Master Thesis Mathematical Sciences

Optimal Departure Advice

Minimizing the waiting time and spreading the inflow at traffic lights

Supervisors:
Dr. Karma Dajani (UU)
Dr. Sandjai Bhulai (VU)
Dr. Rob van der Mei (CWI)
Frank Ottenhof (TrafficLink)

Author:
Maaike Hoogeboom

October 31, 2014

Abstract

During rush hours large congestions occur at intersections of main roads in residential areas. TrafficLink is developing an app to provide personal travel advice in order to spread traffic and decrease waiting times at traffic lights. This thesis focusses on the mathematics of the solution: the algorithm for the optimal departure advice.

First, a fast method is developed to predict the time-dependent queue length distribution based on the inflow and outflow at a traffic light. With this distribution the arrival time can be calculated such that a user is with confidence level $1-\alpha$ out of the queue before the deadline. Next, the users are distributed over both available traffic lights and time slots. After having proved the convexity of the value function of the total waiting time of the app users, the optimal solution can be found by a local search algorithm. This optimal schedule results in lower mean queue lengths and shorter waiting times, with users being on time with confidence level $1-\alpha$. The higher the fraction of users, the better the improvement.

Acknowledgement

First, I would like to thank all colleagues from TrafficLink and Trinité Automation, for the welcoming and educational internship. Especially, I would like to thank my supervisor Frank Ottenhof for the guidance and the opportunity to present my paper at the ITS European Congress in Helsinki. During my internship I learned a lot about traffic management and all the processes that lead to creating innovative software.

My supervisors of the VU University Amsterdam Dr. Sandjai Bhulai and Dr. Rob van der Mei also deserve by gratitude. I really appreciated the discussions about the research and the practical difficulties. My supervisor from University Utrecht, Dr. Karma Dajani was always willing to read my thesis and to give comments on my drafts, for which I thank her a lot.

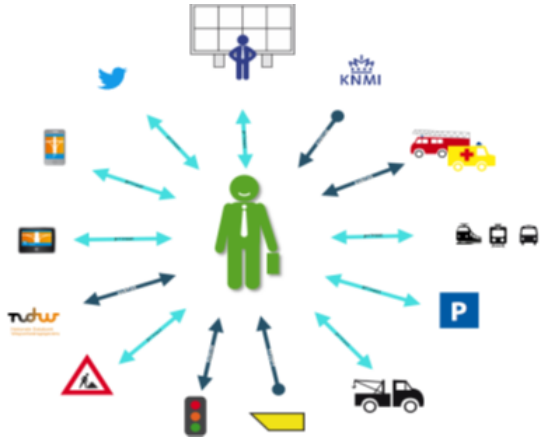
Contents

1 Introduction

1.1 Motivation

Suppliers of all types of navigation systems are routing travellers. This often leads to contradictory routing advices and in many cases even to network instability. Recent research [10] shows that if 15-20% of the users follow the advice of their navigation system, the network becomes unstable. The system can be improved significantly by coordination of all travellers. A striking phenomenon that supports this is that during the weekend the same number of vehicles is on the road as during a working day. Since in the weekend the traffic is more spread over time, there are far less traffic jams then during a normal working day. This shows that a better utilization of the road network by spreading the traffic, through personal travel advice, will help.

What is missing is an integrated approach: a proper virtual coordinator who ensures that all actors, public and private, tune their real time intentions. To realize this interconnected cooperation in which travellers are optimally facilitated, Trinité created a unique collaboration between Dutch companies, knowledge institutes and the government (the triple helix) which resulted in the ambition 2016 program. This program frameworks how parties can collaborate by (electronically) connecting them. The idea is that every physical road authority is coupled to a virtual road authority, the so-called Digital Road Authority (DRA). The Digital Road Authority is a tool that merges all traffic data from various sources into a smart travel advice. It is a coordinator that connects public and private parties. All parties that are connected to the Digital Road Authority can help each other to make their actions more effective.



A DRA is a virtual traffic manager that monitors the performance of the road network, and takes action when necessary. Digital Road Authorities can be connected to all measurement and control systems within their network, not only to roadside equipment (like sensors, ramp meters, traffic regulation installations, cameras, radars, detection loops) but also to in-car systems and applications. To use Digital Road Authorities, the road network is typically subdivided in regions, each of which is managed by a dedicated regional Digital Road Authority. These regions, are subdivided in smaller local sub-regions. In this way, the road network can be seen as a virtual tree of national networks, regional networks, local networks, and so on.

IJburg Pilot
Three pilots will be set up to start and test the Digital Road Authority. In this thesis we will focus on the IJburg pilot. In this pilot citizens of the neighbourhood IJburg will have a tool available, which is focussed on their area and optimizes traffic. In this Amsterdam Smart City project the residents will have the possibility to get a personalized departure advice by means of an app. The recommended departure time will be based on the occupation of the roads that have to be taken to arrive at the destination at the desired time. The main focus in this case is the occupancy of the roads to leave the neighbourhood. To reduce the congestion at these roads the travellers have to be distributed over the time slots in an efficient and smart manner.

