BACHELOR

The performance of platoon forming algorithms in a network of intersections

Clement, Marijn I.

Award date:
2020

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Supervisor:
Marko Boon

Bachelor Final Project: The performance of platoon forming algorithms in a network of intersections

M.I. (Marijn) Clement (1009187)
m.i.clement@student.tue.nl

Contents

1 Introduction

Over the past century the number of vehicles has increased enormously. This has inevitably led to an congestion of the road network with delay and environmental damage as a result. One of the main contributors to the congestion problem are intersections. They are vital for a well-connected road network, but the classic traffic light based intersection does not make optimal use of the road. Hence, an increase in efficiency at intersections could have a positive influence on the amount of delay. As this is desirable, there have been continuous efforts to improve them.

With the rise of technology, research has expanded to a future scenario where all vehicles can drive autonomously. Furthermore, it is often assumed that they are able to communicate with other autonomous vehicles and the infrastructure through vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication respectively. For example, V2I communication could be between a vehicle and an intersection manager, the latter of which organizes the intersection. In this scenario, there is no need for actual traffic lights as visual cues are unnecessary because of the V2V and V2I communication. This opens up the possibility to organize the intersection in many different ways. One of the promising options is platooning. In this approach, vehicles form groups and traverse the intersection closely after each other at high velocity. Platooning allows the intersection to be used extremely efficiently. There are many different algorithms to sort the vehicles into platoons. These algorithms can be compared in terms of, for example, mean delay or fairness. For an isolated intersection, the literature is rich and the performance of these algorithms has been studied thoroughly. In the scenario of an isolated intersection, a certain arrival process has to be assumed. Often, this arrival process is modelled as a Poisson process. However, this arrival process is likely not realistic in a network, as vehicles depart from an intersection in platoons and thus also arrive at the next intersection as a platoon. Therefore, the arrivals are no longer independent, as is assumed in the Poisson process. The effect of this dependence in arrival times on the performance of different platoon forming algorithms has not been studied in depth.

In this bachelor final project, we will use simulation to study this effect. We will first give a short overview of the existing literature in Chapter 2. Next, our model will be explained in detail in Chapter 3. The simulation we use to obtain our results is described in Chapter 4. Then we go on to explain our results in Chapter 5 and draw a conclusion in Chapter 6.

2 Literature overview

The approaches to the congestion problem found in literature differ significantly. Several studies are devoted to options which are relatively easy to implement into the current road network. For example, instead of statically timed traffic lights, queue lengths in the lane can be taken into account. This was studied using fuzzy logic by e.g. Tian, Zhao and Yi [9]. They propose a method which needs as input the number of vehicles currently in the green phase of the traffic light and the number of vehicles who at the same time are in the red phase. These two numbers belong in one of the following four categories: zero, small, medium or big. If the category of the amount of vehicles in the green phase is larger than that of the number of vehicles in the red phase, the green phase is extended. Otherwise, it is terminated. The results show an improvement in performance.

More in line with this project are the studies which consider the scenario of fully autonomous vehicles. Dresner and Stone [4] have suggested a grid-based reservation policy. The intersection is divided into $n * n$ tiles. Based on the expected arrival time at the intersection, vehicles try to make a reservation for the tiles of the intersection that they need to traverse it. If a reservation has already been granted to another vehicle for any of the tiles, the vehicle slows down and tries again in the next time step. Note that this is a first come, first served (FCFS) policy. This approach is a drastic improvement over the normal traffic light system. A similar approach is taken by Wuthishuwong, Traechtler and Bruns [12]. In contrast to Dresner and Stone [4] however, the intersection manager tells the vehicles when to pass, instead of the vehicles having to send a new request upon dismissal. By discretizing both time and distance, it becomes clear until when a vehicle is not allowed to pass because of its predecessor having already reserved some of the necessary space on the intersection. The allowed crossing time for each vehicle is thus dependent on its predecessor. This results in a problem for which the solution is calculated using the forward Euler method. Note that in this study, in contrast to Dresner and Stone [4], vehicles are allowed to turn. The results show a decrease in delay when compared to a traditional traffic light situation.

Another approach is offered by Miculescu and Karaman [6]. Their approach is based on a polling system (see Section 3.1). They explain several polling policies. In the exhaustive policy, the server (in our case the intersection) is serving one queue until it is empty. In the gated policy, the server takes a snapshot of the queue whenever it switches to a new queue. Only the vehicles in the snapshot are then served. In the k-limited policy, the server serves a queue until k vehicles have passed, or the queue is empty, whichever comes first. The time at which a vehicle is supposed to pass is dependent on the polling policy of the intersection. To ensure that a vehicle reaches the intersection safely and on time, they also propose a speed profiling algorithm. This is a procedure that generates the exact trajectory of a vehicle when it enters the control region. This trajectory is dependent on the trajectory of the predecessor vehicle and the time that the vehicle is supposed to reach the intersection, as decided upon by the intersection manager. The result of this approach is that vehicles in the same lane get closer to each other and pass the intersection as a platoon. Another example of platooning can be found in Tachet et al. [8]. They compare the traffic light scenario to a FCFS policy and a platoon-forming policy. In the platoon forming policy, vehicles which are close enough to their predecessor are allowed to cross the intersection right behind them as long as the maximum platoon size has not been reached. The platoon forming policy is superior when it comes to delay, but it has to be noted that it is less fair as it is no longer FCFS. This trade-off is inherent to platoon-forming policies. Bashiri, Jafarzadeh and Fleming [1] look at the scenario in which there are already platoons in order. They study in which order multiple platoons approaching an intersection should cross. They use cost functions based on the waiting time and platoon size to compare the different passing order possibilities. Next, they compare stop signs with a fixed passing order to an approach minimizing the average delay and an approach minimizing the average delay while penalizing large deviations from the mean. Once again, the trade-off between fairness and delay is of importance, as minimizing the mean delay results in large deviations from this mean.

More approaches to find optimal solutions for the intersection management problem have been studied, such as genetic algorithms in e.g. Yan, Dridi and Moudni [13]. They first assign each vehicle to a passing group based on compatible streams and arrival times. The problem remaining is to find an optimal sequence for all the groups. This is solved with the use of a genetic algorithm: they start with a set of possible passing

sequences and take the two best ones. Then they create 'offspring' using two-point crossover. This means that two points in the 'chromosome' are randomly chosen and that the bits in between the two points are swapped between the parents. There is also a probability of a mutation appearing, which means that the passing order is changed. This process is iterated over several 'generations' to find a near-optimal solution. Also auction-based approaches have been explored by e.g. Carlino, Boyles and Stone [3]. In this approach, vehicles bid money against each other for access to the intersection. Only the vehicles which are first in their lane are eligible to pass, but vehicles behind them can also bid to improve their chances of crossing sooner. They then explain several bidding approaches: the free-rider never bids, the static wallet bids a constant amount and the fair wallet divides the money for the entire trip fairly over all intersections. Next, they introduce the system wallet, which helps bid for vehicles who have been waiting for a long time to increase fairness. Finally, they simulate this approach for multiple cities and compare the results from the FCFS policy to several different auction scenarios. The auction-based system mostly performs slightly better than FCFS.

Most of the mentioned works so far only look at a single intersection and assume a certain kind of arrival pattern for vehicles from each direction. In a network of intersections, the arrivals would not be following this pattern anymore after the first intersection and in the case of platoon forming algorithms, vehicles would arrive in groups instead of alone. Thus, an important question to answer is whether the proposed methods still function approximately the same when they are implemented in a network. We will now shortly discuss a few studies which have considered more than a single intersection.

Liu, Wang and Hoogendoorn [5] have looked at taking into account downstream vehicle queues when organizing the intersection. They also studied the merging of queues with arriving platoons. They make use of a speed profiling algorithm to ensure maximum use of the green light period, while maximizing driving comfort for the vehicles who cannot pass during this period. However, they do so in an environment with static traffic lights. Furthermore, they don't allow the vehicles to make turns. Wuthishuwong and Traechtler [11] have studied a $3 \cdot 3$ grid of intersections. Each intersection has an intersection manager that can decide the passing times for the vehicles at that intersection. They describe an algorithm that equally divides the traffic density over the different intersections. In this way, each intersection has approximately the same level of congestion. In Saiáns-Vázquez et al. [7], a $7 * 7$ Manhattan grid is studied. In their approach, a leader vehicle gets appointed at each intersection (until that vehicle leaves and a new leader gets appointed) who takes care of organizing the intersection. The leader will first allow several vehicles from the most congested lane to pass the intersection, as well as the first vehicles in other lanes as long as their trajectories do not interfere. The rest of the lanes will be served sequentially. The number of vehicles that is allowed to pass is proportional to the number of vehicles in each lane. Furthermore, lanes where platoons are coming up, get a higher priority. They also mention maneuvers to join or leave a platoon. The results show significant reductions in delay.

Several approaches to the congestion problem have been mentioned. We will now focus on Timmerman and Boon [10], who have taken the aforementioned polling models approach to organize the intersection. In this work, we will try to expand the ideas of Timmerman and Boon [10] to a network scenario. Their paper will be discussed in more detail in the following section.

2.1 Main contributions

This report is based on Timmerman and Boon [10], which in turn draws mostly from Miculescu and Karaman [6] and Tachet et al. [8]. The approach is based on polling models. They assume a control region in which a central controller sets all the access times for the approaching vehicles. It is assumed that the speed of the vehicle is controllable and that they enter the control region at the maximum allowed velocity. Furthermore, vehicles do not make turns. The main idea is that vehicles do not stop at the stop line right in front of the intersection, but to slow down earlier on so that they can accelerate again and pass the intersection at full speed. Because the vehicles pass the intersection at full speed, the intersection is used more efficiently. Next, they expand upon several policies, or platoon forming algorithms (PFA), regarding the rules to let the vehicles pass. They start off with the exhaustive PFA: this policy will continue to let vehicles from the same direction cross as long as they are within a certain amount of time from each other. Then they explain the

gated PFA, which is similar, but does not allow vehicles that entered the control region after the crossing of the first vehicle to join the platoon. Finally, the Batch algorithm from Tachet et al. [8] is mentioned. This policy is FCFS until a vehicle is delayed, at which point vehicles from that lane that arrive during the delay can pass the intersection together with the delayed vehicle. There is a maximum batch size to avoid giving access to only one lane. This process is then reiterated.

Next, they elaborate upon speed profiling algorithms, which are necessary to provide the vehicles with a safe trajectory that also ensures a punctual arrival at the intersection. One such algorithm is the so-called MotionSynthesize procedure from Miculescu and Karaman [6]. They then suggest to minimize the absolute value of the acceleration instead of the distance to the intersection in order to achieve a better experience for passengers and a lower energy consumption. They go on to show that both speed profiling algorithms can be written in closed form and that the closed form expressions are equivalent to the non-closed expressions in the sense that they minimize the distance to the intersection and the absolute value of the applied acceleration respectively.

Finally, they compare the different PFA's in terms of mean delay and fairness using polling systems theory. It is shown that the gated and batch PFA's behave similar if the intersection is not overcrowded. The exhaustive PFA is close to optimal when it comes to mean delay, but is significantly less fair.

An overview of the characteristics of all the mentioned papers is presented in Table 1.

