledge on models for mixed traffic, two papers will be considered that investigate control strategies at an intersection with mixed traffic. The influence of the proportion of CAVs, the penetration rate, on each of these strategies are also considered.

Guler, Menendex and Meier [20] have proposed a mechanism that could be equipped at intersections with mixed traffic. In their paper, data from CAVs is used to determine a control strategy at an intersection similar to Figure 1 with mixed traffic. Two events are of most interest for their algorithm: the time a car enters the control region, and the distance from the intersection at which the cars comes to a stop. At the intersection, a central controller exists that can obtain these two events for CAVs. For RVs, it is assumed that only the first event can be registered by the central controller using detectors. The number of RVs in a queue, as well as their arrival times, can be estimated using information obtained from CAVs. Two algorithms are proposed for the departure structure of the vehicles in the control region, that use this information from CAVs. At each iteration, the algorithm determines all possible combinations of departures of the N cars in the control region, and selects the combination that performs best on the objective function. Even with a 0% penetration rate of CAVs, their algorithm improves the average delay per car as opposed to a fixed-time traffic signal. Although the value of CAVs is most notable in low-demand scenarios, the most significant value from CAVs is the information obtained from them. With this information, a penetration rate of 60% can decrease the average delay of cars with 60% for low demands. These results are based on simulations, because no mathematical analysis of the underlying model is possible.

Another paper that investigates how mixed traffic at an intersection can be directed, is by Moradi et al. [21]. In this paper, it is investigated how RVs and CAVs in mixed traffic can be grouped together into platoons. An intersection with four approaching directions is considered, where a central controller is assumed to only register CAVs that enter the control region. Information from CAVs is used to obtain an estimate of the total platoon length, consisting of both CAVs and RVs. This estimation allows them to compute a green time for each lane, with a maximum of $T$ time. Connecting this to polling models, the server simultaneously serves compatible lanes for this estimated green time. Once this time is over, the server switches to the next group of compatible lanes following a predetermined pattern. Furthermore, they propose an algorithm for the speed-trajectories of CAVs, based on minimizing the energy consumption of CAVs. Comparing their algorithm to pre-timed signal settings shows a significant improvement in performance of the intersection. Even with relatively low penetration rate of CAVs, their algorithm still outperforms classic traffic light systems.

**Contributions** In this literature overview, multiple articles have been discussed on which our model will be based. Most importantly, our model is an extension to the models from Miculescu and Karaman [6, 7], and Timmerman and Boon [3]. Furthermore, our model differs from the model of Moradi et al. [21] by considering a different and more explicit polling policy. The key contributions to existing literature with our model are:

- The proposed scheduling algorithm to efficiently manage an intersection with mixed traffic, that finds its origin in polling models,
3 Model Description

In this Section, we will propose a model to describe an intersection with mixed traffic, where approaching vehicles are either Connected and Automated Vehicles (CAVs) or Regular Vehicles (RVs). For simplicity, we will consider the vehicles entering only from two directions, as in Figure 2. However, the model can be extended to intersections with \( n \in \mathbb{N} \) crossing lanes. We will first discuss the properties of the intersection that we consider, after which we will elaborate on the vehicles, and how this connects to the polling model.

Similar to Timmerman and Boon, the existence of a central controller at the center of the intersection is assumed [3]. The central controller registers and communicates with all vehicles in its range, which we define as the control region. We assume all vehicles enter the control region at maximum velocity \( v_{\text{max}} \) and a safe distance behind their predecessor. Here, a safe distance is defined as:

\begin{definition}
A safe distance denotes at least the minimum distance necessary between vehicles driving at maximum velocity such that no collisions occur when their predecessor starts to decelerate at maximum rate.
\end{definition}

CAVs are assumed to have a faster response time, which allows them to safely drive at a closer distance behind their predecessor, than compared to RVs.

The central controller has two main tasks: scheduling vehicles that have entered the control region, and suggesting driving trajectories to CAVs. For these two tasks, the control region can be divided into an inner and outer ring, as in Figure 2. Vehicles entering the outer ring are scheduled to cross at assigned times. These scheduled times can still vary while the vehicle is in the outer ring. Once a vehicle enters the inner ring, its scheduled crossing time is fixed, and it is possible for the central controller to provide a driving trajectory. A sufficiently large outer ring ensures scheduled crossing times to be fixed when entering the inner ring, and a sufficiently large inner ring allows for enough space to adapt the trajectory to cross at the scheduled time.

To perform these two general tasks, the central controller is assumed to be able to communicate with all vehicles inside the control region. This communication consists of retrieving and supplying data to and from vehicles. For CAVs this means that the central controller can retrieve data such as speed, direction, location. For RVs, we assume one-way communication with the central controller. The central controller can obtain the speed, direction and location of RVs in the control region, but RVs cannot obtain any information from the central controller. With the vehicle being human-driven, it is assumed the behaviour is only influenced by their predecessor and the traffic light. Once CAVs have entered the inner ring, the scheduled crossing time is fixed and the central controller can supply a driving trajectory that extends beyond the intersection to CAVs. Taking the predecessor and scheduled crossing time into account, a speed profiling algorithm is used to compute a trajectory for the CAV. In our model, CAVs are assumed to drive exactly according to the suggested trajectory. Although RVs are unaware of their scheduled crossing time, the driving behaviour of RVs is assumed to be deterministic, based on the colour of the traffic light. This assumption is necessary such that complete and safe trajectories for CAVs can be computed and executed. In our model, this means that it is known exactly how an RV will drive given their predecessor, and the colour of the light.
To describe the policy the central controller will use to schedule vehicles, we establish the connection between our intersection and a polling model with \( n = 2 \) queues. In Figure 3, we have visualized the polling model that corresponds to our traffic intersection, where each queue resembles a lane, and the server acts as the green light. Each queue has its own arrival distribution, and they are assumed to be independent. To ensure a safe distance between vehicles, we assume the inter-arrival time between vehicles to be drawn (independently) from a shifted Exponential distribution with parameter \( \lambda_i \) at lane \( i \). The distribution is shifted with a constant \( \delta \) that provides at least a safe distance between vehicles upon arrival. Each arriving vehicle is a CAV with probability \( p_A \), or an RV with probability \( 1 - p_A \). The service time of a vehicle is equal to the time between headways of two consecutive vehicles. Due to the complexity of the behaviour of RVs and CAVs, service times are dependent on the algorithm used for the trajectories - which means that service times between vehicles could be dependent. When the server has finished serving one lane according to a specified polling policy, it switches cyclically to the next lane. When switching between lanes, we assume a zero switchover time in our proposed scheduling algorithm. However, once arriving at a non-empty lane, a fixed set-up time \( S \) is required.

In the outer ring, the central controller will schedule vehicles using an algorithm that is based on the exhaustive service discipline of polling models. When using an exhaustive policy in a polling model, the server will keep serving customers until the queue is completely empty. Once the queue is empty, the server switches to the next queue. Translating this to our model of an intersection, this means the following: each vehicle that can get close enough to their predecessor, is allowed to also cross the intersection with their predecessor. Once there are no more vehicles in the lane that can satisfy this condition, the server cyclically switches to the next queue. This requirement allows groups of vehicles that drive closely together, \textit{platoons}, to cross the intersection collectively. Once the schedule has been determined, the central controller gives a fixed scheduled time \( c_V \) to each car \( V \) that enters the inner ring of the control region. Note that RVs will be unaware of their scheduled times, since the central controller cannot supply this information to RVs. Therefore, the scheduled time of the first vehicle of a platoon corresponds to the moment the traffic light turns green. When the head of a platoon is “scheduled” by the policy for time \( c_V \), the light turns green at time \( c_V \). The light will remain green until the scheduled time of last vehicle of the platoon. CAVs can obtain this green-time and take it into account when driving. However, RVs cannot take this estimated green time into account in their driving, and will only adapt their behaviour once the light has turned green.

Although our model to describe an intersection with mixed traffic is based on a polling model,
its dynamics are too complicated to analyze explicitly. Due to the dependence of the service times, classical polling literature cannot be used to determine the mean delay. Therefore, results of the proposed algorithms in our model will be obtained using simulations.

4 Scheduling Algorithm

To describe the scheduling algorithm, note that the planning of vehicles occurs only for cars that are in the outer ring of the control region. Each time a vehicle enters the outer ring, it is registered by the central controller. This specifically means that after registering the car, the central controller knows the location, speed, and lane of the car. With this information, the central controller schedules the car to cross the intersection for a specific time, depending on the scheduling policy. Once the car is scheduled, the scheduled times of other cars in the outer control region might need to be adjusted. The central controller cannot communicate the scheduled times to RVs, who can only act on what is observable for them - i.e. the colour of the traffic light. The scheduling algorithm should take the distinct behaviour of RVs into account to safely schedule all approaching vehicles. For our model, we will create a Platoon Forming Algorithm (PFA) based on the research of Timmerman and Boon [3], that groups vehicles into platoons that cross the intersection together. These platoons will be created by the green intervals of the traffic lights, as well by planning the trajectories of CAVs. In this section, we will focus on the PFA and argue its solution to be safe and feasible, such that no collisions occur, and vehicles can adhere to the traffic rules. In Section 5 the method of trajectory planning for CAVs and RVs will be discussed.

Before elaborating on the PFA, we first introduce some concepts and notations. Each vehicle $V$ is defined by various properties. Firstly, the boolean $A_V$ stores for each vehicle whether it is a CAV (True) or an RV (False). Furthermore, it is assumed that all vehicles drive at a safe distance from their predecessors, using Definition 3.1 for the safe distance. The minimum safe distance for CAVs and RVs is denoted with $B_s$ and $B_d$ respectively, where $B_s \leq B_d$. Also, each vehicle $V$ has a lane $d_V$, an arrival time $rt_V$, a minimum crossing time $a_V$, a scheduled crossing time $c_V$, and the time the light turns green for their platoon $gr_V$. All vehicles that enter the control region are assumed to drive at maximum speed, $v_{max}$.

Exhaustive polling policy

Algorithm 1 has been constructed to schedule approaching vehicles at the intersection. This algorithm continues on the research from Timmerman and Boon [3], and is an implementation of the exhaustive polling policy at an intersection. Their proposed algorithm for the exhaustive polling policy has to be adapted to implement RVs, yet the general idea still holds. To allow for RVs in the approaching traffic, we propose the following rule to govern the formation of platoons: if a vehicle would be able to cross the intersection at a distance $B_d$ or $B_s$ (depending on whether the vehicle is an RV or CAV respectively) behind their predecessor, this car is allowed to join the platoon. If there are no more vehicles in the lane that can join the platoon, the server switches to the next queue. We assume that switching between queues takes zero time. Instead, when the server starts serving a nonempty queue, an additional setup time $S$ is necessary before the first customer can actually be served. Once the setup time $S$ has been completed, the platoon of vehicles in the next non-empty lane may cross the intersection. This results in a cyclic departure of platoons. However, note that in Timmerman and Boon [3], each CAV is assumed to cross the intersection at maximum speed. Hence the service time for each vehicle is a constant. Taking the insusceptible behaviour of RVs into account, the vehicles in our model do not necessarily cross the intersection at maximum speed. Moreover, vehicles might not even cross the intersection exactly at their scheduled time. Therefore, the service times of vehicles in our model are dependent on their trajectory when approaching the intersection. To enable the central controller to still schedule the vehicles according to the exhaustive policy, we require the existence of two auxiliary functions: Motion and SafeDist.
Auxiliary functions for scheduling  

First, we assume the existence of an algorithm \texttt{Motion} that determines the trajectories of CAVs and RVs. We assume the trajectories of RVs to be deterministic, but they can differ from CAVs. The behaviour of vehicles is therefore dependent on their predecessor, as well as their scheduled time and the moment the light turns green in their lane. To compute the trajectory of a vehicle \( V \), \texttt{Motion} uses \( r_{TV} \), \( a_{V} \), \( c_{V} \) and \( g_{TV} \) (which are all properties of \( V \)), and the trajectory of the predecessor (if any) of \( V \). It is assumed that \texttt{Motion} also computes the trajectory for \( V \) a distance \( B_{d} \) beyond the intersection. We will let \texttt{Motion} unspecified for this section, and later in Section 5, we will formulate an exact method and formula for \texttt{Motion}.