1.2 Trinité Automation & TrafficLink

Trinité Automation[1] is the market leader in the field of dynamic traffic management. With the software of Trinité it is possible to manage and optimize the flow of traffic in a large area fully automatically. Trinité manages the rush-hour lanes, safety in tunnels, real-time video images of traffic, and route information on Dynamic Route Information Panels (DRIPs) and many more. All of this is done in an integral manner, from one interface. Trinité allows systems of different suppliers to work together, and makes automatic tuning between road managers from different areas possible.

TrafficLink[2] is a subsidiary of Trinité which sells a dynamic traffic management system that automatically controls all kinds of traffic managements tasks. With some basic functionalities an intersection can easily be controlled, but with advanced functionalities more complex networks like the area of Amsterdam are managed by TrafficLink.

1.3 Outline of the problem

During rush hours large queues are developing at the traffic lights on the main roads of residential areas. The traffic lights at these roads do not have the capacity to handle the large demand of traffic. Giving more green time to the congested road is in most cases no option, since the traffic light can only handle fixed green times or the successive road has a higher priority.

In this thesis we will give personal departure advice to the residents of a neighbourhood by means of an app. The main goal of these advices is to spread the inflow at the traffic lights and decrease the waiting times. We will distinguish two cases:

- A single app user
- Multiple app users

Single app user
It is not possible to spread the inflow and decrease the queue length if there are only a few app users. The personal departure advice will be to inform the user about the waiting time and ensure that the user is on time at the destination.

[1] Information from the Trinité website, www.trinite.nl/trinite.
[2] Information from the TrafficLink website, www.trafficlink.nl.

Before a personal departure advice can be given, a couple of problems have to be solved. First, we need a reliable travel time forecasting model to calculate the travel time from one point to another. In the literature there are many different models available to calculate the travel time, a couple of them are described in Appendix D. However, in this thesis we will focus on the queue at the traffic light, so the precise calculation of the travel time is omitted.

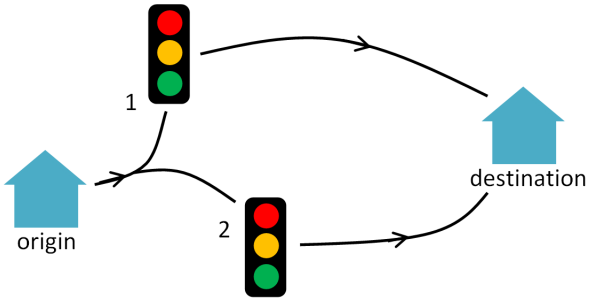


Figure 1: Graphical overview of the route decision problem over two traffic lights.

The three steps needed to achieve a personal departure advice for a single app user are:

Steps:

1. The time-dependent queue length distribution has to be known to calculate the waiting time.
2. The latest arrival time at the queue such that the app user is, with a pre-set confidence level, on time out of the queue has to be calculated.
3. A decision must be made about which route the app user has to take, via traffic light $1, 2, \ldots, S$.

The app distracts the preferred arrival time at the destination from the agenda of the user. Let D_1 be the departure time from traffic light 1 such that the app user is on the desired arrival time at the destination. Since the travel time will be calculated by the to be developed model, this departure time is assumed to be given. The departure time at traffic light D_1 will be used as the deadline to be out of the queue at the traffic light. In this thesis we want to find the optimal personal arrival time at the queue for every app user such that they are, with a pre-set confidence level, on time out of the queue (before D_1). This can be calculated by using the time-dependent queue length distribution. If the latest arrival times for traffic lights $1, 2, \ldots, S$ are known, we can decide which route the user has to take. This will result in an advice of the departure time and corresponding route.

Multiple users

When only a few persons use the app, the steps described above are enough to give a reliable departure advice. When many residents of the neighbourhood will use the app, the app cannot only give the optimal departure time but the users can also be distributed in a smart way over the time slots. Most of the methods developed in the single person case can also be used for multiple users. The components that are given in Figure 2 will all remain, only the algorithm will change. The additional part of distributing the travellers over the time slots will be most challenging.

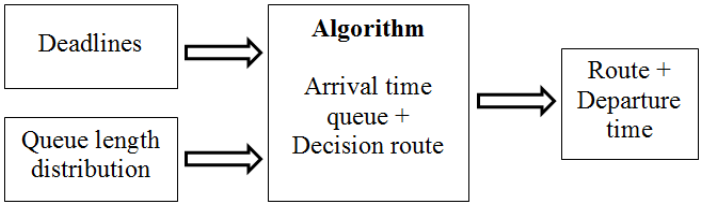


Figure 2: Components of the model.