	Polling systems based	Reservation based	Speed profiling	Platoon forming	Simulates a network
Bashiri, Jafarzadeh and Fleming [1]	✓			✓	
Carlino, Boyles and Stone [3]					✓
Dresner and Stone [4]		✓	✓		
Liu, Wang and Hoogendoorn [5]			✓	✓	✓
Miculescu and Karaman [6]	✓		✓	✓	
Saiáns-Vázquez et al.[7]	✓			✓	✓
Tachet et al. [8]	✓		✓	✓	
Tian, Zhao and Yi [9]				✓	
Timmerman and Boon [10]	✓		✓	✓	
Wuthishuwong and Traechtler [11]		✓	✓		✓
Wuthishuwong, Traechtler and Bruns [12]		✓	✓		
Yan, Dridi and Moudni [13]				✓	✓

Table 1: Literature overview

3 Model description

We will model a network of intersections through which autonomous vehicles navigate. First, we will introduce some notation. The modelled network exists of a grid of r rows and c columns. Hence, there are $r \cdot c$ intersections. Let I be the set of all intersections and denote an element $i \in I$ by $(r(i), c(i))$. Here, $r(i)$ and $c(i)$ are two functions which return the row and column of intersection i respectively. Therefore, $r(i) \in \{0, 1, \ldots, r-1\}$ and $c(i) \in \{0, 1, \ldots, c-1\}$. Next, denote a queue q by $q_{i,j}$ for $j \in \{0, 1, 2, 3\}$ and $i \in I$. This refers to the j^{th} queue at intersection i. The numbering of the queues is clockwise starting from the top lane. For $r, c = 2$, the network is visualized in Figure 1.

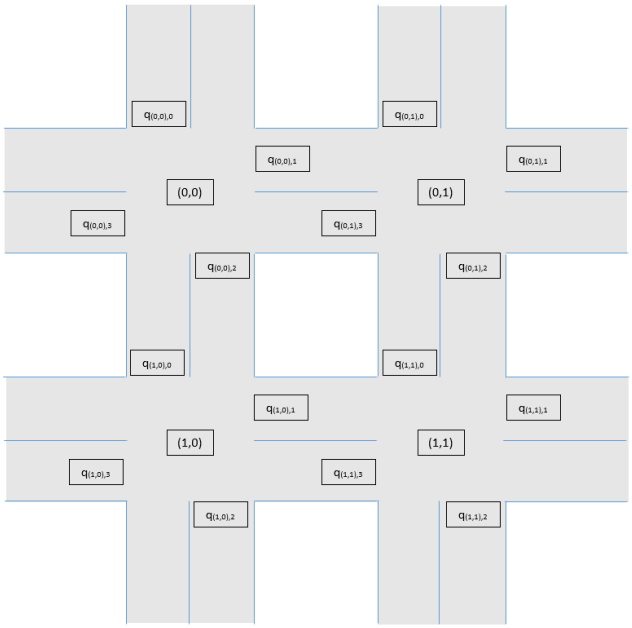


Figure 1: Example of a 2 by 2 network

Each intersection i is assumed to have a centralized controller which organizes the intersection and a sufficiently large control region such that every vehicle arriving at i has space to enter the control region. For simplicity's sake, we will only consider the case in which the control region is of equal size d for all queues at all intersections. Note that this means that the distance from arrival at the control region to the intersection is the same for all lanes. Furthermore, it is assumed that every vehicle in the control region will drive the exact speed as required by the controller. The controller also decides the moment at which a vehicle can pass the intersection. Once a vehicle gets access to the intersection, it has a service time of B seconds. During this time, no other vehicles can enter the intersection. In queueing theory, B is often modelled to come from a probability distribution, but in this model it will be deterministic. This makes sense as all vehicles traverse the intersection at full speed and thus occupy the intersection for the same amount of time. After B seconds have passed, the next vehicle can get access. If the next vehicle is to depart from another lane, an extra switch-over time S is implemented for safety reasons. The order of passing depends on the PFA that is applied. We will only consider networks in which all intersections use the same PFA. These PFA's will be discussed in more detail in Section 3.2.

At the queues on the borders of this grid, $q_{(0,m),0}, q_{(l,c-1),1}, q_{(r-1,m),2}, q_{(l,0),3}$ for $l \in \{0, ..., r-1\}, m \in \{0, ..., c-1\}$, vehicles arrive according to a Poisson process with intensity $\lambda_q \geq 0$. In our model, there are no other external arrivals in the network. As in Timmerman and Boon [10], the vehicles all enter at maximum velocity. As the distance d to an intersection is assumed to be the same for all vehicles at arrival at a control region, this means that the minimum time to the intersection is the same for all arriving vehicles. A vehicle is delayed if it is too close to its predecessor when it enters the control region, if it cannot continue driving

at full speed without following its predecessor too closely (because it is delayed), or if the intersection is occupied by vehicles from other queues. If a vehicle is delayed, the necessary deceleration is assumed to happen well before the ordinary stop line. Similarly, if a full stop is necessary, the vehicle stops at the point at which it is still able to accelerate to the maximum allowed velocity before entering the intersection. The earlier mentioned speed profiling algorithms can ensure this. In this way, all vehicles pass the intersection at maximum allowed velocity and thus use the intersection for the shortest amount of time. As mentioned, this is the service time B. As every vehicle uses the intersection optimally, the number of vehicles that can pass per time interval is maximal. Note that it is discretely assumed that every vehicle can reach the maximum allowed velocity.

When a vehicle gets access to the intersection, it can travel in any direction. Which direction it travels in, will be decided randomly. For this, a $(r \cdot c \cdot 4) \times (r \cdot c \cdot 4)$ matrix P is used. Then for a vehicle at queue $q_{i,j}$ we obtain:

$$q_{i,j} \to q_{k,l} \text{ w.p. } p_{k,l}^{i,j} \tag{1}$$

where $i, k \in I,\ j, l \in \{0, 1, 2, 3\}$ and $p_{k,l}^{i,j}$ is the element of P on row $r(i) \cdot c \cdot 4 + c(i) \cdot 4 + j$ and column $r(k) \cdot c \cdot 4 + c(k) \cdot 4 + l$. We will consider different routing options for the vehicles. These are explained in Section 5.3.2.

After having passed the first intersection, a vehicle drives unhindered for a while. Note that no delay is possible here since all vehicles are driving at maximum velocity and following each other neatly at at least the minimum allowed distance because of the PFA of the first intersection. Also, they are not yet in a new control region, so they do not yet experience delay caused by the next intersection (since the control region is assumed to be large enough). Hence, the time the vehicles drive before arriving at the next control region is constant. After that time, they arrive at the next control region and the controller once again decides their passing time according to the PFA and the corresponding trajectory. This process continues until a vehicle leaves the grid.

3.1 Polling model

Each intersection is based on a polling model. This is a model from queueing theory in which several types of users want access to a common resource which is only available to one type of user at a time. A polling model consists of N queues and a server which visits the queues in some order. A switch-over time is commonly incurred whenever the server switches to another queue. Whenever the server starts service at a queue, the service discipline of the polling model decides which customers will be served during that visit of the server. In our model, the PFA is the equivalent of the service discipline. A typical polling model can be seen in Figure 2.

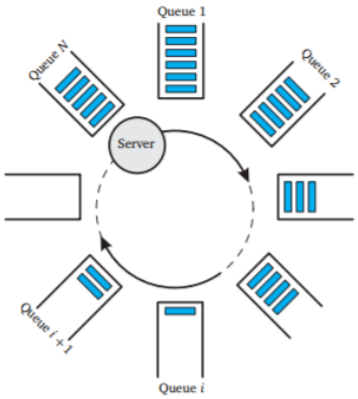


Figure 2: A typical polling system

We use the idea of a polling model to simulate the network. This makes knowledge of the exact velocity and location of a vehicle redundant. Instead a simpler discrete event simulation can be implemented. This simulation still provides most important performance measures. For the delay, the link between the trajectory of the discrete event simulation and the actual trajectory of the vehicle can be seen in Figure 3. The red line represents a vehicle (or customer) in the polling model whereas the black line represents a 'real' self-driving vehicle. Both vehicles have the same arrival time and hence the same earliest crossing time. As the figure demonstrates, the crossing time is the same in both cases as well. As the delay is the difference between the actual crossing time and the earliest crossing time, the delay is the same.

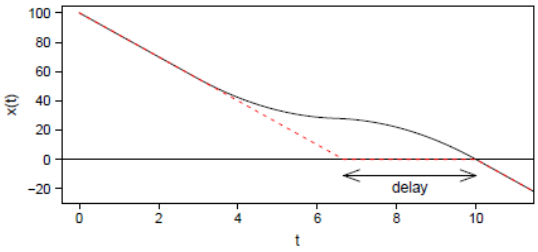


Figure 3: Link between the model trajectory and the true trajectory of a vehicle

3.2 Platoon forming algorithms

A few PFA's were already shortly mentioned in Chapter 2. Here, we will explain two of them in more detail. These PFA's find their origin in polling systems theory. It is these two PFA's that we will compare throughout this study.

Exhaustive algorithm

In this policy, vehicles from one lane get access to the intersection as long as they are within a certain amount of time of their predecessor. To make this rigorous, a vehicle is allowed to join the platoon if that vehicle can pass within s seconds of its predecessor. We will take $s = B$, the amount of time between two departures from the same lane. Note that in this case the actual passing time will be the passing time of the predecessor plus B, even though the vehicle might have been able to pass earlier. If the vehicle cannot pass within s seconds of its predecessor, it forms a new platoon. It also forms a new platoon if there are no other vehicles in the lane at arrival.

After the last vehicle of a platoon has passed, the next lane with delayed vehicles in clockwise order gets access to the intersection after clearance time S. If there are no lanes with delayed vehicles, the lane with the vehicle with the earliest possible crossing time a_v gets access to the intersection. Note that if the closest vehicle is from the same lane as the last vehicle that crossed the intersection, it does not have to wait for the clearance time S.

K-limited algorithm

This policy is similar to the exhaustive algorithm. The main difference is that there is a maximum platoon size k. This means that a vehicle that can get within B seconds of its predecessor is only allowed to join its platoon if the size of the platoon is smaller than k. The switching of lanes follows the same procedure. It should be noted that the platoon size is also reset in the case that the closest vehicle is on the same lane as the last crossing vehicle.

4 Simulation description

In this section, a detailed description of the discrete event simulation will be provided. All classes of the program will be explained. The simulation was written in Java.

We opted to simulate the network of intersections because theoretical analysis is not possible. Without an assumption on the arrival process, we have no knowledge over when and how many vehicles will arrive. This information is a necessity for any theoretical approach. We have made no assumption on the arrival process for any of the lanes in the network, except for those on the border. A simulation allows us to still approximate the performance measures sufficiently well and enables us to draw a conclusion on the performance of different PFA's in a network setting.

The idea of the simulation is to look at three different events: an internal arrival, an external arrival and a departure. These events will be saved in chronological order in a list called the Future Event Set. At an internal arrival event, a vehicle is added to the queue, where a queue is a list of vehicles. An external arrival does the same, but also schedules a new arrival according to the corresponding Poisson process of the lane. At a departure epoch, the next departure event is scheduled according to the applied PFA and (if applicable) a new internal arrival is scheduled according to the routing matrix. This continues for a set amount of time.

Vehicle class

In this class, we create the vehicles. Each `Vehicle` has 6 parameters: the arrival time in the system, the earliest possible crossing time, the actual crossing time and the row, column and lane of the current queue. The earliest crossing time and the actual crossing time can be used to compute the delay. All parameters can be changed. Finally, there is a method which sets all parameters to -1. This represents a vehicle leaving the system.

Queue class

This class keeps track of a queue of vehicles. Hence, a `Queue` is simply a list of `Vehicle` objects. Furthermore, a `Vehicle` can be added or removed, and a `Vehicle` at a specific location in the queue can be returned. The length of the `Queue` can also be returned.

Event class

An event is one of the 'actions' in the model. Each event has three parameters: a type, a time and a `Vehicle`. There are three possible types: an external arrival into the network, an arrival within the network and a departure meaning that a `Vehicle` is leaving its lane. The time is simply the moment at which the `Event` takes place and the `Vehicle` is the vehicle that undergoes the event.

FES class

`FES` is short for Future Event Set. As the name suggests, it is an time-ordered queue of upcoming events. Hence, it is an list of `Event` objects. An `Event` can be added in the right place and the next `Event` can be returned while immediately removing it from the `FES`.