Secondly, we assume the existence of a function \texttt{SafeDist}. In order to be able to schedule the next vehicle, the central controller needs to know when their predecessor has crossed the stop line by a distance \( B_{d} \) or \( B_{s} \). We define \( tD_{V} \) and \( tS_{V} \) as the times the vehicle \( V \) has crossed the stop line a distance \( B_{d} \) and \( B_{s} \) respectively. The function \texttt{SafeDist}(\( V \), \( V_{last} \)), will compute the moment a vehicle \( V \) could cross the intersection a safe distance behind their predecessor \( V_{last} \). More precisely, \texttt{SafeDist}(\( V \), \( V_{last} \)) will compute and return \( tD_{V_{last}} \) or \( tS_{V_{last}} \), depending on whether \( V \) is an RV of CAV respectively.

Using \texttt{Motion} and \texttt{SafeDist}, we propose Algorithm 1 to schedule arriving vehicles at the intersection. We will briefly argue why Algorithm 1, \texttt{Motion} and \texttt{SafeDist} are enough for the central controller to schedule all vehicles. Every vehicle that arrives needs to be registered by the central controller. Once registered, it should receive a scheduled crossing time based on a specific scheduling policy. To make sure that vehicles are safely planned behind their predecessor, we can only schedule new vehicles when we know the time that their predecessor has crossed the stop line by a safe distance, which is the objective of \texttt{SafeDist}. However, with vehicles crossing the intersection at different speeds (depending on their arrival time, scheduled crossing time, and green time), this safe time can only be computed when the trajectory of a vehicle is known. Hence we need \texttt{Motion} to compute the trajectory, after which we can use \texttt{SafeDist} to compute the safe times. Then using these two assumed existing algorithms, we are able to schedule the next registered vehicle in the control region safely with Algorithm 1.

Scheduling Algorithm  

In Algorithm 1, the exhaustive discipline for the scheduling policy is described. In the next paragraph the algorithm will be discussed step by step. We will consider the structure of the algorithm, where the variable \( O \) contains the ordering in which all vehicles are scheduled to cross the intersection. When discussing the algorithm, we assume that the variable \( O \) at the start of the algorithm contains a safe and feasible solution to the scheduling and trajectory planning of the previously arrived vehicles in the control region. However, while we discuss the algorithm, we will also argue that the boundary case of the first car entering the control region will give a safe and feasible solution for \( O \). Next we will also argue that every new vehicle entering the control region will also result in a safe and feasible solution, to conclude that the algorithm indeed returns a safe and feasible solution throughout all situations.

Once a vehicle \( V_{0} \) arrives in the control region, the central controller can use the obtained information, as well as the list \texttt{lastCars} to derive \( V_{0} \)’s predecessor (\( \text{pre}_{V_{0}} \)). Then the central controller checks multiple factors to determine where \( V \) should be placed in the ordering \( O \). We will argue that with our model assumptions, there exists at least one safe and feasible solution that Algorithm 1 could return. Note that as discussed in Section 3, \( V_{0} \) is assumed to arrive at a safe distance from its predecessor. With a sufficiently large control region, at least one solution would therefore be for \( V_{0} \) to come to a complete stop and cross the intersection last. This solution would be feasible because per our model assumptions there is enough distance and time for \( V_{0} \) to come to a complete stop before the intersection, without colliding with its predecessor. Furthermore, there are no vehicles that have arrived later than \( V_{0} \), thus no vehicles will collide with \( V_{0} \) if \( V_{0} \) is scheduled last. For the rest of the schedule, \( O \) is assumed to contain a safe and feasible solution.
Hence, there is at least one solution that Algorithm 1 could return, that is both safe and feasible.

**Lines 2 - 12** Once $V_0$ is registered in the control region, we make the distinction when the minimum time of $V_0$ ($a_{V_0}$) is larger than $\text{SafeDist}(V_0, V_{last})$ (line 2) - i.e. $V_0$ can only reach the intersection after the very last car in the ordering has already left the intersection. If we are in this case, $V_0$ must be planned last in $O$. Therefore, we only need to check whether we need to delay $V_0$ to take into account the set-up time $S$. This set-up time is only necessary if $V_0$ drives on a different lane than $V_{last}$. In that case, $V_0$ is scheduled to cross the intersection $S$ time after the last crossing vehicle is at a distance $B_s$ or $B_d$ from the intersection (depending on $A_{V_0}$). If $V_0$ comes from the same lane as $V_{last}$, $V_0$ is scheduled at their earliest possible crossing time (which is still later than $\text{SafeDist}(V_0, V_{last})$). This scheduled crossing time is saved in $c_{V_0}$, and can be used for the trajectory planning of the CAVs. $V_0$ is also assigned a scheduled green-time $gr_{V_0}$, which is the moment the light turns green for the platoon of $V_0$. The green light is what RVs will be able to see when driving towards the intersection. If $V_{last}$ is from the same lane as $V_0$, then the light will simply remain green until $V_0$ has crossed as well. In our model, this means that $gr_{V_0}$ is equal to $gr_{V_{last}}$ - which will inductively be equal to the green-time of the first vehicle of that platoon. If $V_{last}$ departs from a different lane, the light will turn green at $c_{V_0}$, hence $gr_{V_0}$ equals $c_{V_0}$. Now that $V_0$ has been assigned a scheduled crossing and green time, all that remains is appending $V_0$ to the ordering of the vehicles, and computing the trajectory of $V_0$ with Motion (lines 11-12). This solution is safe and feasible, because $V_0$ crosses at a safe distance behind $V_{last}$, and we do not let $V_0$ cross earlier than $a_{V_0}$. The rest of $O$ remains unchanged, which by assumption was a safe and feasible solution, and we can conclude that the new ordering of vehicles is safe and feasible. Also observe that in the boundary case where the first vehicle approaches an empty intersection, the returned ordering of vehicles is certainly safe and feasible by taking $V_{last}$ as the last car that has crossed the intersection.

For this case, everything has now been determined to allow for the safe crossing of $V_0$ with the light turning green at $gr_{V_0}$. In Figures 4 and 5 we have visualized scheduling a CAV and RV at the end of $O$. In these figures, the purple, green and black lines denote vehicles driving towards the intersection. The line $y = 0$ represents the intersection, where on the $y$-axis we have the distance to the intersection $|x(t)|$. Vehicles approaching from one half in the figure drive on the same lane, and when crossing the line $y = 0$, the vehicles have crossed the stop line. The arrival time of the CAV and RV are equal, yet their scheduled crossing times differ, due to the difference in $B_s$ and $B_d$. The CAV is scheduled to cross at time $tS_{V_{last}}$, when $V_{last}$ is at a distance $B_s$ from the intersection, whereas the RV is scheduled when $V_{last}$ has crossed the intersection a distance $B_d$, at time $tD_{V_{last}}$. Had $V_{last}$ approached from the opposing lane, then Figures 4 and 5 would have looked similar, except with an additional set-up time $S$ between $V_{last}$ and the CAV or RV.

![Figure 4: Scheduling a CAV to cross the intersection last at $tS_V$](image1)

![Figure 5: Scheduling an RV to cross the intersection last at $tD_V$](image2)
Lines 13-26 In the case that the minimum time of $V_0$ is within the scheduled times of the current ordering of vehicles, we need to determine where to place $V_0$ in the ordering. For the solution to be feasible, $V_0$ can only be scheduled later than its minimum time, $a_{V_0}$. First, we check whether $V_0$ can join the previous platoon departing from their lane (lines 15 - 19). This is the case if $V_0$ could arrive at the intersection before the last vehicle of the previous platoon is at a safe distance $B_3$ of $B_4$ from the intersection. When joining a platoon, the light will simply remain green for longer, hence $gr_{V_0}$ is updated to the green time of their predecessor, which will be equal to the green time of the first car of the platoon. $V_0$ is inserted in the ordering behind $pre_{V_0}$. However, if $V_0$ is unable to join the previous platoon leaving from $d_{V_0}$, it will start a new platoon in its lane. For a cyclic departure of platoons, the last previous departing platoon is determined by going against the cyclic direction and checking the safe time from each last departing vehicle (lines 23-28). The new platoon is scheduled after the last departing platoon that comes before lane $d_{V_0}$. This cyclical structure corresponds to the single cyclically roving server in polling models. After inserting $V_0$ behind this last platoon in $O$, there are no later platoons than $V_0$ that will be served before the server visits lane $d_{V_0}$ in $O$. However, it is important to mention that the scheduled times for vehicles in lanes that are served after $d_{V_0}$ in $O$ need to be recomputed. Once planned, $V_0$ is assigned a scheduled crossing time, and a scheduled green time.

Now to ensure that Algorithm 1 returns a safe and feasible solution for $O$, all vehicles that are scheduled to cross later than $V_0$ need to be rescheduled. One important condition for our PFA to be able to reschedule these vehicles is regularity, using Definition 2.1 from Miculescu and Karaman [6, 7].

Since in Algorithm 1 a new vehicle $V$ is placed within $O$ and the rest of the ordering of the vehicles remains the same, we conclude that our PFA is regular. This condition, together with a large enough control region, ensures that we can find safe and feasible trajectories for the rescheduling of the vehicles. Although similar to Timmerman and Boon [3], the rescheduling is where Algorithm 1 differs most from their exhaustive algorithm to schedule CAVs (p. 284, Algorithm 1). With the assumptions that all vehicles cross the intersection at maximum speed, vehicles that are scheduled to cross after $V_0$ would simply need to be delayed by a fixed time. However, in our model, we have dropped this assumption and there is no guarantee that RVs nor CAVs will cross the intersection at specified speeds. Therefore, vehicles will be delayed by a time that is dependent on the behaviour of their predecessor, and the moment the light turns green. When scheduling $V_0$ within $O$, all later scheduled vehicles need to be rescheduled, and their trajectory needs to be recomputed (lines 32 - 42). With the regularity of our PFA and the large enough control region, it is ensured that only vehicles who are still able to adapt their trajectory will be rescheduled. Vehicles that are too close to the intersection to adapt their trajectory will not be rescheduled because the order of all waiting vehicles change and $V_0$ could never reach the intersection before they cross. This implicitly creates the inner and outer ring of the intersection, where the scheduled times of vehicles in the inner ring has become fixed. These assumptions ensure that rescheduling the vehicles is possible and can be done safely and feasibly.