IJburg

This thesis is inspired by the traffic problems in the neighbourhood IJburg. IJburg has two bridges to connect the neighbourhood with the region, these are the only two ways to leave and enter the neighbourhood. Especially in the morning and evening rush large queues are developing at these bridges, since the traffic lights at the end of the bridges can not handle the large demand. Hence IJburg is a special case of our model, since there are only two routes via traffic light 1 or 2.

1.4 Goals

In this thesis the following goals have to be achieved:

- Develop a model to describe the queue length in time.
- Give personal departure advice such that the user is on time at the destination.
- Divide the users over the traffic lights and spread the inflow over time per traffic light.
- Minimize the waiting times of the app users.
- Reduce the queue lengths.

Only the first two goals will be realized when a few citizens use the app. In the multiple users case all goals will be fulfilled.

1.5 Thesis Overview

First, the dynamics at a traffic light have to be known, therefore the arrival and departure processes at a traffic light are discussed in Section 2. In Section 3 different models to calculate the waiting times, as found in the literature, are presented. These theoretical models do not meet the requirements for the model. Therefore new methods to calculate the time-dependent queue length distribution are developed and discussed in Section 4. In Chapter 5 this queue length distribution is used to calculate the optimal arrival time such that the user is with confidence level $1 - \alpha$ on time out of the queue. Based on the arrival times the shortest route to the destination is selected, hence after this section a personal departure advice can be given to a single user. In Chapter 6 the case of multiple app users will be introduced. A value function will be used to minimize the total waiting time of all app users. In this section it is shown that the value function is convex, so a local search method will find the optimal schedule. The results of the local search algorithm are shown in Chapter 7. The last problems of ensuring that all users are on time out of the queue and dividing

the users over the traffic lights are discussed in Section 8. In the last paragraph of Chapter 8, all the parts are put together into one large algorithm. The performance of this algorithm is tested with a simulation and the results are presented and discussed in Section 9. In the next chapter we draw the final conclusions about the performance of our model. Some extensions of our model and future research are discussed in the last section.

2 Parameters traffic light

Arrival and departure processes are important dynamics in queue length prediction models. In this section we will focus on these two processes at a traffic light and the parameters belonging to them.

2.1 Arrival process

A traffic light has to adapt to the amount of traffic that arrives at the intersection. The number of vehicles that pass a point at the road per unit of time is called the flow or intensity. These intensities on a road are not steady, but fluctuate over time. There are many factors that influence the intensities, for example weather conditions, events, vacation periods etc. The intensity pattern also differs per type of road and per day of the week as shown in Figure 3.

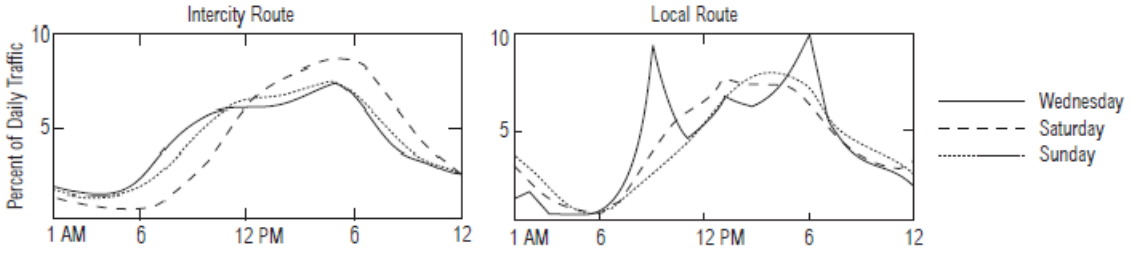


Figure 3: Daily pattern of the flow at a local and intercity road for different days of the week. Source [7].

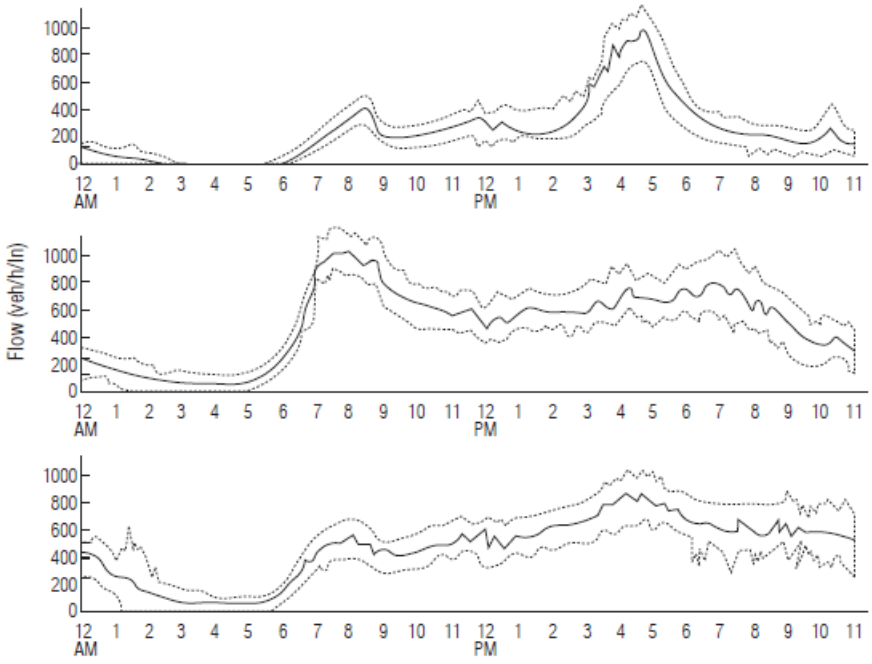


Figure 4: Daily pattern of the flow at three different urban roads in Toronto. Source [16].