Results class

This class keeps track of several values obtained throughout the simulation which ensure that we can calculate the desired results. We keep track of the sum of all the delays at each intersection and lane. Besides this,

we keep track of the number of vehicles at each intersection and lane. In this way, the mean delay for each intersection and lane can easily be calculated. Similarly, we can obtain the mean platoon size and cycle times. Finally, we keep track of the number of vehicles in each lane at a certain time.

Heatmap and Renderer classes

These classes provide a visual representation of the traffic dispersion in the network by creating a heatmap of the network based on the mean delay per lane.

Simulate class

This is the class where the simulation is written. Note that the description of the simulation is for the exhaustive PFA. Whenever the implementation of the k-limited PFA differs, this will be mentioned.

First, we introduce several methods to create the $(r \cdot c \cdot l) \times (r \cdot c \cdot l)$ routing matrices, where r is the number of rows, c the number of columns and l the number of lanes. Every row and column of such a matrix represents one of the queues of the network. The numbers in the matrix are the probabilities to arrive at the queue the column represents when departing from the queue the row represents. We create the straight routing matrix which lets all vehicles only drive straight ahead and a routing matrix which has a 0.5 probability of driving straight ahead, a 0.25 probability of turning left and a 0.25 probability of turning right. A third routing matrix is created, which has a 0.5 probability of turning left and 0.5 probability of turning right. As an example, Algorithm 1 illustrates how the straight routing matrix is obtained.

Algorithm 1: Creating the straight routing matrix

for *all lanes in the network* **do**
  if *lane number=0* $\wedge$ *row number*$\neq$*r-1* **then**
    straight routing matrix[row number $\cdot c \cdot l$+column number $\cdot l$+lane number][(row number+1) $\cdot\, c \cdot l$+column number$\cdot l$+lane number]=1;
  else if *lane number=1* $\wedge$ *column number*$\neq$*0* **then**
    straight routing matrix[row number $\cdot c \cdot l$+column number $\cdot l$+lane number][row number $\cdot c \cdot l$ + (column number+1) $\cdot\, l$+lane number]=1;
  else if *lane number=2* $\wedge$ *row number*$\neq$*0* **then**
    straight routing matrix[row number $\cdot c \cdot l$+column number $\cdot l$+lane number][(row number$-$1) $\cdot\, c \cdot l$+column number $\cdot l$+lane number]=1;
  else if *lane number=3* $\wedge$ *column number*$\neq$*c-1* **then**
    straight routing matrix[row number $\cdot c \cdot l$+column number $\cdot l$+lane number][row number $\cdot c \cdot l$ + (column number+1) $\cdot\, l$+lane number]=1

Next, a method is created which returns the row, column and lane number of the arrival destination of a vehicle based on the column number i of the entry in the routing matrix. It calculates the row, column and lane number as follows:

$$(r(i), c(i), l(i)) = (\lfloor \frac{i}{c \cdot l} \rfloor, \lfloor \frac{i \mod (c \cdot l)}{l} \rfloor, i \mod l) \tag{2}$$

Then the Results, FES and Queue objects are initiated and some necessary constants are declared. These constants are the time to reach an intersection after arriving at the control region (15.0 seconds), the time between the departure from an intersection and the arrival at the control region of the next intersection (10.0 seconds), the necessary time between two departures from the same lane (1.0 second), the service time (1.0 second) and the switch-over time (1.375 seconds). The next step is generating the first arrivals. The approach for this is outlined in Algorithm 2.

Algorithm 2: Generating the first arrivals

for *all lanes on the edge of the network* **do**
    create a new vehicle with arrival time according the corresponding λ_q;
    add an external arrival event for the vehicle above at its arrival time to the FES;

Then, a `while`-loop is started which compares the time of the last event with the total time the simulation is supposed to run. As long as the current time is lower than the total simulation time, the simulation continues. The `while`-loop has the following structure:

- take the next event `e` of the `FES`
- let `v=e.getVehicle` and `t=e.getTime`
- split the `while`-loop into three parts based on `e.getType`

The working of these three parts will now be expanded upon.

External arrival event

An external arrival event has to add the arriving vehicle to the corresponding queue and schedules a new external arrival on the same lane. If all lanes at the intersection are empty at arrival, the vehicle can drive unhindered, except for when it has to wait for the switch-over time. Note that this only happens if the minimal time from entering the control region to arriving at the intersection is smaller than the switch-over time. This is not the case for our chosen values, but for completeness sake, the departure event is scheduled at the maximum of its earliest crossing time and the moment when the switch-over time has passed. Algorithm 3 provides the corresponding pseudocode.

Algorithm 3: External arrival procedure

if *all lanes at the intersection are empty* **then**
    v.crossingTime=max(v.earliestCrossingTime, the switch-over time has passed);
    schedule a new departure event for v at its crossing time;
add v to the corresponding queue;
schedule a new arrival event at t+$exp(\lambda_q)$;

Internal arrival event

An internal arrival event is equivalent to an external arrival event, except for not scheduling a new arrival. This event is described in Algorithm 4.

Algorithm 4: Internal arrival procedure

if *all lanes at the intersection are empty* **then**
    v.crossingTime=max(v.earliestCrossingTime, the switch-over time has passed);
    schedule a new departure event for v at its crossing time;
add v to the corresponding queue;

Departure event

A departure event has to remove the vehicle from the corresponding queue and has to decide on which vehicle can pass the intersection next, according to the applied PFA and visiting order of the lanes. Besides this, it

must plan an internal arrival event for the currently departing vehicle if necessary. Note that Algorithm 5 is for the exhaustive PFA. The difference with the k-limited PFA is small: a departure from the same lane is only allowed if the current platoon size is smaller than k. k simply increases every time a vehicle joins the platoon (in the first `if`-statement) and is reset once a new platoon is allowed access to the intersection (in the first `else`-statement).

Algorithm 5: Departure procedure

```
remove v from its queue;
/* First schedule the next departure. Let w be the vehicle directly behind v on the same lane.  */
if the queue of v is non-empty ∧ w.earliestCrossingTime - t ≤ maximum allowed time between two vehicles
 in a platoon then
    w.crossingTime= max(w.earliestCrossingTime,t+minimum allowed following time);
    schedule a new departure event for w at its crossing time ;
    /* If no departure is scheduled here, another lane gets access to the intersection          */
else
    /* Make a new vehicle to keep track of the vehicle closest to the intersection              */
    construct a new Vehicle closestVehicle with closestVehicle.earliestCrossingTime=Integer.MAX_VALUE
     ;
    /* First check whether there is a delayed vehicle on the intersection                       */
    for all queues of the intersection traversing them in clockwise order starting from the lane next to that
     of v do
        if the queue is non-empty ∧ the first vehicle u in the queue is delayed then
            u.crossingTime=t+switch-over time S;
            schedule a new departure event for u at its crossing time;
            break;
        if the lane is non-empty ∧ the crossing time of the first vehicle u in the
         queue<closestVehicle.crossingTime then
            closestVehicle=u;
    /* If there are no delayed vehicles, the vehicle closest to the intersection gets access    */
    if no departure has been scheduled ∧ closestVehicle.earliestCrossingTime!=Integer.MAX_VALUE then
        schedule a new departure event for closestVehicle at its crossing time;
    /* If no departure has been scheduled at this point, the intersection is empty              */
    if no departure has been scheduled then
        keep track of when the switch-over time has passed for new arrivals;
/* Schedule the internal arrival for v                                                          */
p=random number between 0 and 1;
arrivalDestination=-1;
for i=0 to i= r · c · l do
    p = p-routingMatrix[v.row ·c · l+v.column·l+v.lane][i];
    if p < 0 then
        arrivalDestination=i;
        break;
if arrivalDestination>-1 then
    use the method where Equation 2 is implemented on arrivalDestination;
    set the row, column and lane of v to the values obtained above;
    v.earliestCrossingTime=t+time to reach next control region + time to reach intersection after entering
     the control region;
    schedule an internal arrival event for v at t + time to reach next control region;
else
    /* v leaves the network                                                                     */
    set all parameters of v to -1;
```

5 Performance analysis

In this section the results from the simulation will be discussed. First, an isolated intersection will be considered. In the sections thereafter, the size of the network will slowly be increased. In this way, we will draw conclusions on the effect the network size has on the performance of the exhaustive and k-limited PFA. All the results below have been obtained with a simulation time of at least 1,000,000 seconds, which is roughly 11 days. This is more than sufficient to obtain a decent approximation for the desired performance measures.

5.1 An isolated intersection

The case of an isolated intersection allows us to verify our simulation. Several different cases will be discussed, some of which can be analyzed theoretically.

5.1.1 M/G/1 queue

First, an isolated intersection i with arrivals at a single lane will be discussed. It is a single intersection, so $r(i) = c(i) = 0$. As there are no vehicles on other lanes, there are no switch-over times. This means that there is a single queue of vehicles and one server. Assume that each vehicle has a service time of $B = 1$ second. This situation can be modelled by an M/G/1 queue. This is a well-studied model from queueing theory. For a M/G/1 queue we know:

$$\mathbb{E}[W] = \frac{\rho \cdot \mathbb{E}[R]}{1 - \rho} \tag{3}$$

where W is the delay, R the residual service time and $\rho = \sum_{j=0}^{4} \lambda_{(0,0),j} \mathbb{E}[B]$, the fraction of time the server is busy. Since vehicles solely arrive at lane 3, we only have to take $\lambda_{(0,0),3}$ into account. Hence, $\rho = \lambda_{(0,0),3}\mathbb{E}[B]$. Now let $\lambda_{(0,0),3} = \frac{1}{2}$. As $B = 1$, we know $\mathbb{E}[R] = \frac{1}{2}$ and then $\rho = \frac{1}{2}$. We finally obtain $\mathbb{E}[W] = \frac{1}{2}$. In our simulation, we used the exhaustive PFA and the same values for B and $\lambda_{(0,0),3}$. With a simulation time of 10^8 seconds, we obtain a mean delay of 0.500. The corresponding confidence interval is (0.4998,0.5002). This more than sufficiently approaches the theoretical solution.

Another way to validate the simulation is to look at the number of vehicles in the queue. The arrival epochs can be arbitrarily close to each other, but there must be at least 1 second between two departure epochs, as that is the service time of a vehicle. In Figure 4, it can be seen that this is indeed the case. For example, at approximately $t = 15$ three vehicles enter the queue shortly after another (with an interarrival time of less than 1 second). However, 15 seconds later at approximately $t = 30$, there are 1 second intervals between the departures of the vehicles. We also observe that there are no departures during the first 15 seconds, even though there are vehicles in the queue. This is correct, as the vehicles only arrive at the intersection 15 seconds after having entered the control region. Hence, the figure behaves as expected. As the simulation yields the correct outcomes, we can conclude that our simulation works correctly for arrivals at a single lane.

5.1.2 Number of vehicles

We will now look at the situation where vehicles arrive at all four queues of the intersection. Using plots of the queue length of all four lanes (for both the exhaustive and k-limited PFA), we will show that the simulation still functions properly.

First, consider the case of a intersection where all the lanes are equally busy. Let $\lambda_{(0,0),j} = \frac{10}{41}$ for $j \in \{0, 1, 2, 3\}$. This means that $\rho = \frac{40}{41} < 1$, so the system is stable for the exhaustive PFA. Note that it is not stable for the k-limited(10). This will be explained in Section 5.1.3. In Figure 5, the number of vehicles is plotted for each lane for both the exhaustive and k-limited PFA.