Lines 27-37 To reschedule all vehicles that are scheduled to cross after $V_0$, we do the following: for each vehicle car, we check whether the previous vehicle precar in $O$ departs from the same lane. If this is the case, then either car was part of the same platoon as precar, or the minimum time of car was too late to join the platoon and car had started a new platoon. We reschedule car to cross at $\max(c_{car}, \text{SafeDist}(car, precar))$, with car either joining the platoon of precar or having their own platoon. By taking this maximum, the new scheduled time for car remains feasible and safe. If car does not depart from the same lane as precar, then it is the first car of a platoon. Then car is also rescheduled to make sure it its new scheduled time is feasible and at a safe time from precar taking the set-up time $S$ into account.
After rescheduling car, we use Motion(car, precar) to compute the new trajectory of car. To argue that this once again results in a safe and feasible solution, we start with acknowledging that \( V_0 \) has been scheduled feasibly and safely in lines 15 - 30 of Algorithm 1. Each next vehicle in the ordering is scheduled at least at a safe distance behind the previous vehicle that crosses the intersection - taking \( S \) into account if this vehicle is on a different lane. Note that the regularity of our PFA and the large enough control region ensure that we can do this. Therefore, this solution for the next vehicle is also safe and feasible. Continuing this iteratively, all solutions for the vehicles in the ordering after \( V_0 \) consist of a new safe and feasible scheduled time. Combining this with the assumption that the ordering of \( O \) for vehicles before \( V_0 \) is safe and feasible, we can conclude that Algorithm will always return a safe and feasible solution for \( O \).
Algorithm 1: Exhaustive PFA\((O, \text{lastCars}, V_0, S)\)

**Input:** current ordering \(O\) of vehicles \((V_1, \ldots, V_K)\) ordered on basis of \(c_V\); \(V_{\text{last}}\) defined as \(V_K\) or the last car to cross the intersection; list \(\text{lastCars}\) with the last car departing from each lane; \(a\) to be scheduled vehicle \(V_0\), with minimum time \(a_{V_0}\) in lane \(d_{V_0}\); set-up time \(S\)

**Output:** An ordering of all cars in the control region, a list of the last cars of all lanes

1. \(\text{prev}_{V_0} \leftarrow \text{lastCars}[d_{V_0}]\)
2. If \(a_{V_0} > \text{SafeDist}(V_0, V_{\text{last}})\) then
   3. \(\text{if } d_{V_0} = d_{V_{\text{last}}} \text{ then} \)
      4. \(t \leftarrow a_{V_0}\)
      5. \(gr \leftarrow gr_{V_{\text{last}}}\)
   6. Else
      7. \(t \leftarrow \max(a_{V_0}, \text{SafeDist}(V_0, V_{\text{last}}) + S);\)
      8. \(gr \leftarrow t;\)
      9. \(c_{V_0} \leftarrow t;\)
      10. \(gr_{V_0} \leftarrow gr;\)
      11. Append \(V_0\) to the ordering of vehicles \(O\);
      12. \(\text{Motion}(V_0, \text{prev}_{V_0})\)
3. Else
   4. If \(\text{prev}_{V_0}\) exists and \(a_{V_0} \leq \text{SafeDist}(V_0, \text{prev}_{V_0})\) then
      5. \(c_{V_0} \leftarrow \text{SafeDist}(V_0, \text{prev}_{V_0})\)
      6. \(gr_{V_0} \leftarrow gr_{\text{prev}_{V_0}}\)
      7. Append \(V_0\) in the ordering of vehicles behind \(\text{prev}_{V_0}\)
5. Else
   6. For \(l\) in \((d_{V_0} - 1, d_{V_0} - 2, \ldots, 1, n, n - 1, \ldots, d_{V_0} + 1)\)
      7. Do
         8. \(\text{car} \leftarrow \text{lastCars}[l]\)
         9. \(\text{if } a_{V_0} < \text{SafeDist}(V_0, \text{car}) + S \text{ then} \)
            10. \(c_{V_0} \leftarrow \text{SafeDist}(V_0, \text{car}) + S\)
            11. \(gr_{V_0} \leftarrow \text{SafeDist}(V_0, \text{car}) + S\)
            12. Append \(V_0\) in the ordering behind \(\text{car}\)
     13. \(\text{Motion}(V_0, \text{prev}_{V_0})\)
     14. \(k = \text{index of } V_0 \text{ in } O\)
     15. For \(k < j \leq \text{len}(O)\) do
         16. \(\text{if } d_{\text{car}} == d_{\text{precar}} \text{ then} \)
             17. \(c_{\text{car}} \leftarrow \max(c_{\text{car}}, \text{SafeDist}(\text{car}, \text{precar}))\)
             18. \(gr_{\text{car}} \leftarrow gr_{\text{precar}}\)
         19. Else
             20. \(c_{\text{car}} \leftarrow \max(c_{\text{car}}, \text{SafeDist}(\text{car}, \text{precar}) + S)\)
             21. \(gr_{\text{car}} \leftarrow \text{SafeDist}(\text{car}, \text{precar}) + S\)
             22. \(\text{Motion}(\text{car}, \text{precar})\)
     23. \(\text{return } (O, \text{lastCars})\)
5 Trajectories

Once an approaching vehicle has been scheduled, the central controller can supply the CAV with a driving trajectory. With CAVs being able to anticipate the moment a traffic light turns green or red, their behaviour can be adapted for a specific goal. RVs on the other hand, are not affected by the scheduled times of the central controller. Their behaviour only depends on the current colour of the traffic light, and the behaviour of their predecessor. To distinguish the behaviour of CAVs from RVs, we have come up with two different linear optimization programs to describe their trajectories. For this linear optimization program, the trajectory of the car is discretized. Hence we use \( N \) discrete time-steps of \( \Delta t \), and solve a linear optimization problem with the variables distance, velocity and acceleration. Let vectors \( \mathbf{x}, \mathbf{v} \) and \( \mathbf{u} \) denote the position, velocity and acceleration, with \( x[j], v[j] \) and \( u[j] \), the \( j \)th element of this vector. Furthermore, we use the following notations for a car \( V \):

- \( v_{\text{max}} \): maximum speed of \( V \),
- \( a_{\text{max}} \): maximum acceleration and deceleration of \( V \),
- \( -x_{V,0} \): starting distance from intersection,
- \( rt_V \): arrival time of car \( V \) in control region,
- \( c_V \): scheduled time to cross,
- \( gr_V \): moment the light turns green for the platoon of \( V \).

This notation will be used in the linear programs for both CAVs and RVs. With this notation, \( x[j], v[j] \) and \( u[j] \) denote the location, speed and acceleration at time \( rt_V + j \cdot \Delta t \). The idea is similar to the LP MotionSynthesize of Miculescu and Karaman [7], except some conditions need to be altered to take into account the different behaviour of RVs.

For CAVs, MotionSelf has been constructed to compute the trajectory of a self-driving car, and for RVs we use MotionBest. In Figure 6, the difference in driving behaviour of an RV and CAV is displayed. Both cars have the exact same properties, except for the boolean \( A_V \) that describes whether the vehicle is self-driving (True) or regular (False). The cars arrived at \( t = 0 \) at distance 50 from the intersection. Both cars are scheduled to cross the intersection at \( t = 5 \), with the dotted green line depicting the moment the traffic light turns green. Only the CAV is aware of this scheduled green time, whereas the RV cannot anticipate on it. Note that this enables the CAV to cross at exactly \( t = 5 \), whereas the RV crosses later. In the rest of this section, the details of both behaviours will be described. First, the behaviour of RVs and CAVs will be described with a linear optimization problem, and later a closed-form solution will be constructed.

![Figure 6: Different trajectories of an RV and CAV](image-url)
5.1 Regular Vehicles

To describe the behaviour of RVs, MotionBest is divided in two parts to denote the behaviour before (MotionBest One) and after (MotionBest Two) the light has turned green. This distinction is necessary, since the behaviour of RVs is significantly different after the light has turned green, because they have no prior knowledge on the green times of the traffic light as they are approaching. Uncertain of the moment the light will turn green, an RV will behave as if the light will remain red indefinitely. Moreover, if a complete stop is necessary, an RV will do this as close as possible to the intersection. In Figures 7, 8 and 9, the trajectory, speed and acceleration of an RV using MotionBest One and MotionBest Two is depicted. The red line corresponds to the trajectory computed by MotionBest One; the blue line by MotionBest Two. The final trajectory is created by connecting the trajectories of MotionBest One and Two. To explain how this trajectory is computed, the linear optimization problem used for MotionBest One and Two will be described.

MotionBest One simulates the behaviour, as if the light would remain red indefinitely. In that case, the car will start decelerating only once this is absolutely necessary, to come to a complete stop at the closest possible location to the intersection. Although in Figure 7 the RV could have crossed at $t = 5$, it had to break in the case the light would not turn green, in order to come to a complete stop at the intersection. The RV will remain at this position indefinitely, because RVs do not have prior knowledge about the green time of the traffic light. Coming to a complete stop
as close as possible to the intersection, corresponds to minimizing the distance to the intersection of the car at any point in time. Since \( x_{v,0} \) is a negative value, the problem becomes a linear maximization problem. Let \( y \) be the trajectory of the RVs predecessor, then Algorithm 2 describes the corresponding linear program for MotionBest One.

Algorithm 2: MotionBest One

| Input: | \( x_{v,0}, v_{\max}, a_{\max}, N_0, r_{Tv}, c_{V}, y \) |
| N = \( N_0 + [(10/\Delta t)] \) |
| maximize: | \( \sum_{j=0}^{N} x[j] \) |
| subject to: | \( x[j + 1] = x[j] + (v[j] + v[j + 1]) \cdot \Delta t/2 \) for all \( j \) |
| | \( v[j + 1] = v[j] + u[j] \cdot \Delta t \) for all \( j \) |
| | \( x[j] \leq y(c_v + j \cdot \Delta t) - L \) for all \( j \) |
| | \( 0 \leq v[j] \leq v_{\max} \) for all \( j \) |
| | \( -a_{\max} \leq u[j] \leq a_{\max} \) for all \( j \) |
| return | \( x, v, u \) |

The first two constraints are the discretized versions of the regular relations between the RV’s location, speed and acceleration. Using the trajectory of the RVs predecessor \((y(t))\), MotionBest One will ensure safe distance between the cars. Note that all constraints are similar to the MotionSynthesize procedure of Miculescu and Karaman [7], except for the last line. The condition \( v[N] = v_{\max} \) has been changed to \( v[N] = 0 \) for Algorithm 2. This ensures the inevitable stop of the RV, since the car assumes the light to remain red indefinitely. Note that the final time that the RV is required to be at a complete stop \((r_{Tv} + N \cdot \Delta t)\) is greater than \( c_{V} \). The RV will come to a complete stop at the intersection as soon as possible, and will remain there until the light turns green - which at its latest is \( c_{V} \).

At some point after the arrival of the car, the light will turn green. This event is certain, due to the scheduling of the cars. Exactly from that moment on, the car will accelerate in order to reach the maximum speed as fast as possible. This is where MotionBest Two comes into play, which corresponds to the blue line in Figure 7, 8, and 9. MotionBest Two connects exactly to MotionBest One, matching the location and speed of the vehicle, but now the objective of the linear program is to maximize the speed. Since the vehicle’s trajectory is discretized, MotionBest Two starts at the first time-step after the light has turned green. \( i = [(g_{Tv} - c_{V})/\Delta t] \) corresponds to the first time-step after the light has turned green and when MotionBest Two begins. MotionBest Two can only be performed when the location and speed of the RV at \( t_i = c_{V} + i \cdot \Delta t \) are known, which we will denote with \( x_{gr}, v_{gr} \). If prior to MotionBest Two, MotionBest One was performed, we have \( x_{gr} = x[i] \), and \( v_{gr} = v[i] \). Maximizing the speed and keeping in mind the constraints, gives the linear optimization program described in Algorithm 3.

Note that MotionBest Two is very similar to MotionBest One, however the objective of the function has changed to maximize the velocity. Moreover two constraints from the last line, \( x[N] = 0 \) and \( v[N] = 0 \), could be lifted, because the RV continues driving once it has crossed the intersection. Furthermore, the trajectory should connect to the trajectory of MotionBest One on location and speed, which is ensured by the constraints \( x[0] = x_{gr} \) and \( v[0] = v_{gr} \).