In Figure 4 the daily pattern of the flow is shown for three urban roads in Toronto. The dotted

lines indicate the area in which 95% of the observations lies. The detectors measured traffic in one direction, since the observations have only a morning or evening peak, otherwise we would have seen both.

Before a design of a traffic light control is made, the intensities at the characteristic periods have to be known (morning and evening rush, intermediate period etc.). During rush hours the intensities are increasing and decreasing a lot over time, due to these large fluctuations the intensities have to be measured every 10 or 15 minutes. If the intensities do not fluctuate that much periods of 30 minutes are enough. In forecasting models the flow $\lambda(t)$ is dependent on time and changes every ΔT minutes (usually 10 or 15 minutes), this is shown in Figure 5 by the black lines.

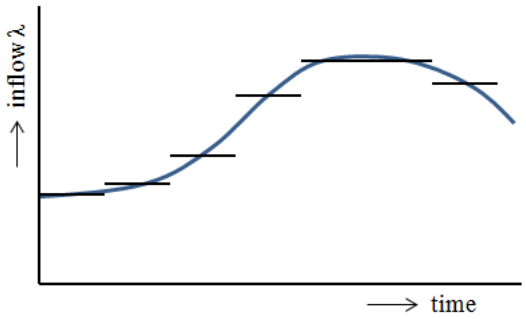


Figure 5: Discretization of the arrival flow.

2.2 Departure process

Different types of vehicles have different characteristics that must be taken into account. At departure a truck or bus has a longer acceleration time than a car, so they need more time to depart from a queue. If t_1 and t_2 are the mean acceleration times of a car and bus, respectively, then a bus needs t_2/t_1 more time than a car. Therefore, in a queue a bus is equivalent to a t_2/t_1 car. This value is called the pce-value (passenger car equivalent value), to determine the departure capacity, these values can be used. The pce-values differ a little bit in the literature, in [20] the following pce-values are given:

vehicle	pce-value
car	1
truck	1.5
bus	2
motor	0.4
moped	0.2

Table 1: Passenger car equivalent value per type of vehicle, source [20].

If the percentage of trucks and buses are low, these values are negligible. The fractions per type of vehicle are not known in most neighbourhoods. Since we are focusing on primarily residential areas, the percentage of trucks and buses will be low, so the pce-values are not taken into account for now.

The departure process of a homogeneous flow of cars at a traffic light can be approximately given by the graph in Figure 6.

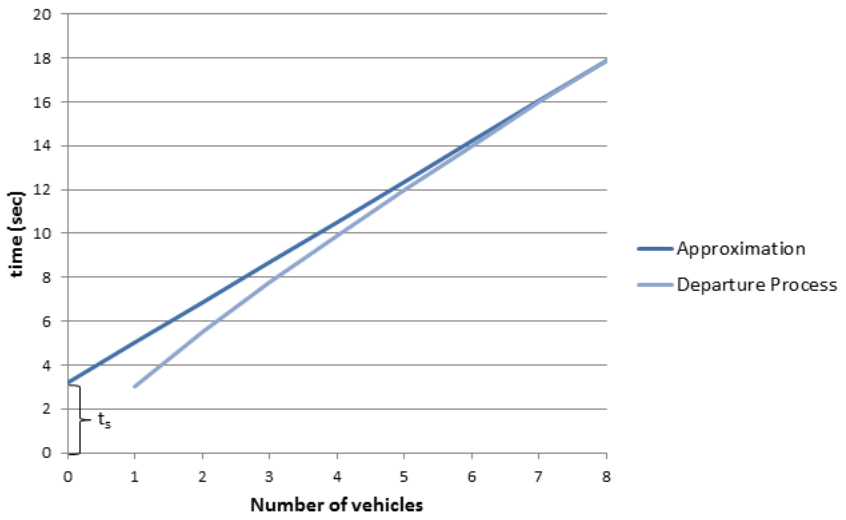


Figure 6: The number of vehicles departing in time during green light and the linear approximation of this process.

The headway times of the first vehicles leaving the queue during green time are dependent on the acceleration and the reaction times of the drivers. During green time the mean headway time decreases to a steady value. Research [20] has shown that 3 to 8 cars have to pass the traffic light before the rest of the cars will depart in a linear process, as can be seen in Figure 6.