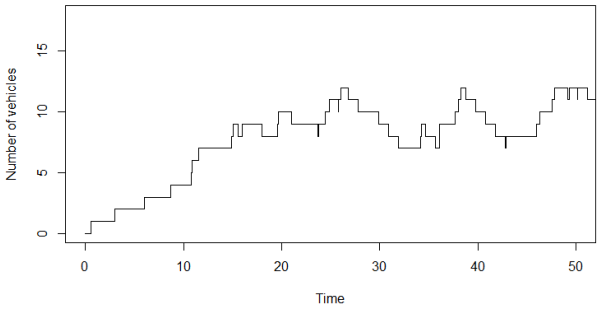
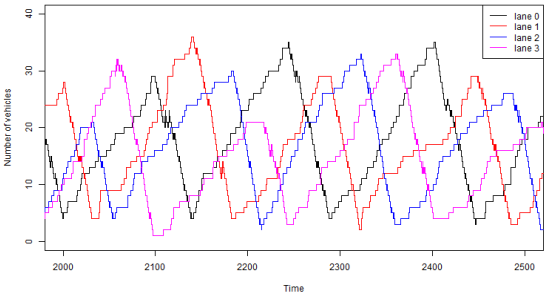


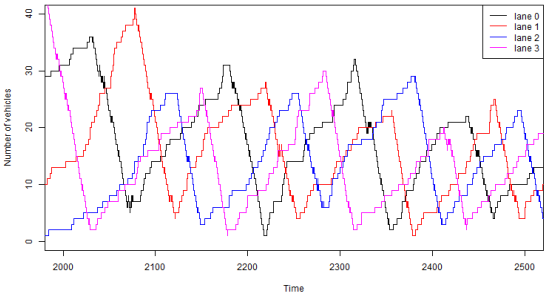
Figure 4: Number of vehicles in the queue for one simulation run

As can be seen in Figure 5a, the queues get access to the intersection one by one, constantly in the same order. This makes sense, as there are (nearly) always delayed vehicles in a lane once that lanes gets access to the intersection, because $\lambda_{(0,0),j}$ is rather large. The triangular shape in the figure are logical as well. The number of vehicles in the queue keeps increasing according to the Poisson process until the server (the intersection) visits the queue. At this point, there is a large group of delayed vehicles which leave as a platoon, one after another at 1 second intervals. The number of vehicles mostly does not reach 0 as there are vehicles arriving while the platoon is departing which can not join the platoon. For Figure 5b, we see that the same characteristics hold. This is to be expected, as the k-limited PFA behaves the same as the exhaustive PFA for $k \to \infty$. In this case, 10000 was chosen for k. As the number of vehicles is not close to this number, the k-limited PFA should have the same effect as the exhaustive PFA. In Figure 5c, the result for k-limited with $k = 10$ is plotted. Once again, all lanes depart in order. The triangular shape can also be seen, although it is less obvious. This is because only 10 vehicles are allowed in a platoon. One can check that indeed no more than 10 vehicles leave in a single visit of the server to a queue.

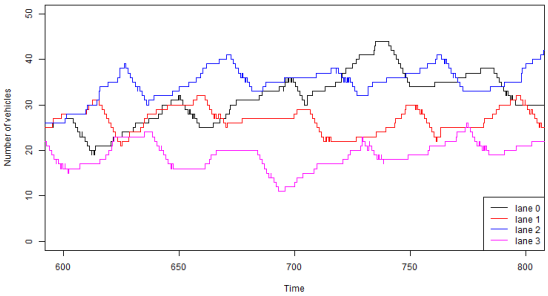
Stability is an important difference between the exhaustive and k-limited PFA. In a stable system, the mean number of vehicles entering the system per time unit is smaller than the mean number of vehicles leaving the system per time unit. If this is not the case, the server cannot handle the number of arriving vehicles and hence the number of waiting vehicles will grow indefinitely. Hence, the waiting times will rapidly increase. For an intersection with the exhaustive PFA, it is either stable in all lanes if $\rho < 1$ or unstable in all lanes if $\rho \geq 1$. If the arrivals are symmetric and $\rho < 1$, the number of vehicles in a platoon is roughly the same for all lanes. However, also when the arrivals are asymmetric, the intersection is still stable as long as $\rho < 1$. The difference is that for the lane j with the largest $\lambda_{(0,0),j}$ the platoon sizes are on average larger than for the other lanes. For a symmetric intersection with the k-limited PFA, it also holds that either the lanes are all stable or all unstable. However, this does not hold for an asymmetric intersection. Note that each cycle, a maximum of k vehicles can leave the busiest lane. After the departure of these at most k vehicles, the other lanes get access to the intersection. If, on average, more then k vehicles enter the queue in the time the other lanes have access plus the switch-over times, the busiest lane is unstable. This is visualized in Figure 6. The Figures 6a, 6b and 6c give an overview of the situation as it was for Figure 5. The arrivals are symmetric and thus the queue sizes are similar. To prevent the figures of becoming too unclear, only one lane is plotted for the exhaustive and k-limited(10000) PFA's. The systems are clearly stable. For k-limited(10), all lanes are plotted. The number of vehicles in the queues continuously increases and thus this situation is unstable. The Figures 6d and 6e show the situation for an asymmetric intersection with $\lambda_{(0,0),j} = \frac{1}{8}$ for $j \in \{0, 1, 2\}$, $\lambda_{(0,0),3} = \frac{1}{2}$ and thus $\rho = \frac{7}{8} < 1$. For the exhaustive PFA, lane 3 has larger platoon sizes, but it is still stable as a large number of vehicles is able to depart every cycle. For k-limited(10), this is not the case. This is why the number of vehicles continuously increases for lane 3.



(a) Exhaustive



(b) k-limited(10000)



(c) k-limited(10)

Figure 5: Number of vehicles in the queues

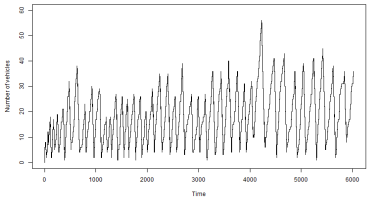
5.1.3 Comparison of k-limited and exhaustive PFA

As mentioned before, the k-limited PFA behaves the same as the exhaustive PFA for $k \to \infty$. Therefore, the mean delay should be the same for both PFA's for all $\rho < 1$ for k large enough. In Figure 7a, the mean delay has been plotted against ρ. This figure (based on a symmetric intersection) indeed confirms that the mean delay is equal.

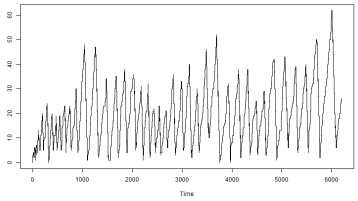
The stability of an intersection has shortly been discussed. However, a question that remains to be answered is when an intersection using the k-limited PFA becomes unstable. It is especially interesting to know how the value of k influences the stability. For the system to remain stable, the number of vehicles arriving at each lane has to be, on average, less than k per cycle. A cycle C is the period in which all lanes get access to the intersection exactly once. As the arrival rate for queue q is λ_q, we know $\lambda_q \mathbb{E}[C] < k$ has to hold. For $\mathbb{E}[C]$, the following holds: $\mathbb{E}[C] = \frac{\mathbb{E}[S_{cycle}]}{1-\rho}$. This equation for $\mathbb{E}[C]$ is derived under the assumption that empty queues are also visited by the server. This is not the case in our simulation, but as we are looking at the stability and thus consider high values of ρ, it is safe to assume that queues are never empty when the server arrives. Additionally, we know that $\rho = \sum_{i=0}^{N-1} \lambda_i \mathbb{E}[B_i]$ and $\mathbb{E}[S_{cycle}] = \sum_{i=0}^{N-1} \mathbb{E}[S_i]$ as the total switch-over time in a cycle is the sum of all switch-over times in a cycle. Substituting this yields Equation (4).

$$\lambda_q \frac{\mathbb{E}[S_{cycle}]}{1-\rho} = \lambda_q \frac{\sum_{i=0}^{N-1} \mathbb{E}[S_i]}{1 - \sum_{i=0}^{N-1} \lambda_i \mathbb{E}[B_i]} < k \tag{4}$$

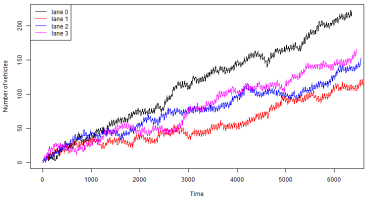
For symmetric arrivals, λ_i is the same for all i. Furthermore, $\mathbb{E}[B_i] = B \; \forall i$ as B is equal and deterministic for all lanes, so $\sum_{i=0}^{N-1} \lambda_i \mathbb{E}[B_i] = N\lambda B$. Similarly, $\sum_{i=0}^{N-1} \mathbb{E}[S_i] = NS$ as the switch-over time is also deterministic and equal for all lanes. As we have four queues, Equation (4) can be rewritten to:



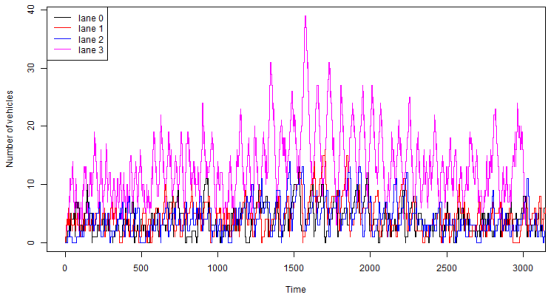
(a) Exhaustive symmetric



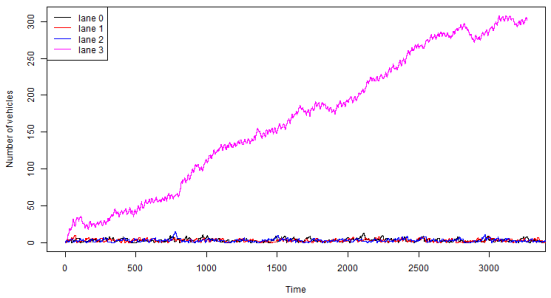
(b) k-limited(10000) symmetric



(c) k-limited(10) symmetric



(d) Exhaustive asymmetric

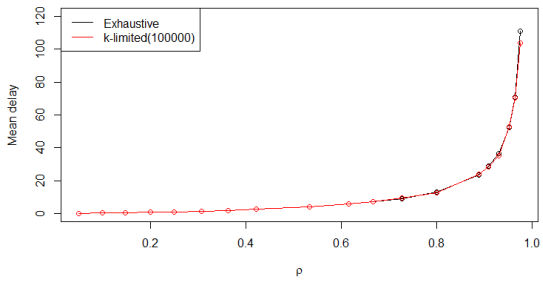


(e) k-limited(10) asymmetric

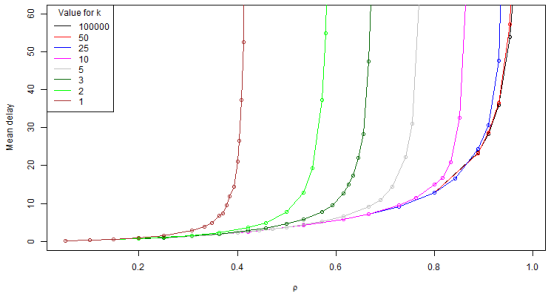
Figure 6: Overview of the number of vehicles in the queues

$$\lambda \frac{NS}{1 - N\lambda B} = \lambda \frac{4 \cdot S}{1 - 4 \cdot \lambda B} < k \tag{5}$$

Figure 7b gives a visualization of the influence of k. One can check that the system destabilizes at the theoretical values for λ that can be obtained from Equation (5) with $S = 1.375$ and $B = 1$. Another important observation to be made from this figure is that, for an isolated intersection, the exhaustive PFA is superior in terms of mean delay.