As opposed to MotionSynthesize [6, 7], MotionBest Two does not require the RV to cross the intersection at the scheduled time with maximum speed. The scheduled time is the first moment the car would have been able to cross the intersection, had it known when the light would turn
Algorithm 3: MotionBest Two

Input: $x_{gr}, v_{gr}, v_{\text{max}}, a_{\text{max}}, N0, gr_V, t_{\text{sched}}, y$

1 $\Delta t = (t_{\text{sched}} - c_V)/N0$

2 $N = N0 + \lceil(10/\Delta t)\rceil$

3 maximize: $\sum_{j=0}^{N} v[j]$

4 subject to: $x[j + 1] = x[j] + (v[j] + v[j + 1]) \cdot \Delta t/2 \quad \forall 1 \leq j < N$

5 $v[j + 1] = v[j] + u[j] \cdot \Delta t \quad \forall 1 \leq j < N$

6 $x[j] \leq y(c_V + j \cdot \Delta t) - L \quad \forall 1 \leq j \leq N$

7 $0 \leq v[j] \leq v_{\text{max}} \quad \forall 1 \leq j \leq N$

8 $-a_{\text{max}} \leq u[j] \leq a_{\text{max}} \quad \forall 1 \leq j \leq N$

9 $x[0] = x_{gr}, \ v[0] = v_{gr}$

10 return $x, v, u$

green. Since an RV cannot anticipate the scheduled green time of the traffic light, RVs will need to decelerate when they are close to the intersection and the light is still red. In this case, the RV will slow down at a distance from the intersection such that they could come to a complete stop at the intersection, assuming the light will remain red. Once the light has turned green, the car will accelerate to maximum speed, but this does not ensure the car will cross at the scheduled time nor maximum speed. Therefore, for RVs, the scheduled time can be interpreted as a lower bound for the moment the car could cross the intersection, but is not definitive. For the scheduling algorithm, this means that $c_V$ cannot be used to schedule the next car safely, because it is unknown when $V$ will be at a safe distance from the intersection. Therefore, to accurately schedule RVs and CAVs in our model, we must compute their trajectories to determine $s_{t_V}$. To illustrate this, we have computed and plotted the different trajectories for different green times $gr_V$ for a vehicle $V$ with arrival time $r_{t_V} = 0$ at a distance of 50 from the intersection, in Figure 10. Additionally, for different green times $gr_V$, the velocity of $V$ after crossing the stop line a distance $B_d$ and $B_s$ - at times $tD_V$ and $tS_V$ respectively - has been computed. This has been visualized in Figure 11. These two figures and show that the trajectory and consequently the speed of $V$ at time $tD_V$ and $tS_V$ is dependent on $gr_V$. If $gr_V$ is earlier, $V$ does not have to come to a standstill, and can cross the intersection with a higher velocity than when $gr_V$ for example is at $t = 8$.

Figure 10: Different trajectories of an RV for different green times $gr_V$. The RV has arrived at $t = 0$ at a distance 50.
5.2 Automated Vehicles

For our model, two-way communication between the central controller and CAVs has been assumed, such that the central controller can suggest a driving trajectory to CAVs. In real life, the CAV possibly needs to adjust this trajectory due to unforeseen circumstances, but for our model CAVs are assumed to be able to drive exactly according to this trajectory.

Algorithm 4: MotionSelf

Input: $x_0, v_{max}, a_{max}, N_0, c_V, t_{sched}, y$

1. $\Delta t = (t_{sched} - c_V)/N_0$
2. $N = N_0 + \lceil (10/\Delta t) \rceil$
3. if predecessor in platoon then
4. $OPT = \sum_{j=0}^{N} x[j]$
5. else
6. $OPT = \frac{1}{x_{v_{max}, N_0}} \sum_{j=0}^{N} x[j] + \sum_{k=N_0}^{V} v[k]$
7. maximize: $OPT$
8. subject to: $x[j + 1] = x[j] + (v[j] + v[j + 1]) \cdot \Delta t/2 \quad \forall 1 \leq j \leq N$
9. $v[j + 1] = v[j] + u[j] \cdot \Delta t \quad \forall 1 \leq j \leq N$
10. $x[j] \leq y(c_V + j \cdot \Delta t) - L \quad \forall 1 \leq j \leq N$
11. $0 \leq v[j] \leq v_{max} \quad \forall 1 \leq j \leq N$
12. $-a_{max} \leq u[j] \leq a_{max} \quad \forall 1 \leq j \leq N$
13. $x[0] = x_0; \ x[N_0] = 0; \ v[0] = v_{max}$
14. return $x, v, u$

The linear optimization program MotionSelf, Algorithm 4, has been constructed for the trajectories of CAVs. MotionSelf is similar to MotionSynthesize [7], but required small changes to take the RVs in traffic into account. To increase throughput at the intersection for traffic with only CAVs, the speed of the CAVs at the intersection is maximized in MotionSynthesize. However in our model, this cannot be used as an equality constraint $v[N] = v_{max}$, because MotionBest does not ensure RVs to cross exactly at the scheduled time ($c_V$) or at maximum speed ($v_{max}$). The scheduled time of a CAV relies on the distance of its predecessor to the intersection. Its predecessor might be at a safe distance from the intersection, but this does not mean the predecessor drives at maximum speed. In the case the predecessor is an RV accelerating from e.g. standstill, the CAV cannot cross the intersection at maximum speed at its scheduled time while

Figure 11: Decreasing velocity of an RV at $tD_V$ and $tS_V$ when the light has turned green at different $t$.

5.2 Automated Vehicles
also keeping a safe distance from its predecessor before and after the intersection. This is solved by both distinguishing whether the CAV has a predecessor in the same platoon (lines 3 - 6), as well as extending the time-frame for the linear program (line 2 in Algorithm 4).

By distinguishing when a CAV is the first vehicle of a platoon, the linear optimization program can ensure a CAV to cross the intersection at maximum speed at the scheduled time. Crossing at maximum velocity would minimize the occupation of the intersection. The objective of the linear optimization program becomes to maximize the speed at the intersection \( v[N0] \). In order to prevent (possible) collisions, this has been chosen as part of the objective of the program, instead of an equality constraint as in MotionSynthesize. Specifically, the sum of \( v[k] \) for \( N0 \leq k \leq N \) is maximized, to extend the trajectory beyond the intersection. This is not enough for the linear optimization program, since we also want CAVs to break or stop as close to the intersection, to make the queue does not develop far away from the intersection. Hence the term \( \frac{1}{x_{v,0} \cdot N0} \sum_{j=0}^{N0} x[j] \) is added to the objective function, to ensure CAVs minimize the distance to the intersection. Note that \( x[j] \geq x_{v,0} \), hence \( -x_{v,0} \cdot N0 \geq \sum_{j=0}^{N0} -x[j] \). The factor \( \frac{1}{x_{v,0} \cdot N0} \) is chosen such that the absolute value of the first term of \( OPT \) in line 6 is smaller than or equal to 1. This way, the linear optimization program will minimize the distance of the CAV to the intersection, but the objective to maximize the speed at and beyond the intersection has greater priority.

To ensure the intersection is used efficiently, CAVs with a predecessor in their platoon will drive as close as possible to its predecessor, instead of maximizing the velocity at and beyond the intersection. We will illustrate the reasoning with an example: Suppose the CAV is the second car in a platoon, with an RV as predecessor. This RV will break at the closest possible point of the intersection, and will possibly not cross the intersection at maximum speed. The CAV is scheduled exactly at the time the RV is at a safe distance from the intersection, but at this point the RV could still drive slower than the maximum speed. If the RV would wait until it would be able to safely cross at maximum speed, the light would have to be green longer. Since the moment the next lane is able to get a green light is dependent on the last moment the light of the previous lane was green, this would mean the next lane would get green later than if the CAV had simply crossed the intersection at a lower speed, directly behind its predecessor. Therefore, CAVs drive as close as possible to their predecessor, which gives rise to the maximization of \( \sum_{j=0}^{N} x[j] \), and cross exactly at their scheduled time \( c_V \), which results in the equality constraint \( x[N0] = 0 \).

5.3 closed-form solution for exhaustive policy

For our model, we have changed the MotionSynthesize procedure [6, 7] to distinguish between RVs and CAVs. Consequently, the closed-form solutions for MotionSynthesize constructed by Timmerman and Boon [3] are not accurate anymore. However, two observations that form the basis for their closed-form solution also hold for MotionBest and MotionSelf ([3], p288):

1. the optimization problems formulated in MotionBest and MotionSelf always lead to piece-wise constant acceleration;

2. if all vehicles decelerate (and possibly stop) at most once, at most four changes in the acceleration occur.

Note that Algorithm 1 considers the exhaustive policy, and from polling literature it is known that the exhaustive service ensures that customers will always be served before the end of the cycle in which they have arrived. In other words, vehicles that cannot join the platoon of their predecessor, will certainly be served the next time the server visits their lane. Hence, all arriving vehicles only decelerate at most once. If we can find the moments the acceleration changes, the trajectory can be determined analytically and a closed-form solution can be constructed. This closed-form solution can be used with the scheduling of Algorithm 1, as well as any other service discipline that ensures vehicles decelerate at most once.
In the remaining part of this section, the closed-form solution of MotionBest and MotionSelf will be constructed, and it will be argued that this is indeed the solution to the linear optimization problems. With the distinction between RVs and CAVs, the behaviour of vehicles in a platoon is also dependent on the vehicles planned earlier in the platoon. Therefore, the closed-form solutions need to be determined for multiple cases, depending on the type of vehicles preceding in the platoon. In order to simplify notation, the following abbreviations will be used:

- $v_{\text{max}}$: maximum speed of a vehicle,
- $a_{\text{max}}$: maximum acceleration of a vehicle,
- $B_s$: distance cars keep from their predecessor,
- $B_d$: distance driven cars keep from their predecessor,
- $r_{TV}$: arrival time of car in control region,
- $g_{rV}$: green time of a vehicle
- $c_V$: scheduled time of a vehicle
- $t_{\text{full}}$: time a vehicle reaches full speed,
- $x_V(t)$: distance of vehicle $V$ to the intersection at time $t$,
- $x_{V,0}$: starting distance to intersection

5.3.1 MotionBest

We first start with the closed-form solution for the first RV in a platoon. Once this solution has been obtained, it can be used to construct the closed-form solution for platoons of RVs (with no CAVs in between).

First RV

By construction, MotionBest consists of two parts: MotionBest One and MotionBest Two. Using this division, the behaviour can also be described by two different principles. In the MotionBest One, the distance to the intersection is minimized, such that i) the vehicle remains at a safe distance from its predecessor, ii) the vehicle comes to a standstill exactly at the intersection. The MotionBest Two considers the moment the light has turned green, maximizing the speed while keeping a safe distance from its predecessor. Since we are first considering the first vehicle in a platoon, predecessors do not need to be taken into account.

By our first observation, MotionBest always has piece-wise constant acceleration. Since the MotionBest One minimizes the distance to the intersection, and the MotionBest Two maximizes the speed, the vehicle will only decelerate and accelerate with $a_{\text{max}}$ in MotionBest One and Two respectively. Combining this with the exhaustive discipline, the vehicle only decelerates once, and the behaviour consists of five parts:

- constant speed $v_{\text{max}}$ from $r_{TV}$ until $t_{\text{dec}}$,
- decelerating from $t_{\text{dec}}$ until $t_{\text{stop}}$,
- standstill from $t_{\text{stop}}$ until $t_{\text{acc}}$,
- accelerating from $t_{\text{acc}}$ until $t_{\text{full}}$,
- constant speed $v_{\text{max}}$ for $t \geq t_{\text{full}}$. 
To construct the closed-form solution, we start by determining $t_{\text{dec}}$, and from there the complete trajectory can be determined.