Let t_s be equal to the sum of the acceleration and reaction times of the first vehicles leaving the queue. t_h is the fixed headway time of the vehicles in the second phase of the green time. Hence, the mean number of vehicles that depart during green time (T^{green}) can be calculated by $\frac{T^{green}-t_s}{t_h}$.

The departure capacity can best be determined by measurements, but this is a time consuming task, so in practice the departure capacity is often predicted.

Research [17] has shown that for the Netherlands 1900 pce/h for urban roads and 2000 pce/h for lanes outside urban areas are acceptable basic values. This basic value has to be multiplied with correction factors, i.e., factors that influence the departure capacity. There are many factors that affect the departure capacity, for example the width of the lane, the quality of the road, the weather and the slope of the road. Also driving direction, presence of bikers and blocking effects have to be taken into account. A detailed list of factors and how they can be used as correction factors of the departure capacity can be found in [17].

2.3 Scenarios

In this section a couple of scenarios of arrival flows during rush hours at a traffic light will be presented. Since there is no real inflow data of IJburg available, we will use these scenarios to test different models. The scenarios are based on available outflow data from the traffic light at IJburg [6] and data from [20].

In Table 2 four rush hour scenarios are given. The outflow rate of all scenarios is $\mu = 12$ vehicles/minute and the arrival flows change every quarter.

time	7:00	7:15	7:30	7:45	8:00	8:15	8:30	8:45	9:00	9:15	9:30	9:45
high	6	8	11	14	14	15	13	12	10	9	8	8
medium	6	8	10	12	13	13	12	11	9	9	8	7
low	6	7	8	9	10	11	11	10	9	7	7	6
peak	6	7	7	8	10	14	10	8	7	6	6	5

Table 2: Four scenarios of arrival flows (vehicles/minute) per quarter during rush hour.

In the first scenario high the system is overloaded for a long time, so the mean queue length will be high as can be seen in Figure 7. A medium and low scenario are also given, in the low scenario the inflow is always lower than the outflow. The last scenario peak has over all a low inflow rate, at only one time interval the arrival rate is high. This scenario can be seen as an example of a road nearby a school, with the peak just before the start of the school day. The one high inflow rate has a large impact on the mean queue length and on the profile of the scenario as shown in Figure 7.

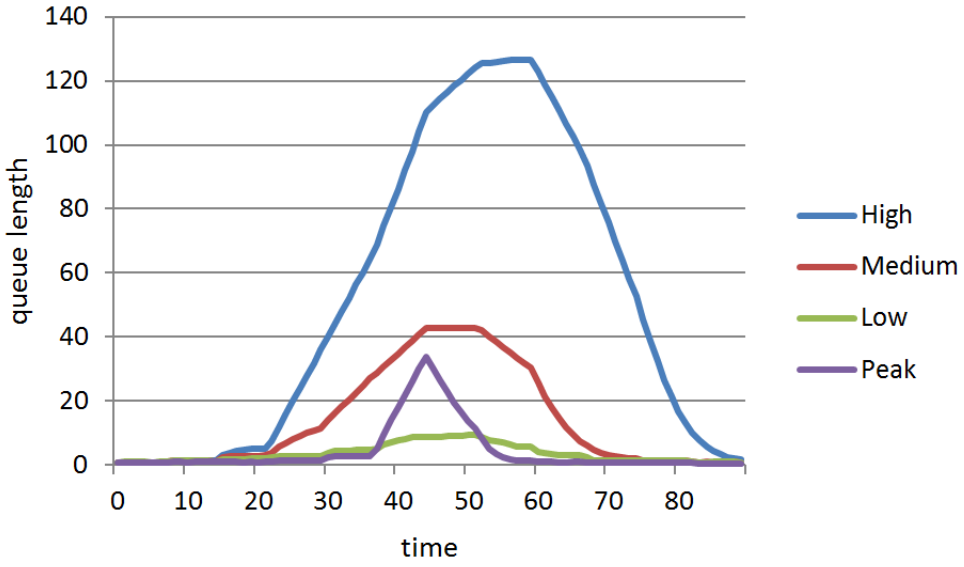


Figure 7: Mean queue lengths over time of the four scenarios.

3 Theoretical formulas

To give a reliable advice about the optimal arrival time at a queue at a traffic light, the waiting time distribution at this traffic light has to be known. A lot of research has been done on waiting times and queueing, in this section a couple of results will be presented. First, some formulas used in traffic engineering to calculate the average waiting time are shown. These formulas rely on many parameters and do not work well for overloaded systems. Therefore, we continue with M/M/1 queues in the second subsection. In that section exact formulas are given to calculate the state distribution of the queue which can be used to calculate the waiting time at a certain moment in time.

3.1 Mean waiting time

In the field of traffic engineering the fixed-cycle traffic light (FCTL) queue is one of the best studied models [8]. The first queueing models were developed between 1950s and 1960s. Webster is regarded as the founder of delay models, his formula is still the most famous result in FCTL queueing. Therefore Webster's formula will be presented as described in [20] and a couple of extensions will be discussed.