(a) Exhaustive and k-limited (100000)



(b) k-limited for different k

Figure 7: The relation between mean delay and ρ

5.1.4 Pseudo-conservation law

Boxma and Groenendijk [2] have derived the pseudo-conservation law (PCL) for the mean waiting times of polling systems. The PCL states:

$$\sum_{i=1}^{N} \rho_i \mathbb{E}[W_i] = \rho \frac{\sum_{i=1}^{N} \rho_i \mathbb{E}[B_{i,res}]}{1-\rho} + \rho \mathbb{E}[S_{cycle,res}] + \frac{\mathbb{E}[S_{cycle}]}{2(1-\rho)}(\rho^2 - \sum_{i=1}^{N} \rho_i^2) + \sum_{i=1}^{N} \mathbb{E}[Z_{ii}] \tag{6}$$

In this equation, W_i denotes the mean delay, B the service time and S_{cycle} the switch-over time in each cycle. The subscript res is short for residual. Z_{ii} denotes the amount of work left behind at a queue immediately after the visit of the server. This is entirely dependent on the service discipline. As the exhaustive PFA serves a queue until it is empty, the $\mathbb{E}[Z_{ii}] = 0$. The pseudo-conservation law can be used to validate our simulation by considering a symmetric, exhaustive system. In the case of a symmetric system, the mean delay should be equal at all queues. Furthermore, the service time ($B = 1$) and switch-over time per cycle ($S_{cycle} = 4 \cdot 1.375$) are deterministic in our model. For any deterministic variable X, the following holds:

$$\mathbb{E}[X_{res}] = \frac{\mathbb{E}[X^2]}{2\mathbb{E}[X]} = \frac{X^2}{2X} = \frac{1}{2}X \tag{7}$$

Hence, in our model, Equation (6) can be rewritten to:

$$\mathbb{E}[W] = \frac{\rho \cdot \frac{1}{2}B}{1-\rho} + \frac{1}{2}S + \frac{S}{2\rho(1-\rho)}(\rho^2 - \sum_{i=1}^{N} \rho_i^2) \tag{8}$$

With the values as described above, a single, symmetric intersection with the exhaustive discipline was simulated for several values of ρ. The resulting mean delays have been plotted in Figure 8 together with the theoretical results obtained from Equation (8). For high values of ρ, the results are indistinguishable as is desirable. However, for low values of ρ, the simulated mean delay is constantly lower than the theoretical mean delay. This can be explained by one of the assumptions used to derive the PCL. It was assumed that the server always visits the queues in a fixed order, even if a queue is empty. This is irrational in traffic management and hence this was not implemented in the simulation. To clarify this, suppose ρ is almost 0. Then the PCL states that the mean delay is $\frac{1}{2}S_{cycle}$ as it is expected that the customer has to wait for half of the switch-over times before the server reaches its queue. In the simulation, this is not the case as the server does not visit empty queues and thus immediately visits the queue of the arriving vehicle. Therefore, the mean delay goes to 0.

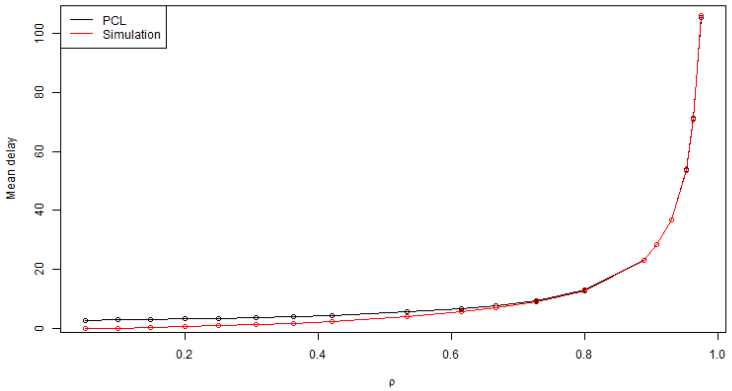


Figure 8: Comparing PCL to simulation

5.2 Two connected intersections

For a single intersection the simulation functions correctly. In this section, a network existing of two connected intersections will be discussed. It will be verified that the vehicles indeed arrive at the next intersection at the correct time. Furthermore, a comparison between the performance of the k-limited and the exhaustive PFA will be provided.

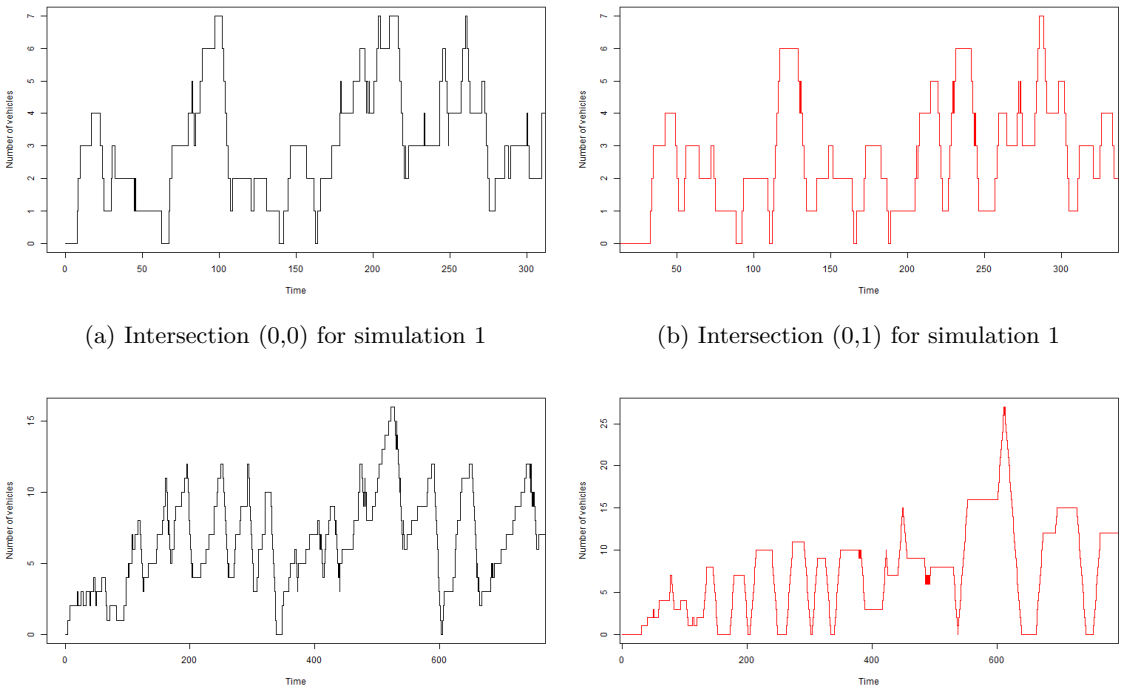
5.2.1 Number and trajectory of the vehicles

To verify that vehicles navigate through the network as they are supposed to, two simulations were done with the straight routing matrix and the exhaustive PFA. For the first simulation, the arrival rate (for all lanes with external arrivals) was $\lambda = 1/8$ and for the second $\lambda = 10/45$. The results of the amount of vehicles in lane 3 can be seen in Figure 9. It should be noted that x-axis of Figure 9b and Figure 9d start and end 25 seconds later than that of Figure 9a and Figure 9c. This is precisely the amount of time between arrival at the first intersection and arrival at the second intersection without experiencing delay. As no vehicles turn at the intersection, all the vehicles arriving at lane 3 of intersection (0,0) should arrive at lane 3 of intersection (0,1). For the first simulation, the occupation rate of the intersections is rather low ($\rho = 0.5$) and thus it is uncommon to have large delays. Hence, one would expect the shape of the two plots to be approximately the same, as vehicles can often traverse the first intersecting without delay. Even though the two figures are different, the overall shape is indeed similar. The arrival times at the second intersection also have to be correct. The fact that the overall shape is the same already suggests that this is the case. Closer examination of, for example, the first three arrivals in the system reaffirms this. The simulation seems to function properly. The differences between the figures of the first simulation can be explained by the obligatory departure intervals and the possible delay at both intersections. For the second simulation, the occupation rate is significantly larger ($\rho = 8/9$). This means that vehicles are delayed more frequently and hence one would expect the two plots to differ more. Figures 9c and 9d indeed show fewer similarities. Whereas in the first simulation the two plots had quite some of the same details, this is not the case in the second simulation. This has to do with the fact that larger platoons are formed because of the higher delays. Since mostly platoons arrive at the second intersection on lane 3, the 'detail' of single arrivals is lost. Furthermore, note that the platoons departing from intersection (0,0) arrive at intersection (0,1) on the expected time.

For the same simulation, the trajectory of the vehicles has been visualized in Figure 10. In this plot, it is assumed that the vehicles drive at a constant (maximum allowed) velocity when entering and can come to a standstill instantly. In the same way, they can instantly accelerate to the original velocity again. This is not realistic, but despite these assumptions, the resulting figure gives us more insight in the network. From the figure, it is clear that the vehicles continue to drive at maximum velocity until they are too close to their predecessor at standstill or until they are delayed because the intersection is occupied. After departing at the first intersection, they drive for a certain distance before arriving at the control region of the next intersection. As the vehicles are already at least a second apart from each other, they are only delayed if the intersection is occupied. For the exhaustive PFA in combination with straight routing, the following holds: if a platoon is delayed, the delay is equal for all vehicles (from the original platoon) as they both arrive and depart one second after one another and are always in the same platoon. For k-limited, this is not true as it is possible for a platoon to merge with a platoon that is waiting at intersection (0,1) (see $t \approx 360$). Hence, k-limited could split up the original platoon and thus cause a difference in delay.

5.2.2 Symmetric and asymmetric systems

In a symmetric system where none of the vehicles turn, both intersections have to handle the same number of vehicles. The lanes with external arrivals all have the same arrival rate. Hence, the lanes connecting the two intersections behave similar. This suggests that the mean delay for both intersection has to be equal. In Figure 11a, the mean delay has been plotted against ρ for both intersections and the mean delay is indeed equal for both intersections. The same approach was taken for a (partially) asymmetric system, for which



(a) Intersection (0,0) for simulation 1

(b) Intersection (0,1) for simulation 1

(c) Intersection (0,0) for simulation 2

(d) Intersection (0,1) for simulation 2

Figure 9: Number of vehicles in lane 3

the result can be found in Figure 11b. In this simulation, $\lambda_{(0,0),3}$ is twice as small as that of any other exterior lane. Note that the vehicles from $q_{(0,0),3}$ also have to traverse intersection $(0, 1)$. Therefore, the two intersections still have to handle the same number of vehicles. However, the figure reveals a slight difference in mean delay between the two intersections. The origin of this difference lies in the PFA of the first intersection that the vehicles traverse. As ρ increases, it increasingly often happens that two vehicles enter lane 3 within a 1 second interval from each. These vehicles entering the network within a second of their predecessor are delayed by at least:

1-(arrival time of predecessor - own arrival time)

On top of this minimum delay, they are delayed if other lanes have access to the intersection. As mentioned earlier, this delay is equal for all vehicles in a platoon for the exhaustive PFA in combination with straight routing. Independent of this delay, it is always true that the interdeparture time of vehicles is at least 1. Thus, for intersection (0,1), vehicles never enter lane 3 within 1 second from each other. This means that the delay caused by the minimum following distance never occurs at intersection (0,1) (for lane 3). For high values of ρ, this means that the mean delay of intersection (0,1) is slightly lower.

5.2.3 Comparison of k-limited and exhaustive PFA

To compare the two different PFA's, the mean delay per intersection will be considered. For intersection (0,0), the mean delay has been plotted against ρ for different values for k in Figure 12. Note that $k = \infty$ is equivalent to the exhaustive PFA. For small values of k, the difference in delay from the exhaustive PFA is large. The system quickly becomes unstable. For all traffic densities, the k-limited PFA performs worse or at best the same as the exhaustive PFA. However, the mean delay for k-limited rather quickly approaches that of the exhaustive PFA for increasing k. For $k = 25$, the mean delays are extremely close to each other and for $k = 50$, the difference has nearly vanished. However, when only taking mean delay into account, the exhaustive PFA still remains the better algorithm.