Keeping in mind MotionBest One, the first RV in a platoon will decelerate as if it were to reach a complete stop exactly at the intersection, regardless of whether this would have actually been necessary. The light will only turn green at the vehicle’s scheduled time, which could be later than when the vehicle started decelerating. Since the first part of MotionBest minimizes the distance to the intersection, the car continues with its initial speed $v_{\text{max}}$ until it has to break at a distance $1/2 \cdot v_{\text{max}}^2/a_{\text{max}}$ before reaching the intersection - the car will then decelerate with maximum rate and come to a complete stop exactly at the intersection. We equate the distance covered by driving and decelerating to the starting distance from the intersection, and solve for $t_{\text{dec}}$:

$$(t_{\text{dec}} - rt_{V}) \cdot v_{\text{max}} + 1/2 \cdot v_0^2/a_{\text{max}} = |x_{V,0}|.$$ 

This gives $t_{\text{dec}} = |x_{V,0}|/v_0 - 1/2 \cdot v_0/a_{\text{max}} + rt_{V}$.

Once the light turns green, MotionBest Two maximizes the speed of the car. Due to the discretization of MotionBest, this happens the first time-step after the light has turned green, but with the closed-form solution, this is not necessary and $t_{\text{acc}} = \text{gr}V$. If $t_{\text{dec}} + v_{\text{max}}/a_{\text{max}} < \text{gr}V$, the car will come to a full stop. In this case, $t_{\text{stop}} = t_{\text{dec}} + v_{\text{max}}/a_{\text{max}}$. Otherwise, the car will not come to a stop, and $t_{\text{stop}} = \text{gr}V$ such that this case is skipped. When the light has turned green at $t = \text{gr}V$, the car will be at a distance $x_{V}(\text{gr}V)$ from the intersection, with speed $v_{\text{gr}} = v_{\text{max}} - (t_{\text{stop}} - t_{\text{dec}}) \cdot a_{\text{max}}$. These values can be deducted from the trajectory formulae. Motion Best Two maximizes the speed, hence the car will accelerate with $a_{\text{max}}$, to get to maximum speed as quickly as possible. Once the car has reached maximum speed $v_{\text{max}}$ at time $t_{\text{full}}$, it will continue at full speed. In total, we find for the times:

$$t_{\text{dec}} = |x_{V,0}|/v_{\text{max}} - 1/2 \cdot v_{\text{max}}/a_{\text{max}} + rt_{V}$$

$$t_{\text{stop}} = \begin{cases} t_{\text{dec}} + v_{\text{max}}/a_{\text{max}} & \text{if } t_{\text{dec}} + v_{\text{max}}/a_{\text{max}} < \text{gr}V \\ \text{gr}V & \text{otherwise} \end{cases}$$

$$t_{\text{full}} = t_{\text{stop}} + (v_{\text{max}} - v_{\text{gr}})/a_{\text{max}}.$$

and for the acceleration, we have:

$$a(t) = \begin{cases} 0 & rt_{V} \leq t \leq t_{\text{dec}}, \\ -a_{\text{max}} & t_{\text{dec}} \leq t \leq t_{\text{stop}}, \\ 0 & t_{\text{stop}} \leq t \leq \text{gr}V, \\ a_{\text{max}} & \text{gr}V \leq t \leq t_{\text{full}}, \\ 0 & t \geq t_{\text{full}}. \end{cases}$$

Knowing $a(t)$, $x_{V}(t)$ can be computed by integrating twice, and using the boundary conditions $x_{V}(rt_{V}) = x_0$, and $v(0) = v_{\text{max}}$.

It is important to observe that the linear optimization method is dependent on the timestep to discretize the motion. Compared to the closed-form solution, this results in slightly different behaviour around $t_{\text{dec}}$ and $\text{gr}V$, because the acceleration cannot be changed continuously with MotionBest. This difference is illustrated in Figures 12 and 13, where accelerating and decelerating at slightly different instants results in different values for $x_{V}(t)$. Taking $\Delta t$ small enough will ensure these differences are negligible, but makes the linear optimization method very slow. The closed-form solution is therefore preferred for the trajectory planning of the vehicles.
5.3 closed-form solution for exhaustive policy

Figure 12: Overlaying the speed of an RV computed with MotionBest and the closed-form solution

Figure 13: Zooming in on the difference between MotionBest and the closed-form solution

Platoon of RVs We will now continue to construct the closed-form solution for RVs driving in a platoon with only RVs. We start with the second RV in a platoon, and then extend the formulae to hold for general platoons of RVs.

Let $V_1$ and $V_2$ be two consecutive RVs in the same platoon, where $V_1$ is the head of the platoon. Since the behaviour of $V_2$ is dependent on $V_1$, we write the times as $t_{stil,1}$ and $t_{stil,2}$ to denote $t_{stop}$ for $V_1$ and $V_2$ respectively. With $V_2$ joining the platoon of $V_1$, we have $gr_{V_1} = gr_{V_2}$. Given the trajectory of the first car in the platoon, the second RV is allowed to join this platoon if and only if it would be able to cross the intersection a distance $B_d$ from its predecessor. It is important to note, that once $V_2$ is at a distance $B_d$ from $V_1$, the speeds of their MotionBest trajectories should exactly line up. If at that moment, $V_2$ drove slower than $V_1$, MotionBest would not be minimizing its distance to the intersection. If $V_2$ drove faster, the distance between the two vehicles would become less than $B_d$, and the situation would become unsafe. Thus from the moment the two vehicles drive at a distance $B_d$, denoted with $t_{B_d}$, the speed of $V_2$ is exactly equal to the speed of $V_1$. Furthermore, observe that the distance between $V_1$ and $V_2$ before $t_{B_d}$ should be larger than $B_d$. Since MotionBest One minimizes the distance to the intersection, $V_2$ will brake at the latest possible moment with maximum rate. To illustrate the computation, Figure 14 depicts the behaviour of a vehicle $V_2$ with $v_{max} = 10$ and $a_{max} = 4$, that comes to a complete stop for 0 time.

In general terms, the total area under the graph (i.e. distance traveled) until $t_{full}$ equals

$$L_{V_2} = v_{max} \cdot (t_{full} - rt_{V_2}) - \frac{v_{max}^2}{a_{max}}. \tag{1}$$

Therefore, if $L_{V_2} \geq |x_{V_2,0}| + (x_{V_1}(t_{full}) - B_d)$, we are in the case that $V_2$ has to come to a complete stop.
Now consider a vehicle $V_2$ with $L_{V_2} \geq |x_{V_2,0}| + (x_{V_1}(t_{full}) - B_d)$. To construct the trajectory of $V_2$, $t_{dec,2}$ and $t_{acc,2}$ need to be computed, because $t_{stil,2}$ can be derived from $t_{dec,2}$. We have observed that the distance between $V_1$ and $V_2$ at $t_{B_d} \leq t_{full}$ equals exactly $B_d$, and that after this moment, the speeds of $V_1$ and $V_2$ exactly match up. In the case that $V_2$ has to come to a complete stop, note that $V_1$ also had to come to a complete stop - and more specifically, $V_1$ will be at a complete stop once $V_2$ also has reached a complete stop, otherwise $V_2$ would not have needed to come to standstill. Moreover, Motion Best One would not be minimizing the distance to the intersection if $V_2$ would not park right behind $V_1$. Hence at $t_{stil,2}$, $V_2$ is also exactly a distance $B_d$ behind $V_1$. Since $V_1$ is also at a standstill, we see that $x_{V_1}(t_{stop}) = x_{V_1}(gr_{V_1})$, and find:

$$t_{dec} = \frac{||x_{V_2,0}| + x_{V_1}(gr_{V_1}) - B_d}{v_{max}} - \frac{v_{max}}{2 \cdot a_{max}} + rt_{V_2}. \quad (2)$$

To minimize the distance to the intersection, $V_2$ decelerates at maximum rate, and we derive $t_{stil,2} = t_{dec,2} + \frac{v_{max}}{a_{max}}$. Moreover, Motion Best Two maximizes the speed, and comes into play once $V_2$ needs to be computed, because $t_{br,2} \leq t_{full}$, and from $t_{B_d}$ onward, the distance between $V_1$ and $V_2$ will equal $B_d$. Therefore, the $(v,t)$ graph of $V_2$ looks similar to Figure 14, except $v(t)$ does not go to 0. To compute the area under the graph, let $t_{br,2}$ be the total time that $V_2$ has to brake (i.e. $a < 0$), to find the following equations for the times and the area under the $(v,t)$ graph:

$$\begin{cases} t_{full} = t_{dec,2} + 2 \cdot t_{br,2}, \\ |x_{V_2,0}| + (x_{V_1}(t_{full}) - B_d) = (t_{full} - rt_{V_2}) \cdot v_{max} - t_{br,2}^2 \cdot a_{max}. \quad (3) \end{cases}$$

Solving this gives

$$t_{br,2}^2 = \frac{(t_{full} - rt_{V_2}) \cdot v_0 - (|x_{V_2,0}| + x_{V_1}(t_{full}) - B_d)}{a_{max}}. \quad (4)$$

In the case that $t_{br,2}^2 \leq 0$, $V_2$ does not have to break and we take $t_{br,2} = 0$.
5.3 closed-form solution for exhaustive policy

We find for the times of the second RV that does not have to come to a complete stop:

\[ t_{\text{dec},2} = t_{\text{full}} - 2 \cdot t_{\text{br},2}, \quad (5) \]
\[ t_{\text{acc},2} = t_{\text{full}} - t_{\text{br},2}, \quad (6) \]

and the total acceleration trajectory of the second RV that does not have to come to a complete stop becomes:

\[ a(t) = \begin{cases} 
0 & rt_{V} \leq t < t_{\text{dec},2}, \\
-a_{\text{max}} & t_{\text{dec},2} \leq t < t_{\text{acc},2}, \\
a_{\text{max}} & gt_{V} \leq t < t_{\text{full}}, \\
0 & t \geq t_{\text{full}}. 
\end{cases} \]

These formulae can be extended to a platoon of RVs, because each trajectory is computed with MotionBest according to the same principles. In the entire reasoning to construct the closed-form solution for \( V_{2} \), it was of no importance whether \( V_{1} \) was the first vehicle of the platoon, or simply a predecessor. The trajectory of each vehicle in a platoon of RVs consists of five different parts, and each of the times \( t_{\text{dec}}, t_{\text{stop}}, t_{\text{acc}} \) and \( t_{\text{full}} \) can be computed. Therefore, these formulae can be used to determine the trajectories of a platoon of RVs, and this has been implemented in Algorithm 5.

5.3.2 MotionSelf

Similarly to Section 5.1, we also want to construct a closed-form solution for the CAVs in our model. By basing our model on MotionSynthesize \([6, 7]\), we only need to establish that the resulting optimization model of 4 and MotionSynthesize are equal. In that case, Algorithm 1 of \([3]\), p. 209, can be used to describe the closed-form solution for the trajectories of CAVs.

**First CAV** First, consider a CAV that is the head of a platoon. Then MotionSelf aims to maximize the term stated on line 6 of Algorithm 4. The first term on this line can almost be neglected. This term is necessary such that MotionSelf will minimize the distance of the CAV to the intersection, but it is reduced by a large factor. Therefore, the objective to maximize the speed at and beyond the intersection has greater priority. Note that for a large part of the trajectory, these two factors are independent - except for the area close to the intersection. There, these factors determine where the CAV will come to a stop, and with what speed it will cross the intersection. Since each term of the first sum is less than 1, the second sum certainly receives priority if \( v_{\text{max}} > 1 \). Since MotionSelf does not have the equality constraint \( v[N] = v_{\text{max}} \) from MotionSynthesize, this second factor ensures the speed at the intersection (and beyond) is maximized. Since \( V \) has no predecessors, the speed at the intersection can be maximized - which is more important in the sum than minimizing the distance to the intersection. Therefore, the speed at the intersection and beyond will equal \( v_{\text{max}} \). The relations between \( x, v, u \) and \( t \) are equal to MotionSynthesize. Hence, the trajectory of a CAV as head of a platoon will be equal to MotionSynthesize.