Assumptions of Webster's model:

- Traffic arrives uniformly distributed over time.
- In a cycle the inflow of traffic is less than the outflow.
- No initial queue.

In Figure 8 the situation of Webster's model is sketched, here C indicates the length of the traffic light cycle. The traffic light cycle consists of a red and green phase, g is the fraction of time the traffic light is green. Arrivals occur uniformly with q cars/sec, during green time s cars/sec are departing from the queue, so the queue is decreasing with $s - q$ cars/sec during green light.

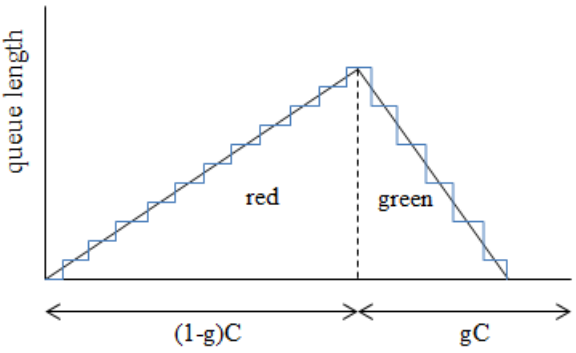


Figure 8: Queue length during a traffic light cycle under the assumptions of Webster's model.

The total waiting time is equal to the surface beneath the stepped graph, this is approximatively equal to the surface of the triangle.

The queue is empty at the start of the red period, so at the end of the red phase there are $q(1-g)C$ cars in the queue. The queue will decrease with $s-q$ cars/sec in the green phase, so the queue is empty again after $\frac{q(1-g)C}{s-q}$ seconds. Therefore, the total waiting time and the mean waiting time per vehicle is given by:

$$W_{total} = \frac{(1-g)Cq(1-g)C}{2} + \frac{q(1-g)C}{s-q}\frac{q(1-g)C}{2} = \frac{qC^2(1-g)^2}{2(1-\frac{q}{s})}$$
$$W = \frac{W_{total}}{qC} = \frac{C(1-g)^2}{2(1-\frac{q}{s})}$$

If $\frac{q}{s} \geq 1$ then W_{total} becomes negative, this is of course not possible, so the formula does not work for $\frac{q}{s} \geq 1$. The formulas are advised [20] to be used only for $\rho = \frac{q}{gs} < 0.8$, with ρ the occupation rate at the traffic light.
Since in rush hours this equation does not apply, we need a different model which can handle overloaded systems. In reality traffic arrives approximately Poisson distributed, therefore the number of vehicles that arrives during a cycle varies. If fewer cars arrive than depart in one cycle ($gs \gg q$), the fluctuations per cycle are small and the consequences of uniform arrivals will be little. If this is not the case, cars have to wait for the next green time which will have a huge influence on the average waiting time.

The formula of Akcelik can handle inflow rates that are higher than the number of cars leaving in the green time. Also Poisson distributed arrivals are assumed in this model. The first term in Akcelik's formula is equal to Webster's formula, but the second term is based on the so-called overflow queue (N_0), this is the mean number of cars in the queue at the end of the green phase. The overflow queue length is calculated by the formula given below as given in [20] and [17]:

$$N_0 = \begin{cases} 900sT\left(\rho - 1 + \sqrt{(\rho-1)^2 + \frac{12(\rho-\rho_0)}{3600sT}}\right) & \text{if } \rho > \rho_0 \\ 0 & \text{if } \rho \leq \rho_0 \end{cases}$$

Here T is the time period for which the given intensity holds and ρ_0 is the saturation level at which the overflow queue is zero. The formula of Akcelik [20] is given by:[3]

$$W = \frac{C(1-g)^2}{2(1-\frac{q}{s})} + \frac{N_0}{gs}$$

Note that gs is the departure rate over the whole traffic light cycle. The downside of this formula is that it only works for an empty initial queue, which is often not the case in overloaded systems. If in a period $\rho > 1$, then the initial state of the next period is larger than zero. Let q_b be the initial queue length, in the Highway Capital Manual [17] a formula is given to include the additional waiting time due to non-empty initial states $q_b > 0$.

If there is no initial queue Akcelik's formula holds. Three scenarios are described in [17] when $q_b > 0$: In the first scenario q_b plus the cars that arrive in the period T can be fully served in the

[3] In the Highway Capital Manual [17] a slightly different formula is given, which includes an extra factor for upstream controlling and a factor for incremental delay factor based on the controller settings.

time period T, hence $q_b + qT < gsT$. In the second scenario there is still some unmet demand left after period T, but the unmet demand is smaller than q_b, i.e., $q_b + qT > gsT$ and $qT < gsT$. The case of increasing unmet demand, $qT > gsT$, is covered in the last scenario.