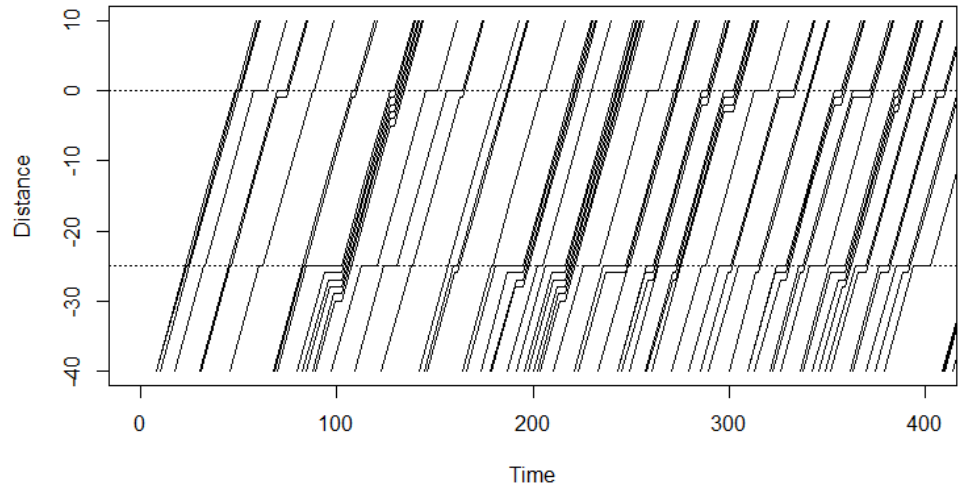
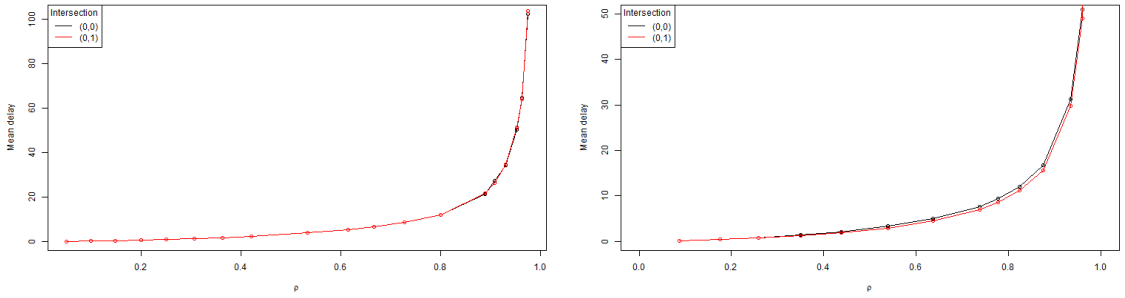


Figure 10: Trajectory of vehicles



(a) Symmetric

(b) Asymmetric

Figure 11: Mean delay for both intersections

5.3 A tandem network

After having discussed a system of two connected intersections, the next step is to examine a tandem network. This is a $1 \cdot c$ network. The total mean delay of the network will be compared for several values of c. The influence of the chosen routing matrix will also be discussed.

5.3.1 Trajectory of the vehicles

First, the trajectory of the vehicles on the lane connecting all the intersections will be discussed. The path of the vehicles entering on lane 3 of intersection (0,0) have been plotted in Figure 13. Note that the intersections are at 15, 40, 65, 90 and 115. These times follow from the time between entering a control region and arriving at the intersection (15 seconds) and the time until reaching the next control region (10 seconds). The data was obtained from an symmetric 1·5 network with $\rho = \frac{4}{5}$ with straight routing and the exhaustive PFA. The simulation seems to function properly as there are no signs that the rules of the exhaustive PFA have been violated. At the first intersection, every vehicle joins a platoon or forms its own. For the exhaustive PFA in combination with straight routing, it holds that these platoons never split up again. These platoons then

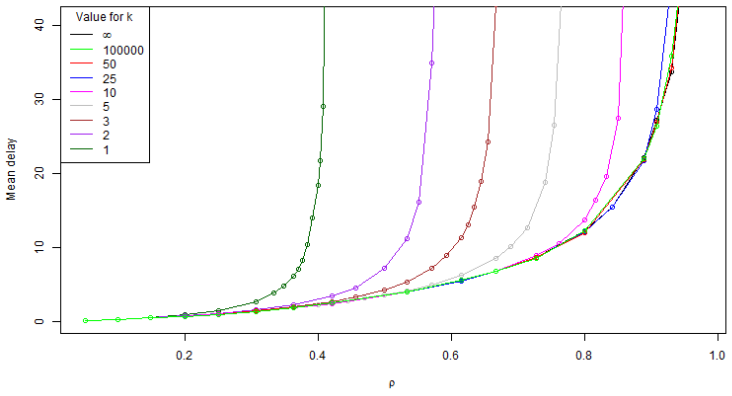


Figure 12: Comparing exhaustive and k-limited PFA

start to traverse all the other intersections with the possibility of merging with another platoon. This can result in rather large platoons (see e.g. $t \approx 380$). For even larger tandem networks, the platoon size would only increase. This is not entirely realistic, as normally vehicles do turn, resulting in gaps in the platoons. These gaps could cause the platoon to be split up again. Hence, Figure 13 might not give an entirely accurate overview, but it does validate the simulation to a certain extent and gives insight in the trajectory of the vehicles.

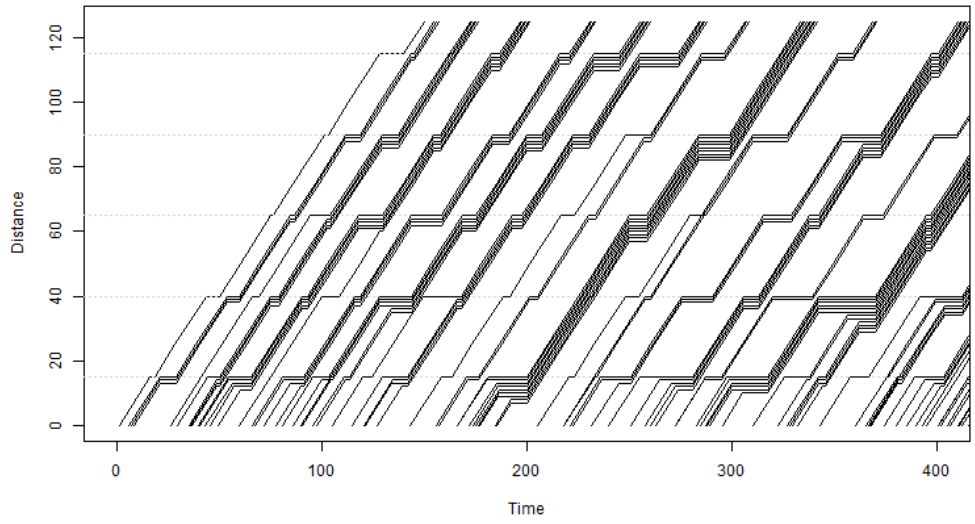


Figure 13: Trajectory of vehicles

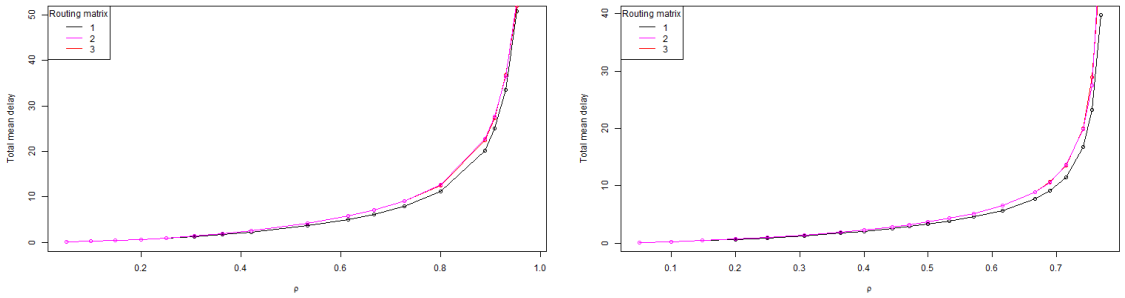
5.3.2 Routing influence

Up until now, we only considered the case in which all vehicles go straight ahead. It is clear that this is not a realistic situation. Hence, it is important to compare straight routing with routing rules that do allow turns, as different routing rules might heavily influence the mean delay, which is our main performance measure. To fairly compare routing matrices, the congestion level of each intersection must remain the same. If not, it is

obvious that the mean delay will increase or decrease depending on the value of ρ. This makes it impossible to find the actual effect of the routing rules. Furthermore, note that it is assumed that taking a turn does not take additional time as the service time is deterministic. The following three cases will be compared:

1. Straight routing
2. 50% straight, 25% left and 25% right
3. 50% left and 50% right

In Figure 14, the mean delay has been plotted against ρ for all three of the cases. This has been done for both the exhaustive and k-limited (5) PFA in a symmetric 1 by 5 network. Note that there is difference in scale between the two subfigures. We see that for both PFA's, straight routing results in the lowest mean delay, especially for high values of ρ. The other two routing options behave almost identically independent of the PFA. The superiority of the straight routing might be explained by the fact that it ensures that the largest allowed amount of vehicles follow each other at the minimum allowed distance. In other words, no gaps occur in the platoons. Therefore the entire platoon (or a maximum of k vehicles) can depart in the same cycle. If a gap does occur, which is possible for the routing option 2 and 3, some of the vehicles of the platoon might have to wait for the next cycle. This would result in a higher delay. Even though the routing has an influence on the mean delay, the differences are relatively small.



(a) Exhaustive

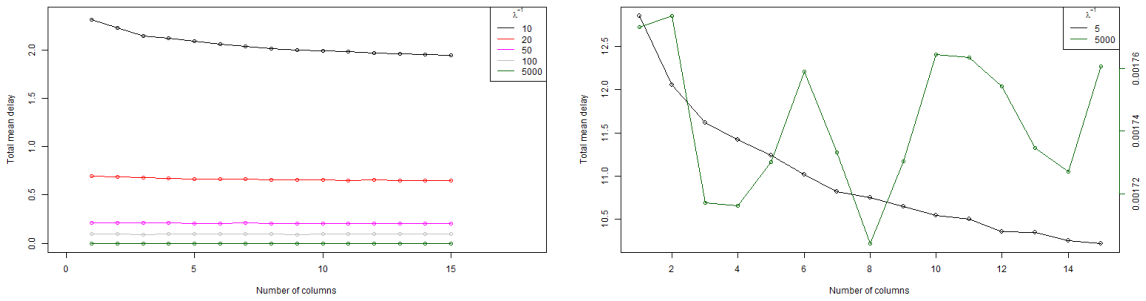
(b) k-limited (5)

Figure 14: Mean delay for different routing options

5.3.3 Influence of the size of the tandem network

Another factor that might influence the mean delay is the size of the tandem network. To study this, we only consider routing matrix 1. Note that only the two intersections at the ends of the tandem network have three lanes with Poisson arrivals. All the intersections in between only have two lanes with Poisson arrivals. Thus, as the size of the tandem network increases, the influence of the external arrivals on the mean delay slowly decreases. For the exhaustive and k-limited PFA, the total mean delay has been plotted against the amount of intersections in a symmetric network for several λ in Figure 15a and 15c respectively. For low values of λ, and thus ρ, the size of the network does not seem to influence the mean delay. For extremely low values of ρ, vehicles rarely enter the control region within a second from each other and the probability of an intersection being occupied are low. Hence, vehicles are generally not delayed at the first intersection, which means that the interarrival times at lane 1 or 3 of the next intersection are almost equal to the intervals of the first intersection. This then holds again for the next intersection. Therefore, it makes sense that there is no significant trend for increasing tandem network size. However, as ρ increases, the difference between the arrival processes of the first intersection and the next increases. For $\lambda^{-1} = 20$, there seem to be a slight decrease in mean delay when increasing the network size. For $\lambda^{-1} = 10$, the same effect is clearly visible. In Figure 15b, the total mean delay has been plotted for $\lambda^{-1} = 6$ and $\lambda^{-1} = 5000$ for the exhaustive PFA. For 15d, the same been done for the k-limited PFA for $\lambda^{-1} = 5$ and $\lambda^{-1} = 5000$. Note that the y-axis is different for both lines. The seemingly large fluctuations for $\lambda^{-1} = 5000$ are actually not that large when

looking at the y-axis on the right side of the figure. The differences from the true mean are simply caused by simulation. The figures indeed verify that for low values of ρ, there is no trend when increasing the network size, as opposed to high values of ρ, for which the trend is clear. These two values for λ are both rather extreme. For values in between, the trend gradually arises. This suggests that for both PFA's arrivals in platoons are preferable and that the influence of these PFA's is even more beneficial than is to be expected based on simulations of a single intersection with Poisson arrivals.