**Platoon of CAVs** Next, consider a CAV within a platoon of CAVs, with at least one predecessor. In that case, the term on line 4 of Algorithm 4 is maximized. This term is equal to the term maximized in MotionSynthesize. The only difference now, is that MotionSelf does not have the equality constraint of \( v[N] = v_{\text{max}} \). We will now reason why all CAVs in a platoon of CAVs will cross the intersection with maximum velocity. With the reasoning of the previous paragraph, we have concluded that a CAV will cross the intersection with maximum velocity if it is the head of a platoon. If constraints would allow for it, MotionSelf would let the second CAV simply drive towards and cross the intersection with maximum speed - because maintaining the maximum speed would maximize \( \sum_{j=0}^{N} x[j] \). Constraints that could prevent this, are the CAVs predecessor and the CAVs scheduled crossing time. However, a CAV as head of a platoon will cross the intersection...
Algorithm 5: ClosedFormBest

**Input:** vehicle \( V_2 \), starting location \( V_2 \) denoted with \( x_0 \), if any: predecessor \( V_1 \) and trajectory of \( V_1 \) denoted with \( y(t) \)

**Output:** trajectory of \( V_2 \), denoted by \( w(t) \)

1. if \( V_2 \) first vehicle of a platoon then
2. \( t_{dec} = |x_0|/v_{max} - \frac{1}{2} \cdot v_{max}/a_{max} + rt_{V_2} \)
3. \( t_{acc} = gr_{V_2} \)
4. if \( t_{dec} + v_{max}/a_{max} \leq gr_{V_2} \) then
5. \( t_{stop} = t_{dec} + v_{max}/a_{max} \)
6. \( t_{full} = gr_{V_2} + v_{max}/a_{max} \)
7. else
8. \( t_{stop} = gr_{V_2} \)
9. \( t_{full} = gr_{V_2} + gr_{V_2} - t_{dec} \)
10. else
11. \( L = v_{max} \cdot (t_{full} - rt_{V_2}) - \frac{v_{max}^2}{a_{max}} \)
12. if \( L_{V_2} \geq |x_0| + (y(t_{full}) - B_2) \) then
13. Completely to standstill
14. \( t_{dec} = \frac{|x_0| + y(gr_{V_1}) - B_2}{v_{max}/a_{max}} - \frac{v_{max}}{2a_{max}} + rt_{V_2} \)
15. \( t_{stop} = t_{dec} + \frac{v_{max}}{a_{max}} \)
16. \( t_{acc} = gr_{V_1} \)
17. \( t_{full} = t_{acc} + \frac{v_{max}}{a_{max}} \)
18. else
19. \( t_{dec} = \sqrt{\frac{t_{full,t} + v_{max} \cdot |x_0| + y(t_{full}) - B_2}{a_{max}}} \)
20. \( t_{dec} = t_{full} - 2 \cdot t_{dec} \)
21. \( t_{acc} = t_{full} - t_{dec} \)
22. Construct \( w(t) \) as follows
23. if \( rt_{V_1} \leq t \leq t_{dec} \) then
24. \( w(t) = x_0 + v_{max} \cdot (t - rt_{V_2}) \)
25. else if \( t_{dec} < t \leq t_{stop} \) then
26. \( w(t) = w(t_{dec}) + v_{max} \cdot (t - t_{dec}) - \frac{1}{2} \cdot a_{max} \cdot (t - t_{dec})^2 \)
27. else if \( t_{stop} < t \leq t_{acc} \) then
28. \( w(t) = w(t_{stop}) \)
29. else if \( t_{acc} < t \leq t_{full} \) then
30. \( v_{acc} = \max(v_{max} - a_{max} \cdot (t_{acc} - t_{dec}), 0) \)
31. \( w(t) = w(t_{acc}) + v_{acc} \cdot (t - t_{acc}) + \frac{1}{2} \cdot a_{max} \cdot (t - t_{acc})^2 \)
32. else if \( t > t_{full} \) then
33. \( w(t) = w(t_{full}) + v_{max} \cdot (t - t_{full}) \)

at maximum velocity. Furthermore, observe that breaking further than a distance \( B_s \) behind its predecessor, would only increase the average distance to the intersection - this is illustrated in Figure 15. The green (\( V_1^1 \)) and orange line (\( V_2^2 \)) denote two different possible trajectories for \( V_1 \), but note that the average distance to the intersection (area between the green and grey line) of \( V_1^1 \) is larger than the average distance to the intersection of \( V_1^1 \). Hence, the distance to the intersection is minimized by decelerating as late as possible, right behind the predecessor \( V_0 \), and accelerating as early as possible - together with \( V_1 \). With \( V_0 \) crossing the intersection with \( v_{max} \), and \( V_1 \) accelerating together with \( V_0 \), but crossing the intersection later, we also conclude that \( V_1 \) crosses the intersection with \( v_{max} \). This idea can then be inductively extended to each CAV in a platoon consisting of only CAVs.

All in all, we conclude that each CAV in a platoon of CAVs will behave according to the
MotionSynthesize procedure - hence the closed-form solution computed in [3] can be used to describe the trajectories of platoons of CAVs. An algorithm for these trajectories can be found in [3], page 290.

Figure 15: Two different trajectories for a CAV $V_1$ with CAV $V_0$ as predecessor

5.3.3 Mixed Traffic

Now for the mixed traffic model, a closed-form solution can be established to describe the behaviour in the case of a platoon with both CAVs and RVs. In this subsection, we will give an argument on how to construct this closed-form solution, as well as which conditions should hold. The particular solution is left for future research.

It is important to note that only the behaviour of the second till last vehicles of mixed platoons need to be determined - since the trajectories of the head of the platoon and homogeneous platoons will be exactly as described in Sections 5.3.2 and 5.3.1. Similar principles from these sections can be used for all other vehicles in a mixed platoon.

We first consider a platoon with an RV as head of the platoon. The case distinction on lines 3 - 6 of Algorithm 4, makes MotionSelf in case of a predecessor the same linear optimization program as MotionBest Two. Therefore, any CAV in a platoon of RVs will behave like an RV keeping a distance $B_s$ from its predecessor - and thus the closed-form solution from Section 5.3.1 can be used to describe their behaviour.

In Figures 16 and 17, we have visualized the behaviour of an RV with a CAV driving behind it. As opposed to platoons of CAVs, note that the CAV decelerates close to the intersection, and does not cross the intersection with maximum velocity. Instead, it minimizes its distance to its predecessor like an RV would.

On the other hand, suppose we have a CAV as the head of a platoon, with an RV driving behind it. The CAV will decelerate (if needed) further away from the intersection, in order to maximize its speed when crossing the intersection. Note that the CAV will start accelerating before the light has turned green, to ensure the highest velocity when crossing the intersection. At the same time MotionBest One still determines the RV’s trajectory, where the distance to the intersection is minimized under the assumption that the light remains red indefinitely. Hence, the RV will also start accelerating when the CAV starts accelerating in order to minimize its distance to the intersection. However, while the light is still red, the RV follows a trajectory that would come to a complete stop at the intersection. Depending on $B_d$, $v_{max}$ and $a_{max}$, the RV might therefore need to break once more before accelerating and crossing the intersection. In Figure 18,
5 TRAJECTORIES

5.3 closed-form solution for exhaustive policy

Figure 16: Trajectories of a CAV behind an RV

Figure 17: Speed profiles of a CAV behind an RV

we have visualized the acceleration of two vehicles $V_1$ and $V_2$ approaching the intersection. $V_1$ is a CAV and head of the platoon, $V_2$ is an RV and the second vehicle in the platoon. $V_1$ is scheduled to cross the intersection at $t = 8$, which is for $V_2$ the same moment the light turns green. Observe that just before $t = 8$, $V_2$ starts decelerating, because the RV still assumes the light to remain red indefinitely. Only once the light has turned green at $t = 8$, the RV will start accelerating. This behaviour would complicate the closed-form solution for mixed traffic platoons, since the discrepancy can continue in all vehicles in the same platoon.

However, this complication can be avoided in two ways: 1) giving RVs a green light when the head CAV starts accelerating, or 2) having $B_d \geq \frac{v_{\text{max}}^2}{2 \cdot a_{\text{max}}}$. The first option requires a small change in the scheduling algorithm, and ensures that RVs driving behind CAVs will not have to break before the intersection. With the second condition, we let the headway between an RV and its predecessor be larger than its braking distance. When $V_1$ is exactly at the stop line, $V_2$ will be at least a distance $B_d$ behind it. If $B_d$ is larger than the breaking distance, $t_{\text{dec}}$ from the trajectory of MotionBest One will be after the moment the light has turned green - hence when MotionBest Two has already started to maximize the speed of $V_2$. Hence, we can avoid RVs decelerating when driving behind a CAV by taking $B_d$ larger than the braking distance of RVs.

Under the assumption that $B_d \geq \frac{v_{\text{max}}^2}{2 \cdot a_{\text{max}}}$, the construction of the closed-form solution of a platoon with head CAV becomes straightforward. For a platoon of length $n$, the trajectories of the first $k$, $1 \leq k \leq n$ CAVs can be described as in Section 5.3.2. The next vehicle, an RV, will only decelerate to keep a safe distance from its predecessor, and will accelerate with its prede-
censor. Therefore, the RV will behave as a CAV, and its trajectory can be taken from Section 5.3.2, with a headway distance of $B_d$. We have visualized this in Figures 19 and 20. Both the CAV as well as the RV cross the intersection at each of their scheduled crossing times. Even more so, both vehicles cross the intersection with maximum velocity. This behaviour is exactly as described for platoons of CAVs, hence the RV behaves like a CAV keeping a distance $B_d$ from its predecessor.

In total, we have argued how to construct the closed-form solution of RVs and CAVs. An important observation is that the behaviour of each vehicle in a platoon becomes dependent on the first vehicle of a platoon. All vehicles in a platoon with head RV will behave as RVs, whereas all vehicles in a platoon with head CAV will behave as CAVs. As a result, not all vehicles in a platoon with head RV will cross the intersection at maximum velocity. On the other hand, having a CAV at the front of a platoon allows for all vehicles of that platoon to cross at maximum velocity. The computations of Sections 5.3.1 and 5.3.2 can be used to construct these trajectories.

6 Results

In this section we will take a look at the results of the algorithms we have described in the previous chapters. First the trajectories of an isolated pair of vehicles will be considered. Since already discussed in length in Section 5, we will only remark on some interesting observations. Next, the influence of CAVs at an intersection with a fixed-time process will be investigated. Lastly, the behaviour and performance of the proposed Platoon Forming Algorithm will be considered. The average delay of vehicles will be compared for different proportions of CAVs in traffic. Furthermore, the average delay will be compared to an intersection with a fixed-time process.

For the results, we present examples where we set the values of the parameters as defined in Table ??.

The algorithms are implemented in Python in a Jupyter Notebook. The linear optimization modeller PuLP has been used to program MotionBest and MotionSelf.

<table>
<thead>
<tr>
<th>Control region</th>
<th>$l$</th>
<th>$B_s$</th>
<th>$B_d$</th>
<th>$v_{max}$</th>
<th>$a_{max}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>2</td>
<td>6</td>
<td>12</td>
<td>10</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 1: Values used for parameters for the simulation results
6.1 Trajectories

The two algorithms, MotionBest and MotionSelf as described in Section 5, will be used to determine the trajectories of RVs and CAVs. First, the behaviour of two RVs driving behind each other will be considered, and we will demonstrate the influence of the timestep $\Delta t$. Next, we will look at the behaviour of two CAVs driving behind each other, and illustrate the influence of the implementation of SafeDist.