The formula that contains the additional waiting time due to an initial queue at beginning of the period stated in the Highway Capital Manual is given below:

$$W_2 = \frac{1800 q_b (1 + dp) t}{gsT} \quad \text{with}$$

$$t = \begin{cases} \min(T, \frac{q_b}{gs(1-\min(1,\rho))}) & \text{if } q_b > 0 \\ 0 & \text{if } q_b = 0 \end{cases}$$

$$dp = \begin{cases} 1 - \frac{gsT}{q_b(1-\min(1,\rho))} & \text{if } t \geq T \\ 0 & \text{if } t < T \end{cases}$$

With t the duration of the unmet demand in T and dp the delay parameter. Adding W_2 to Akcelik's formula solves the problem of the initial queue for all scenarios described above. In the case of no initial queue W_2 is zero.

The formulas contain many factors that have to be set, but the main disadvantage is that the formulas only give the mean waiting time. To know if someone will be on time with a certain precision the distribution or percentiles of the queue length has to be known. In the literature we have found few formulas to calculate the percentiles of the queue length at a traffic light. Unfortunately these formulas only hold for systems with lower inflow than outflow rates. Since we are especially interested in overloaded systems, we will look for a different model.

3.2 State distribution of an M/M/1 queue

The time-dependent queue length distribution can be used to calculate the waiting time at a certain moment in time. There are many formulas known in the literature to calculate the state distribution of an M/M/1 queue, in this subsection we will present some of them.

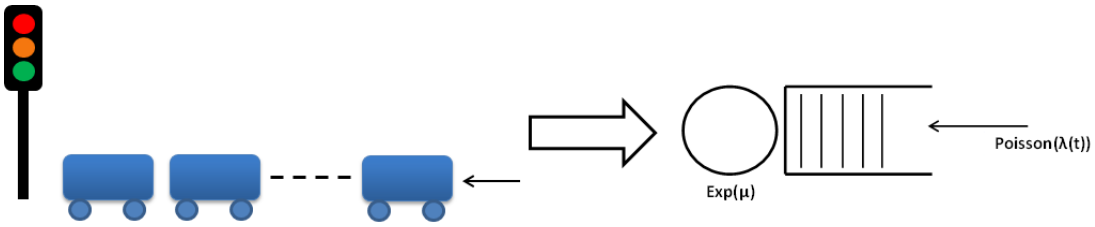


Figure 9: Graphical representation of translating a queue at a traffic light into an M/M/1 queue.

Before the queue length distribution can be calculated, the dynamics of the queue need to be set. We assume that the arrival times are $Poisson(\lambda(t))$ distributed, where $\lambda(t)$ is the arrival rate which changes over time. The outflow of the queue at a traffic light only occurs when the light is green. To make the case more general we assume that the 'service' time of a car in the queue is $Exponential(\mu)$ distributed [4], with of course FIFO discipline. The departure rate μ is dependent

[4] In reality this is of course not true, but we will relax this assumption later on (see Section 5 and 7).

on the length of the green time, it can be seen as the maximum number of vehicles that can cross the junction at the green time divided by the length of the traffic light cycle. For now we assume that the departure rate is fixed, so the green time does not change over time.
With these assumptions the queue is just a simple $M/M/1$ queue, with time-dependent arrival rates.

Let $\pi(n)$ be the exact probability that there are n vehicles in the queue. If the queue is in steady-state and the occupation rate $\rho = \frac{\lambda}{\mu} < 1$, then the well-known formula $\pi(n) = (1-\rho)\rho^n$ is used to calculate the state probabilities in the limit of an M/M/1 queue. However, in general the queue at a traffic light is not in steady-state, since the inflow changes over time. The most theoretical formulas to calculate the queue length distribution only hold for fixed arrival rates. We will discuss a couple of these formulas and at the end of the paragraph we show how to rewrite these formulas into the setting of time-dependent arrival rates.

Let $\pi_t^{(i)}(n)$ represents the probability that there are n vehicles in the queue at time t and i vehicles at time $t = 0$. In 1954 Bailey [14] used generating functions and Laplace transformations to obtain a formula to calculate this probability. For an empty initial state Bailey's formula is given below.

$$\pi_t^{(0)}(n) = \frac{1}{\mu t} e^{-(\lambda+\mu)t} \rho^n \sum_{r=n+1}^{\infty} \rho^{-r/2} r I_r(2t\sqrt{\lambda\mu}) \tag{1}$$
$$\text{with } I_r(z) = \sum_{m\geq 0} \frac{(z/2)^{r+2m}}{m!(m+r)!} \text{ the modified Bessel function}$$

Pegden and Rosenshine [15] investigated the joint probability $\pi_t^{(0)}(i,j)$ that during the interval $(0,t)$ i arrivals and j departures occur, when starting in an empty initial state. Sharma used this method to develop a different formula for $\pi_t^{(0)}(n)$, since $\pi_t^{(0)}(n) = \sum_{j\geq 0} \pi_t^{(0)}(j+n, j)$. This resulted in the following formula (for detailed information see [15]):

$$\pi_t^{(0)}(n) = (1-\rho)\rho^n + e^{-(\lambda+\mu)t}\rho^n \sum_{m=0}^{\infty} \frac{(\lambda t)^m}{m!} \sum_{k=0}^{m+n} (m-k) \frac{(\mu t)^{k-1}}{k!} \tag{2}$$

An attractive feature of this formula is that it is the sum of a constant part and a time-dependent part. The constant term is the steady-state limit of $\pi_t(n)$ as $t \to \infty$ and $\rho < 1$. Sharma's formula can be obtained directly from Bailey's formula. To show this we need the generating function of the modified Bessel function.