(a) Overview for exhaustive

(b) Detailed shape for exhaustive

(c) Overview for k-limited (5)

(d) Detailed shape for k-limited(5)

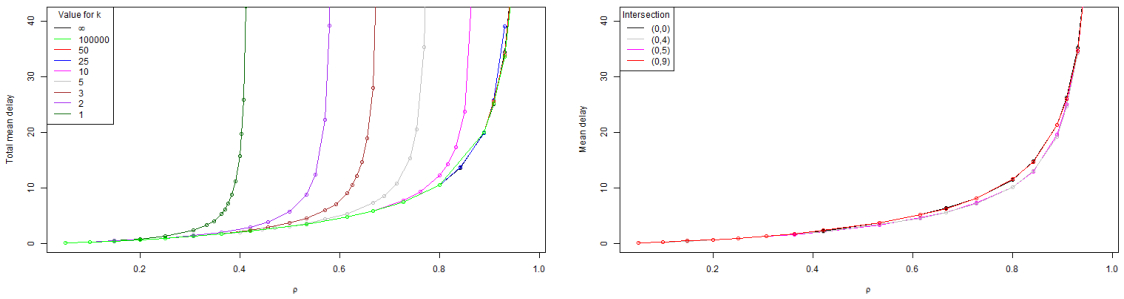
Figure 15: The influence of number of columns on mean delay

5.3.4 Comparison of k-limited and exhaustive PFA

For the tandem network, a conclusion on which PFA is to be preferred, has yet to be taken. In Figure 16a, the total mean delay of a $1 \cdot 10$ network has been plotted against ρ. Based on this figure, it is clear that the exhaustive PFA once again outperforms the k-limited PFA. It should be noted that the mean delay per intersection varies slightly. For straight routing, the intersections in the middle outperform those on the outskirts. This is visualized in Figure 16b. It seems like the mean delay is lower when the external arrivals have a smaller influence. This effect is similar for both PFA's.

5.4 A $n \cdot n$ network

We will now study a $n \cdot n$ network for several values of n. We will verify whether the patterns found in the tandem network can be extrapolated to this scenario. The influence of the size and routing will be discussed and the k-limited ($k = 25$) and exhaustive PFA's will be compared. This value for k was chosen such that the system would remain stable up to $\rho = 0.9$ according to Equation (5). For smaller values of k, the system will simply destabilize faster without performing better for smaller values of ρ, as we have seen in the earlier comparisons between the two PFA's. We have opted to not consider smaller values of k, as the running time of the simulation is rather long.



(a) Exhaustive and k-limited

(b) Difference among intersections

Figure 16: Mean delay for different ρ

As the number of intersections in a $n{\cdot}n$ network increases rapidly with n, looking at data of a single intersection is no longer sensible. Therefore, heatmaps will be used in this section. These indicate the relative mean delay per lane, providing us with a clear overview of the traffic density in the network. In Figure 19a, an example of a heatmap with labels of the lanes and intersections can be found. Note that the size of the intersections in these heatmaps remains constant independent of the chosen n. Therefore, the roads shorten if the size of the network increases. Another consequence of the increase in the number of intersections is that it takes longer for the system to be in steady state. This means that long runs of the simulation are necessary to approximate the true values sufficiently well. Therefore, we decided to use fewer data points in some of the figures.

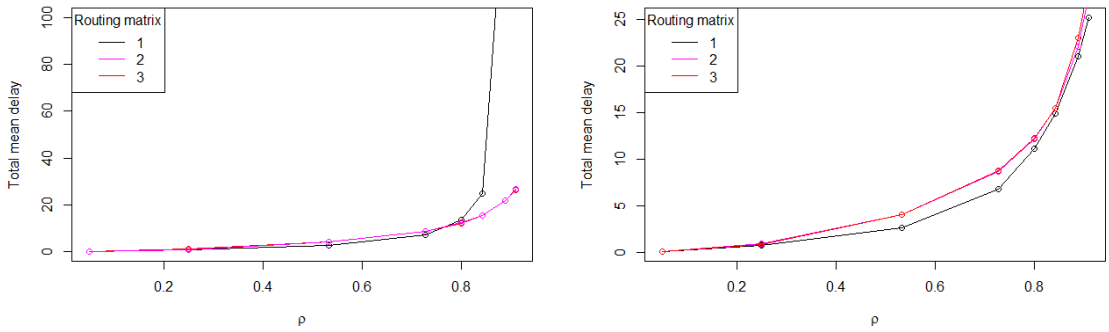
5.4.1 Routing influence

First, the effect of the routing matrix will be discussed. The same routing matrices will be used as in Section 5.3.2. For both the exhaustive and k-limited PFA, a symmetrical $15 \cdot 15$ network has been simulated for several arrival rates. The total mean delay of the network has been plotted in Figure 17. Once again, the differences between routing matrices 2 and 3 are minimal. For both PFA's, they perform somewhat worse than routing matrix 1 for low values of ρ. For higher values of ρ, this difference decreases in the case of the k-limited PFA. However, for the exhaustive PFA, routing matrices 2 and 3 actually outperform routing matrix 1 for high values of ρ. Routing matrix 1 behaves as if the system is unstable, even though the $\rho < 1$ for each intersection in the network. This seems to be an odd result, as the mean total amount of work arriving at each intersection per time unit remains constant. The cause of this surge in mean delay might lie in the fact that, as mentioned before, the platoons never split up with routing matrix 1 in combination with the exhaustive PFA, resulting in ever increasing platoon sizes. This will be discussed in more detail in Section 5.4.2.

5.4.2 Influence of the size of the network

The influence of the size of the $n \cdot n$ network also has to be studied. A symmetrical $n \cdot n$ network has been simulated with $\lambda = \frac{10}{45}$ for $n \in \{1, 2, \ldots, 15\}$. For both PFA's and all three routing matrices, the total mean delay has been plotted against the n in Figure 18. Note that $k = 25$ for the k-limited PFA.

In both subfigures, the results from the routing matrix 1 stand out. Before discussing these results, we will shortly go over the results from the remaining routing matrices. The mean delay for routing matrices 2 and 3 is similar. We observe that an increase in the network size results in a decrease of the mean delay. The decrease in delay is likely caused by the fact that the intervehicle time is always sufficient after passing the first intersection. As the intersection increases in size, the effect of this cause of delay (from which only external arrivals suffer) declines. This explains why the mean delay stabilizes. For both routing matrices,



(a) Exhaustive PFA

(b) k-limited ($k = 25$) PFA

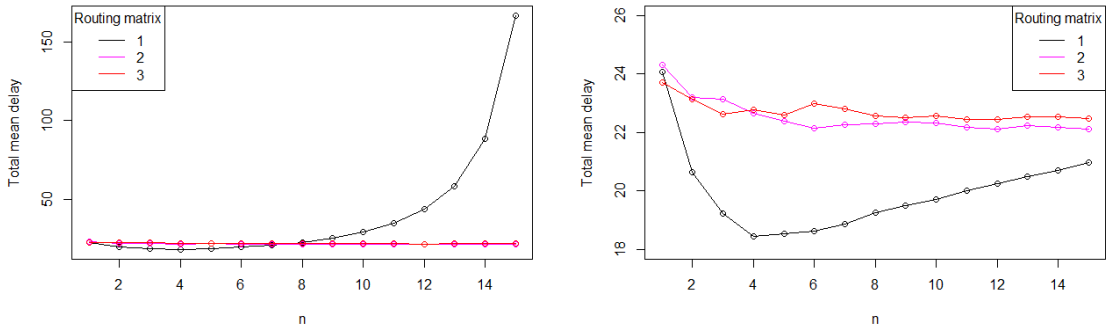
Figure 17: Total mean delay for different ρ

we found that the k-limited PFA performs slightly worse than the exhaustive PFA. The difference in mean delay between the two PFA's is roughly 0.5 seconds.

Whereas the total mean delays for random routing matrix 2 and 3 are close to each other, routing matrix 1 does not seem to follow a similar trend whatsoever. In Figure 18a, the total mean delay initially decreases, but destabilizes as n increases. As was mentioned in the previous section, we expect that this has to do with the ever increasing size of the platoons. To give an indication: just before a platoon leaves the network, the average platoon size is around 400 vehicles. Note that if the mean platoon size increases, the cycle time (which is the time between the departure of the first vehicle of two different platoons from the same lane) increases as well, since the arrival rate remains constant. As these large platoons are the most relevant difference between the routing matrices and PFA's, it is expected that the cause of the rapid increase in mean delay lies in the platoon sizes. With the k-limited PFA, this destabilization does not occur, strengthening the belief that it is caused by the size of the platoons, as the size is restricted for the k-limited PFA.

Figure 19b shows the corresponding heatmap for the exhaustive PFA. The heatmap is almost perfectly symmetrical, which is to be expected because the arrival process is symmetrical. It shows that the lanes with external arrivals perform the worst and that the lanes with large platoons in the corners perform best. Overall, lanes with large platoons compared to the other lanes at the same intersection, have small delays. This observation might explain the surge in mean delay. The clearest example is in one of the four corner intersections. It seems that the huge platoons arriving from two of the lanes suffocate the external arrivals. For intersection (14, 14), a histogram of the delays for all lanes is provided in Figure 20. We see that the large platoons on lane 0 and 3 are rarely significantly delayed. This is easily explained, as the vehicles departing from lanes 1 and 2 only have a small effect on the delay. Large delays only happen if a platoon arrives while a large platoon is still departing from another lane, which is unlikely as the cycletimes of the platoons are large. For lanes 1 and 2, large delays are more common, as on average a vehicle arrives every 4.5 seconds, meaning that there are nearly always vehicles having to wait for the large platoons to pass. We also see that the delays for lane 1 are slightly smaller than those of lane 2. This is explained by the fact that vehicles on lane 2 do not only have to wait on large platoons, but also on all the vehicles that have arrived at lane 1 during the departure of the platoon. This is because the lanes get access to the intersection in a clockwise order.

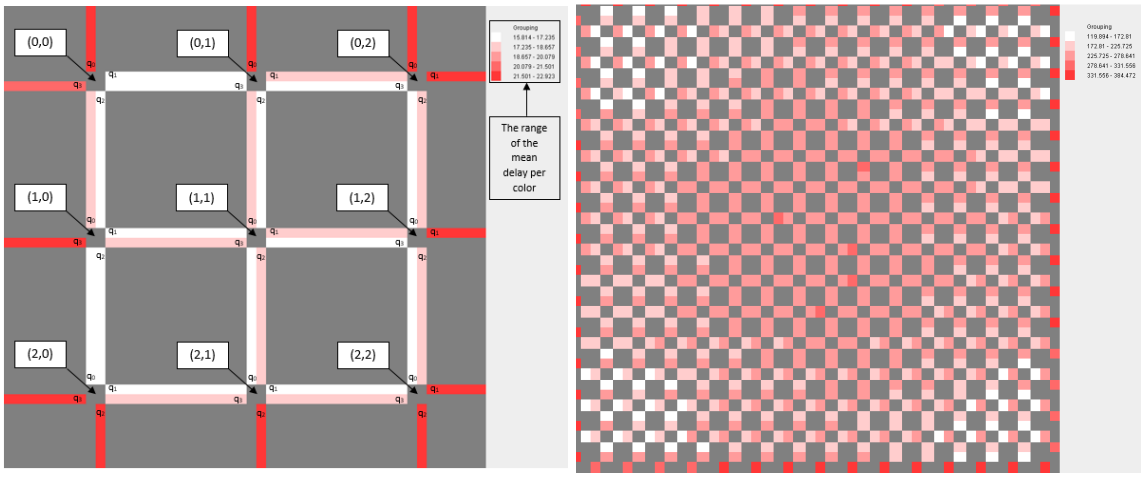
Apparently, the vehicles that are delayed more because of the increase of the mean platoon size with n have a larger impact on the mean delay than the vehicles that are delayed less because of the increasing cycle times of these large platoons. The histograms provide insight in the distribution of the delays, but do not provide a complete explanation for the fact that the system becomes unstable. We have tried to gain more insight in this destabilization process, but have not been able to find a manner in which we can conclusively give a cause.