**MotionBest** Consider two RVs ($V_1$ and $V_2$) approaching the intersection from the same lane. Let $V_1$ arrive at a distance 60 m from the intersection at $t = 0$, and $V_2$ at $t = 1.5s$. We examine two cases with varying green times, that are of interest for the behaviour of these vehicles: $gr_{V_1} = 6$, and 10. In each of the figures below, the green dotted line denotes the moment the light has turned green - thus when MotionBest Two comes into play. In the following results, MotionBest One and MotionBest Two are used both with $N = 1000$ timesteps.

In Figure 21, the light turns green at $t = 6s$, exactly at the moment $V_1$ could reach the intersection. Had $V_1$ been a CAV, the car would have known the green time, and could continue with maximum speed. However, the two RVs that are considered are unaware that the light will turn green at $t = 6s$. This unawareness is captured in MotionBest One, where $V_1$ will brake as if the light would remain red. Observe that in Figure 21, $V_1$ and $V_2$ decelerate before the light turns green, but neither one comes to a complete standstill. Once the light has turned green, both vehicles will accelerate again, to reach $v_{max}$.

![Figure 21: Trajectory and speed of $V_1$ and $V_2$ when the light turns green at $t = 6$.](image)

In Figure 21 we can clearly see that $V_1$ starts both decelerating and accelerating earlier than $V_2$. Considering the distance between $V_1$ and $V_2$: $d(V_1, V_2)(t) := |x_{V_1}(t) - x_{V_2}(t)|$, visualized in Figure 22, observe the following: when the light turns green (the green dotted line), the distance between $V_1$ and $V_2$ is larger than $B_d = 12m$. The two vehicles are at a distance $B_d$ and at equal speed once $V_2$ starts accelerating - exactly what we argued in Section 5.3.1.

When zooming in on the distance between $V_1$ and $V_2$, the influence of the timestep and interpolation method becomes visible. First, as displayed in Figure 13, due to the timestep the vehicles are unable to decelerate and accelerate at specific moments in-between timesteps. Therefore, the distance between $V_1$ and $V_2$ will not equal exactly $B_d$ when $V_2$ starts accelerating. Moreover, depending on the interpolation method used for $x$, $v$ and $u$, a pattern can be observed in the distance between $V_1$ and $V_2$, as in Figure 23. This pattern arises from $V_1$ and $V_2$ not being at exactly a distance $B_d$ once they start accelerating.
MotionSelf  Now consider the same situation as before, except with \( V_1 \) and \( V_2 \) both CAVs. Using MotionSelf to compute their trajectories, we want to remark upon the significance of the scheduled times of \( V_1 \) and \( V_2 \).

Let \( V_1 \) and \( V_2 \) arrive at a distance 60 m from the intersection at \( t = 0 \) s and \( t = 1.5 \) s respectively.

Let \( V_1 \) be scheduled to cross at \( t = 8 \) s. With \( c_{V_1} > a_{V_2} \), \( V_2 \)'s scheduled time will also be delayed to keep a safe distance between the two vehicles. \( V_2 \) will be scheduled to cross at \( t_{S_{V_1}} \), where \( t_{S_{V_1}} \) will be computed with an implementation of \texttt{SafeDist}. With the first implementation, \( V_2 \) receives a scheduled time \( a_{V_2} \approx 8.60 \) s. In Figure 24, the trajectory and speeds of both vehicles has been visualized. As expected, both vehicles cross the intersection at their scheduled times, with maximum speed. Also note that \( V_1 \) and \( V_2 \) decelerate ample before the intersection, to ensure their speed at (and after) the intersection is maximized.

To show that the accuracy of \texttt{SafeDist} has a significant influence on the behaviour of \( V_2 \), we consider the same situation, except with a different implementation of the \texttt{SafeDist} function. The different implementation causes \( V_2 \) to be scheduled to cross at \( t \approx 8.65 \) s. Observe in Figure 25 that \( V_2 \) displays similar behaviour as in Figure 24, except right before its scheduled crossing time. This is a direct consequence of the later crossing time: once \( V_2 \) has started accelerating, it is at a distance \( B_s \) from \( V_1 \). However, due to the inaccuracy of \texttt{SafeDist}, \( c_{V_2} \) is not the moment when \( V_1 \) has crossed the stop line exactly a distance \( B_s \). This discrepancy requires \( V_2 \) to decelerate right before crossing the intersection, in order to cross exactly at \( c_{V_2} \) - with \( d(V_1,V_2)(c_{V_2}) > B_s \). This behaviour can be avoided by implementing a more accurate version of \texttt{SafeDist}.  

Figure 22: Distance between \( V_1 \) and \( V_2 \)
Figure 23: Zooming in on the distance between \( V_1 \) and \( V_2 \)

Figure 24: Trajectory and speed of \( V_1 \) and \( V_2 \) when scheduled to cross at \( t = 8.0 \) s and \( t \approx 8.60 \) s respectively.

Figure 25: Trajectory and speed of \( V_1 \) and \( V_2 \) when scheduled to cross at \( t \approx 8.65 \) s.
6 RESULTS

6.2 Influence CAVs with fixed-cycle switching process

To visualize the general influence of CAVs in traffic, consider an intersection with two lanes and a fixed-time switching process. We will consider the intersection with only RVs present, and compare it with only CAVs present. Furthermore, we will also look at the intersection with a certain fraction of CAVs present in arriving traffic. We assume the inter-arrival times of vehicles at each lane are drawn from a shifted Exponential distribution with parameter \( \lambda \). Under the assumptions that \( B_d \geq B_s \) is a safe distance for all vehicles, and that vehicles enter the control region at a safe distance from each other at speed \( v_{\text{max}} \), the time between arriving vehicles should be at least \( B_d v_{\text{max}} \). In order to also account for long queues, the Exponential distribution is shifted with \( \delta = 1.5 \cdot B_d v_{\text{max}} = 1.8 \) seconds, to ensure a safe distance between all vehicles. Furthermore, a fraction \( p_A \) of the arriving vehicles will be CAVs, the rest will be RVs. MotionBest and MotionSelf are used to describe the trajectories of vehicles. For all results in this subsection, we have set \( G = 6 \) and \( S = 2 \) seconds. The control region is 500 m, such that each vehicle is registered 500 m from the intersection.

**Fixed-time** A fixed-time switching process in practice means that the light cyclically turns green at each lane for a fixed amount of time: the green interval \( G \). Once a lane has received a green light for \( G \) time, the light turns yellow for \( S \) time, after which it turns red and the next lane receives a green light. Vehicles that can cross the intersection a safe distance from their predecessor and within the green interval, are scheduled to cross the intersection in that green interval. Note that with the MotionBest procedure, it could occur that vehicles cross when the light has already turned yellow, because the scheduled time does not always coincide exactly with the actual crossing time. Therefore, a positive yellow time \( S \) is necessary to ensure the safety of the intersection. This yellow time can be compared to the required switch-over time at intersections. Vehicles that cannot cross within the green interval are scheduled in the next green interval at their lane.

**RVs** In Figure 26 the fixed-time switching process with only RVs for \( \lambda = 1/4 \) has been visualized. The vertical green and orange lines denote the instances at which the light turns green and yellow at that lane respectively, for \( G = 6 \) and \( S = 2 \). Observe that the red and orange line intersect around time \( t = 3810 \), which is an example of a vehicle crossing the intersection after the light has turned yellow. Furthermore, note that we have simulated the intersection for 4000 seconds and queues have not accumulated. Hence a \( \lambda \) of 1/4 can be considered to be a low load for this intersection.

---

Figure 25: Speed profile of two CAVs with a less accurate implementation of the Motion algorithm.

---

Platoon Forming Algorithms for an Intersection with Mixed Traffic
6.2 Influence CAVs with fixed-cycle switching process

Therefore, we will next take a look at a (more) heavily loaded intersection. The accumulation of the queues for an arrival process with \( \lambda = 1 \) is depicted in Figure 27. When zooming in on a system with a heavy load in Figure 28, observe that each green interval 4 vehicles are able to cross the intersection. Hence every 14 seconds, exactly 4 vehicles are served in one lane. In order to ensure a stable system, the arrival rate at each lane should therefore certainly be less than \( 4/14 \) vehicles per second. This means that the inter-arrival times between vehicles should be greater than \( 14/4 = 3.5 \) s. For any \( \lambda \), vehicles arrive every \( 1.8 + 1/\lambda \) seconds on average. A necessary condition for a stable system would be for \( 1/\lambda > (3.5 - 1.8) = 3.5/1.7 \). Hence, we conclude that \( \lambda = 1 \) surely results in an unstable system, which shows by the growth of the queues in Figure 27. At both lanes, the queues grow gradually, until they have reached the end of the control region. In that situation, the linear optimization problem has no feasible solution, and the algorithm does not finish.

### CAVs

In Figure 29 the fixed-time scheduling process with only CAVs for \( \lambda = 1/1.5 \) has been visualized. Comparing with 27, note that more vehicles are able to cross the intersection during one green interval: around time \( t = 3520 \) s, nine CAVs can cross the intersection in a green interval. In this interval, observe that the time (and distance) between the last vehicle and its predecessor is larger than between the other cars. Therefore, we expect that even more than nine CAVs could cross the intersection during one green interval. In fact, since platoons of CAVs cross at maximum speed, the time between CAVs at the intersection is \( B_s/v_{\text{max}} = 6/10 \), hence at most
\( G/(B_s/v_{\text{max}}) = 10 \) CAVs can cross during one interval.

With 10 CAVs being able to cross every 14 seconds, the system would become unstable for inter-arrival times less than \( 14/10 = 1.4 \) seconds. However, having shifted the Exponential(\( \lambda \)) distribution with 1.8 seconds, this system with only CAVs will never become unstable. Furthermore, this also means that each waiting CAV will be served in the next green interval, i.e. the waiting time of a vehicle will be at most 14 seconds. Comparing this to a system with only RVs as in Figure 28, we see that it is possible for vehicles to wait multiple rounds of green intervals before being allowed to cross the intersection. The system with only RVs has a lower arrival rate and is unstable, whereas the system with only CAVs is stable for any arrival rate and even ensures waiting times less than 14 seconds.

\[ \text{Figure 29: An intersection with only CAVs arriving with a } \lambda = 1/1.5 \]

**Mixed traffic** Lastly, the influence of different proportions of CAVs in mixed traffic will be investigated. Consider an intersection with the same fixed-time switching process, where a fraction \( p_A \) of the arriving vehicles is a CAV. We have simulated the arrival process with \( \lambda = 1/2, 3 \), for both \( p_A = 0.2 \), and \( p_A = 0.8 \). The behaviour of the vehicles at the intersection is visualized in Figures 30 and 31. The process was simulated until \( t = 4000 \), and observe that queues have not accumulated beyond the control region in either figure. Therefore, both systems are expected to be stable with the previously mentioned parameters.

\[ \text{Figure 30: } p_A = 0.2, \lambda = 1/2, 3 \]  
\[ \text{Figure 31: } p_A = 0.8, \lambda = 1/2, 3 \]
Comparing between the two figures, observe that the queues at Figure 30 are much longer than in Figure 31. Vehicles that have come to a standstill even need to wait more than one green interval at their lane, before being able to cross the intersection. However, around \( t = 3470 \) s in Figure 30, the presence of CAVs allows for more vehicles to cross during one green interval, than with no CAVs present (around \( t = 3415 \)). A larger proportion of CAVs results in more vehicles crossing each green interval - limiting the build-up of queues. This can be seen when comparing Figures 30 and 31.

### 6.3 Exhaustive scheduling algorithm

Finally, the behaviour and performance of our proposed exhaustive scheduling algorithm will be investigated. The evolution of the queues at the intersection will be considered, as well as the delay for different proportions of CAVs \( p_A \). The mean delay using Algorithm 1 will be compared to the mean delay with a fixed-time process for different proportions of CAVs. This proportion of CAVs \( p_A \) is called the penetration rate of CAVs. Since our model has become too complex for analytical results, the results are based on simulations. For each simulation, the scheduling process has been simulated for the arrivals of 4000 vehicles, in order to get the system into steady-state. The inter-arrival times between vehicles are drawn from an Exponential(\( \lambda \)) distribution shifted with \( \delta \) time. As previously, the distribution will be shifted with \( \delta = 1.8 \) seconds, to ensure a safe distance between all vehicles. Furthermore, we take the set-up time \( S = 1 \), and the control region to be 500 m. Algorithm 1 is used to determine the scheduled crossing times of vehicles, and MotionBest and MotionSelf were used to compute the trajectories of RVs and CAVs respectively.

#### Delay

In order to compare the influence of CAVs, a performance measure needs to be established by which compare different systems can be compared. Having the model of our intersection based on a polling model, we are inclined to use the delay as a performance measure. Except in our model, service times are dependent on the service policy and other vehicles. The delay of a vehicle can also be interpreted as the difference between arriving at a completely empty intersection, as opposed to the actual situation. Consider a vehicle \( V \) arriving at the control region of size \( d_V \) at time \( r_{tv} \). Furthermore, at time \( t_{DV} \), the vehicle has crossed the stop line by a distance \( B_d \). Had the intersection been completely empty, \( t_{Dv} \) would be \( r_{tv} + (|x_{V,0}| + B_d)/v_{max} \). Therefore, we arrive at the following definition:

**Definition 6.1.** We define the delay, \( D_V \) of a vehicle \( V \) as

\[
D_V := t_{DV} - r_{TV} - (|x_{V,0}| + B_d)/v_{max}
\]

This definition will be used in this section, to compare the influence of different penetration rates of CAVs on the average delay of vehicles arriving at an intersection.

#### RVs

The first intersection we will consider, is with a penetration rate of 0, i.e. only RVs arrive at this intersection. The arrival and scheduling process has been simulated for \( \lambda = 1/2.3 \) and for \( \lambda = 1 \). In Figures 32 and 33, the behaviour of the RVs is visualized, when using Algorithm 1 to schedule the green light. In Figure 32 the intersection has a relatively low arrival rate, as opposed to Figure 33. The effect of platoon-forming becomes more evident at higher arrival rates, where vehicles’ delays become larger and more vehicles can come within \( B_d \) distance of their predecessor. Especially in Figure 33, the platoon forming property of the algorithm is visible: all waiting vehicles on one lane are allowed to cross the intersection in one platoon. This pattern is typical for the exhaustive polling policy that we have considered. Observe that in Section 6.2, a \( \lambda = 1 \) for a fixed-cycle process resulted in an unstable system, whereas the system does not become unstable when using our proposed Platoon Forming Algorithm.
6 RESULTS

6.3 Exhaustive scheduling algorithm

CAVs To investigate what influence CAVs will have on the queue development and delay of vehicles, we also consider the other boundary case of $p_A = 1$, i.e. only CAVs approach the intersection. The arrival process has been simulated for a low load with $\lambda = 1/2.3$, and a high load with $\lambda = 1/0.1$. In Figures 34 and 35 the trajectories of approaching vehicles are visualized for the respective $\lambda$'s. At the relatively calm intersection, platoons of small sizes are created and cross the intersection together. With a higher arrival rate, the platoons also grow in size. Observe that the size of platoons at the intersection with only RVs in Figure 33 are significantly larger than with CAVs in Figure 35, even though the second situation has a higher arrival rate. We argue this to be a result of at least the chosen values for $B_b$, $B_d$ and $\delta$. Due to the inter-arrival times being drawn from a shifted exponential distribution, arriving vehicles will be at least $\delta = 1.8$ seconds behind their predecessor. In other words, they will arrive $\delta \cdot v_{max} = 18m$ behind their predecessor, while the distance CAVs need to keep from their predecessor is only $6m$. No CAV entering the control region would be allowed to join the platoon of their predecessor, unless their predecessor was (significantly) delayed. Expecting the delays of an intersection with only CAVs to be less than with RVs, fewer vehicles are able to join one platoon. These factors could, at least partly, explain the relatively small platoon sizes in Figure 35, compared to Figure 33.

Mixed traffic To further investigate the performance of the proposed algorithm for the scheduling and trajectories of vehicles, the intersection has been simulated for different penetration rates of CAVs, $p_A$. The delay and queue development will be compared with the fixed-time switching process, as well as for the different penetration rates.
6.3 Exhaustive scheduling algorithm

Figures 36 and 37 show intersections with $\lambda = 1$ and $p_A = 0.2$ and $0.8$ respectively. Observe that shorter queues have developed compared to an intersection with only RVs in Figure 33, and the pattern from the platoons is less obvious. When taking a closer look, note that each vehicle decelerates only once before being allowed to cross the intersection. Although having the same arrival rate, platoon sizes for $p_A = 0.2$ in Figure 36 are significantly larger than for $p_A = 0.8$ in Figure 34, indicating higher delays.

Comparing between the exhaustive algorithm and the fixed-time system, is most interesting for systems with higher arrival rates, due to more visible platoon-forming. For $\lambda = 1$ and a penetration rate of 0 CAVs, the simulated intersection with the exhaustive scheduling algorithm shows queue developments and stability. However, the intersection with a fixed-cycle switching process becomes unstable for $\lambda = 1$. Therefore, the queue developments of the exhaustive algorithm will be compared to the queues with a fixed-cycle process. Furthermore, the delay between both policies can be compared for penetration rates higher than 0.5. The stabilization of the system shows that using Algorithm 1 evidently improves vehicles’ delays compared to the fixed-cycle process for $p_A \leq 0.5$.

The intersection has been simulated for $\lambda = 1$ for different $p_A$, and the mean delay of the last 2000 (out of 4000) scheduled vehicles have been computed. In Figure 38 the mean delay with a 95% confidence interval of both scheduling policies has been visualized, for increasing penetration rates of CAVs. Observe that the mean delay decreases when the penetration rate $p_A$ increases. For the exhaustive algorithm, increasing the penetration rate from $p_A = 0$ to 0.3 more than halves the mean delay of vehicles. The fixed-time policy becomes unstable for $p_A \leq 0.4$ and also experiences a high mean delay for $p_A = 0.5$: the mean delay is tripled compared to $p_A = 0.6$. For $p_A \geq 0.8$, the mean delay of the fixed-cycle policy does not seem to improve much, whereas the mean delay of the exhaustive policy continues to decrease for higher penetration rates.

For higher arrival rates, the influence of the penetration rate of CAVs becomes even more visible. When considering an intersection with $\lambda = 1/0.2$, the queue grows beyond the intersection for low penetration rates $p_A$. However, comparing the average delays for $p_A = 0.5$ to $p_A = 0.8$, the average delay still improves from 9.0838 seconds to 3.1801 seconds. In Figures 39 and 40, this difference becomes visible in the difference in platoon sizes. The larger platoon sizes indicate vehicles experience larger delays on average.
7 Conclusion and discussion

Throughout this paper an isolated intersection has been considered, with specific assumptions on the behaviour of RVs and CAVs. In this section, we will discuss conclusions that can be derived from the results, as well as the restrictions of our model. Possible ideas for future research will be proposed.

A Platoon Forming Algorithm was considered, to schedule mixed traffic arriving at an intersection. The two algorithms (MotionBest and MotionSelf) have been proposed to describe driving trajectories of RVs and CAVs approaching an intersection. These algorithms can be used to describe vehicles’ behaviour in mixed traffic, and even have a closed-form solution to the trajectories. The most important observation was that the type of vehicle at the head of a platoon determines the behaviour of the rest of the vehicles in the platoon. When approaching vehicle consists of a larger proportion of CAVs, the probability that the head of a platoon is a CAV becomes larger, which allows for a more efficient use of the intersection. At intersections with the fixed-light switching process, as well as our proposed Platoon Forming Algorithm, the results show that the presence of CAVs significantly improves queue developments and delays. The exhaustive algorithm we have proposed to schedule vehicles arriving at an intersection (Algorithm 1) is also an improvement of the fixed-cycle switching process that was considered on the performance measure of delay. These
results are valid for the model that we have constructed for an isolated intersection with mixed traffic, which might be restricted due to assumptions.

Firstly, we have considered a very simple intersection, with only two approaching lanes. Although the model could be extended to an intersection with \( n \) lanes, only one lane at a time can receive a green light. Comparable intersections do exist in real life, though many more intersections have lanes crossing in different directions, creating subsets of different lanes that could receive a green light at the same time. Adapting our scheduling algorithm to better fit more complex intersections, or even networks, could be a topic for future research.

Another assumption of our model is on the behaviour of RVs and CAVs. All vehicles have been assumed to drive at least a safe distance from each other at all times. We remark that ANWB recommends a two-second following distance \([22]\): each successor should pass the same location at least two seconds later than its predecessor. However, note that the distance of this recommended following distance is dependent on the velocity of the vehicles. More distance is covered in two seconds when driving at \( v_{\text{max}} \) - thus more distance must be kept between vehicles at higher velocities. On the other hand, in our model vehicles need to keep the same safety distance at all speeds. To make the model more realistic, it could be interesting to look into implementing the two-second rule. Instead of distance, time could be used as the constant between cars when scheduling the intersection. Furthermore, the trajectories for vehicles can also be adapted to keep a minimum amount of time between vehicles, instead of distance. This would be interesting to look into for further research.

Another assumption on the driving trajectories of both RVs and CAVs, is that they are deterministic. Even more so, all RVs are assumed to drive according to the exact same principle, whereas in reality, distinct vehicles drive differently. To make the model more realistic, the behaviour of the RVs could for example become stochastic. This could be done by randomizing the moment the RV will break within some bounds for safety. The RVs could then gradually come to a complete stop at the intersection, either cruising at a speed lower than \( v_{\text{max}} \), or decreasing their speed continuously. By altering the assumptions on the trajectories of RVs, the benefits of the proposed scheduling algorithm can be determined more accurately. If the proposed scheduling algorithm significantly improves the throughput at the intersection in a realistic scenario, it could be looked into how to implement this algorithm at a real intersection.

Lastly, it would also be interesting to more closely analyze the scheduling algorithm to even further optimize the intersection. Finding a formula to determine the service times, could help to analytically approximate the waiting time of vehicles. Note that the service time of each platoon is completely determined by its composition and scheduled crossing time. Deriving these service times per platoon, can allow us to use formulae from polling models to determine various performance measures. Furthermore, it allows for a comparison of our proposed scheduling policy against other well-known policies. The proposed algorithm might be improved for example by changing the cyclical switching of the server. A first-come-first serve algorithm could be considered, such that the server switches to the platoon closest to the intersection instead of cyclically. Another option might be to switch to the lane that has already formed the largest platoon. In a model with multiple approaching lanes, these changes affect the outcome of the scheduling policy. The effect of such model changes, might be investigated.

One last point we want to touch upon, is the choice of using PuLP for the linear optimization programs. We first implemented MotionBest and MotionSelf in scipy's linear optimization program (scipy.optimize.linprog). However, even with feasible conditions, this program returned incorrect solutions that did not obey the bounds of the linear optimization problem. Therefore, we would not recommend scipy’s linear optimization program to implement MotionBest and MotionSelf. Although PuLP does return correct results, in case of an infeasible solution, PuLP will either return a solution that does not obey the bounds, or it continues running and does not stop.
Beware of this behaviour when testing systems that could extend beyond the border of the control region. We recommend building in a mechanism that checks whether the queue extends close to the border.
References


[22] “Bumperkleven - ANWB.”