(a) Exhaustive PFA

(b) k-limited ($k = 25$) PFA

Figure 18: Total mean delay for different sizes of the $n \cdot n$ grid



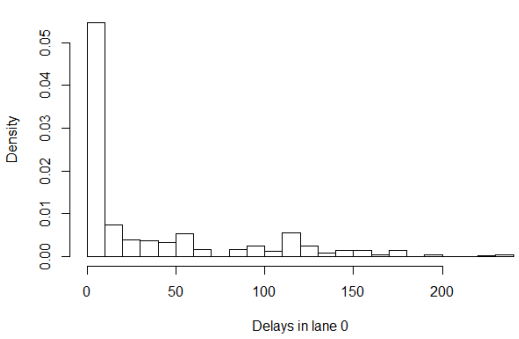
(a) Example

(b) $n = 15$ network with exhaustive PFA

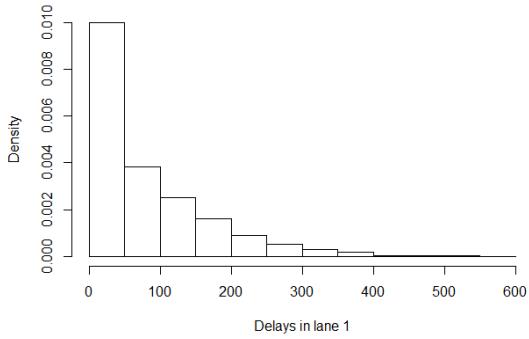
Figure 19: Heatmaps

5.4.3 Comparison of k-limited and exhaustive PFA

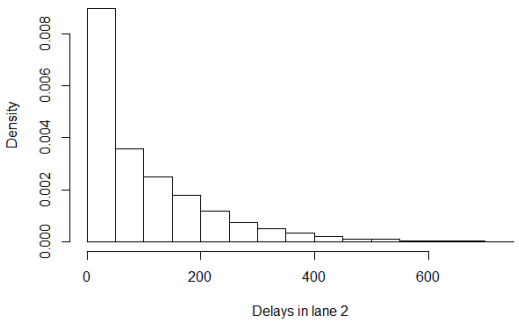
We have used the exhaustive and the k-limited ($k = 25$) PFA to approximate the mean delay in several situations. In a $n \cdot n$ network, the k-limited PFA performs slightly worse than the exhaustive PFA when applying routing matrix 2 or 3. However, this difference is less than a second. This is logical, as platoons of size 25 rarely happen with these routing rules. To give an indication, the mean platoon size is approximately 11 for both routing matrices. However, the performance measures for routing matrix 1 differ heavily. For the exhaustive PFA, the mean platoon size grows with every passing intersection, resulting in a destabilization of the network. For the k-limited PFA on the other hand, routing matrix 1 results in a lower mean delay than for the other routing matrices on a $15 \cdot 15$ network. In this case, the k-limited PFA significantly outperforms the exhaustive PFA. Furthermore, note that the seemingly linear increase of the mean delay for the k-limited PFA in combination with routing matrix 1 in Figure 18 does not continue to be linear. This was verified using $25 \cdot 25$ network. The total mean delay in this network was 22.4. This shows that the k-limited PFA prevents destabilization, also in even larger networks. There do not seem to be any significant drawbacks to the k-limited PFA when compared to the possible drawback of the exhaustive PFA. Therefore we conclude that the k-limited PFA should be preferred, even though routing matrix 1 is not realistic. This is a strong contrast with an isolated intersection, where the exhaustive PFA is provably optimal in terms of mean delay.



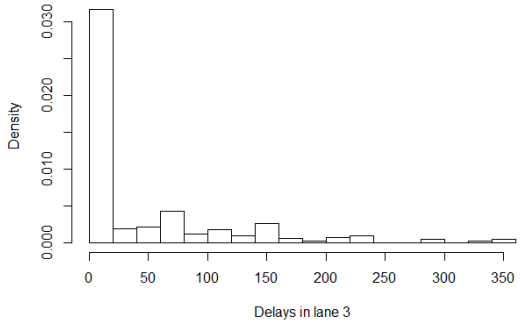
(a) Lane 0



(b) Lane 1



(c) Lane 2



(d) Lane 3

Figure 20: Histograms of the delays of intersection $(14, 14)$ per lane

6 Conclusion and discussion

There are numerous studies which suggest methods of organizing autonomous vehicles at intersections. In this final project, we have focused on one of these methods of organization; the platoon forming algorithm. More specifically, we compared the exhaustive and k-limited algorithm. We have studied the performance of these algorithms in several different networks. As opposed to a single intersection scenario, a larger network is no longer theoretically approachable. Therefore a simulation was required to compare the two PFA's. This comparison has been made for a multitude of scenarios. The main performance measure throughout the different scenarios has been the mean delay.

For a single intersection, we found that the exhaustive PFA performs best. This came as no surprise, as Timmerman and Boon [10] already found that the exhaustive PFA performs close to optimal in terms of mean delay. For two connected intersections and a tandem network of intersections, the same conclusion followed. The analysis of a tandem network even showed that the intersections in the middle, where the influence of the Poisson arrival process is the lowest, performed best. The analysis of a $n \cdot n$ network showed that it is actually the lanes with the largest platoons compared to the other lanes at the intersection that performed best, but that there is a turning point after which further increases in the platoon size cause a decrease in the mean delay. Furthermore, we found that routing plays an important role, even though the results from a small 1·5 network did not suggest this. A 15·15 network revealed that the combination of straight routing and the exhaustive PFA can lead to a destabilization of the network, due to ever increasing platoon sizes. The decrease in delay for some vehicles (either in one of the large platoons or vehicles from other lanes that can depart in the also increasing interarrival times of these large platoons) apparently do not outweigh the increase in delay experienced by the other vehicles that have to wait for these platoons. This phenomenon does not occur with the other tested routing matrices or the k-limited PFA. Even though such a large network in which all vehicles only drive straight ahead is unrealistic, the drawbacks to using the k-limited PFA are small and therefore it might be preferred in large networks. This is an interesting conclusion, as it showcases that a network can behave radically different from a single intersection on the same set of rules.

It has to be noted that we compared the two PFA's solely on bases of the mean delay. Timmerman and Boon [10] also took fairness into account and they found that the exhaustive PFA performs relatively poorly in this aspect. Therefore it might be better to compare the PFA's using multiple performance measures as this gives a well-rounded view. Also, we have not been able to validate the simulation for any networks larger than a single intersection. As theoretical approaches are not available for large networks, validation becomes difficult. Hence it would be sensible to reproduce the results, even though the output from the simulation does seem credible. Also, a simulation which continuously visualizes the position of the vehicles might provide more insight, as the results for large networks are difficult to interpret. This might also help to draw a more definitive conclusion regarding the destabilization of the network with the exhaustive PFA and straight routing. Another point of improvement is that we assumed that vehicles do not occupy any space. To clarify this, if a platoon of size m is delayed, these m vehicles are assumed to fit on the lane connecting the intersections. If the platoon sizes are large or the connecting lanes short, this is not realistic. Therefore, vehicles from other intersections that want to enter the lane with the delayed platoon, should have to wait for the delayed platoon to pass the intersection. Furthermore, the influence of many variables has not been studied. We have, for example, considered only a single value for the distance between intersections, the allowed following distance, the service time and the switch-over time between lanes. In this aspect, this study is rather narrow. Taking more of these variables and the effect they have on each other into consideration would be a useful research direction. Other expansion of this study could be to use different arrival processes or compare more PFA's. There are also other useful possibilities to expand this research. Ideas put forward by other studies, such as using compatible streams or green waves, might be compatible with the proposed model. Incorporating these might have a positive effect on the delay as well. A final recommendation is to apply this program to a network representing a city, so that the performance of the PFA's can be compared to the current situation. For this to be realistic, the vehicles would also have to travel according to a route, instead of randomly deciding the travel direction at each intersection.

Acknowledgments

First and foremost, I would like to thank my supervisor Marko Boon for assisting me throughout this project. I am really appreciative of all the help he offered me. Furthermore, I would like to thank my mother, Wilma Clement, for proofreading my report. Finally, I would like to thank everybody who provided me with some well-earned distraction after a long day of working on this project.

7 References

[1] M. Bashiri, H. Jafarzadeh, and C. H. Fleming, "Paim: Platoon-based autonomous intersection management," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 374–380.

[2] O. J. Boxma and W. P. Groenendijk, "Pseudo-conservation laws in cyclic-service systems," *Journal of Applied Probability*, vol. 24, no. 4, pp. 949–964, 1987.

[3] D. Carlino, S. D. Boyles, and P. Stone, "Auction-based autonomous intersection management," *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pp. 529–534, 2013.

[4] K. M. Dresner and P. Stone, "Multiagent traffic management: a reservation-based intersection control mechanism," *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 2004.*, pp. 530–537, 2004.

[5] M. Liu, M. Wang, and S. Hoogendoorn, "Optimal platoon trajectory planning approach at arterials," *Transportation Research Record*, vol. 2673, no. 9, pp. 214–226, 2019.

[6] D. Miculescu and S. Karaman, "Polling-systems-based Autonomous Vehicle Coordination in Traffic Intersections with No Traffic Signals," *IEEE Transactions on Automatic Control*, pp. 1–1, 2019.

[7] J. V. Saiáns-Vázquez, E. F. Ordóñez-Morales, M. López-Nores, Y. Blanco-Fernández, J. F. Bravo-Torres, J. J. Pazos-Arias, A. Gil-Solla, and M. Ramos-Cabrer, "Intersection intelligence: Supporting urban platooning with virtual traffic lights over virtualized intersection-based routing," *Sensors (Switzerland)*, vol. 18, no. 11, 2018.

[8] R. Tachet, P. Santi, S. Sobolevsky, L. I. Reyes-Castro, E. Frazzoli, D. Helbing, and C. Ratti, "Revisiting street intersections using slot-based systems," *PLOS ONE*, vol. 11, 2016.

[9] Y. Tian, D. Zhao, and J. Yi, "A fuzzy logic controller with adaptive dynamic programming optimization for traffic signals," in *Proceedings - 5th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2008*, vol. 3, 2008, pp. 191–195.

[10] R. W. Timmerman and M. A. A. Boon, "Platoon forming algorithms for intelligent street intersections," *arXiv*, 2019.

[11] C. Wuthishuwong and A. Traechtler, "Consensus coordination in the network of autonomous intersection management," *ICINCO 2014 - Proceedings of the 11th International Conference on Informatics in Control, Automation and Robotics*, vol. 2, pp. 794–801, 2014.

[12] C. Wuthishuwong, A. Traechtler, and T. Bruns, "Safe trajectory planning for autonomous intersection management by using vehicle to infrastructure communication," *Eurasip Journal on Wireless Communications and Networking*, vol. 2015, no. 1, pp. 1–12, 2015.

[13] F. Yan, M. Dridi, and A. E. Moudni, "An autonomous vehicle sequencing problem at intersections: A genetic algorithm approach," *International Journal of Applied Mathematics and Computer Science*, vol. 23, no. 1, 2013.