Corollary 3.1. *The generating function of the modified Bessel function is given by:*

$$\sum_{r=-\infty}^{\infty} I_r(z) s^r = e^{\frac{z}{2}(s+\frac{1}{s})} \tag{3}$$

Proof. To prove this corollary the generating function at the right hand side of Equation (3) will be rewritten into the left hand side.

$$\begin{aligned} e^{\frac{z}{2}(s+\frac{1}{s})} &= e^{\frac{zs}{2}} e^{\frac{z}{2s}} = \sum_{j=0}^{\infty} \frac{1}{j!} \left(\frac{zs}{2}\right)^j \sum_{k=0}^{\infty} \frac{1}{k!} \left(\frac{z}{2s}\right)^k \\ &= \sum_{j=0}^{\infty} \frac{1}{j!} \left(\frac{z}{2}\right)^j s^j \sum_{k=0}^{\infty} \frac{1}{k!} \left(\frac{z}{2}\right)^k s^{-k} \\ &= \sum_{r=-\infty}^{\infty} \left(\sum_{j-k=r, j,k\geq 0} \frac{(\frac{z}{2})^{j+k}}{j!k!} s^{j-k} \right) \end{aligned}$$

Take $j = k + r$ and fill this in, then we will find the modified Bessel function.

$$\begin{aligned} &= \sum_{r=-\infty}^{\infty} \left(\sum_{k=0}^{\infty} \frac{(\frac{z}{2})^{2k+r}}{(k+r)!k!} \right) s^r \\ &= \sum_{r=-\infty}^{\infty} I_r(z) s^r \end{aligned}$$

□

Now Corollary 3.1 is proven, we are ready to prove the following lemma, as done in [15].

Lemma 3.2. *Sharma's formula (2) can be derived from Bailey's formula (1).*

Proof. By differentiation of both sides of Equation (3) with respect to s we get:

$$\sum_{r=-\infty}^{\infty} r s^{r-1} I_r(z) = e^{\frac{z}{2}(s+\frac{1}{s})} \left(\frac{z}{2} \left(1 - \frac{1}{s^2} \right) \right)$$

By substituting $s = \rho^{-\frac{1}{2}} = \sqrt{\frac{\mu}{\lambda}}$ and $z = 2t\sqrt{\lambda\mu}$ on both sides, the equation becomes:

$$\sum_{r=-\infty}^{\infty} r \rho^{-\frac{1}{2}(r-1)} I_r(2t\sqrt{\lambda\mu}) = e^{t\sqrt{\lambda\mu}(\rho^{-\frac{1}{2}} + \rho^{\frac{1}{2}})} (t\sqrt{\lambda\mu}(1-\rho))$$

$$\rho^{\frac{1}{2}} \sum_{r=-\infty}^{\infty} r \rho^{-\frac{r}{2}} I_r(2t\sqrt{\lambda\mu}) = e^{(\lambda+\mu)t} t\sqrt{\lambda\mu}(1-\rho)$$

By rearranging and simplifying we get:

$$1 - \rho = e^{-(\lambda+\mu)t} \frac{1}{t\sqrt{\lambda\mu}} \frac{\sqrt{\lambda}}{\sqrt{\mu}} \sum_{r=-\infty}^{\infty} r \rho^{-\frac{r}{2}} I_r(2t\sqrt{\lambda\mu})$$

$$1 - \rho = \frac{1}{\mu t} e^{-(\lambda+\mu)t} \sum_{r=-\infty}^{\infty} r \rho^{-\frac{r}{2}} I_r(2t\sqrt{\lambda\mu})$$

$$(1-\rho)\rho^n = \frac{1}{\mu t} e^{-(\lambda+\mu)t} \rho^n \sum_{r=-\infty}^{\infty} r \rho^{-\frac{r}{2}} I_r(2t\sqrt{\lambda\mu})$$

By multiplying both sides with ρ^n the last equation is found. The right hand side of this equation is similar to Bailey's formula (1), only the summation starts at $r = -\infty$ instead of $n+1$. By using this last equation, Bailey's formula (1) can be written as:

$$\pi_t^{(0)}(n) = (1-\rho)\rho^n - \frac{1}{\mu t} e^{-(\lambda+\mu)t} \rho^n \sum_{r=-\infty}^{n} r\rho^{-\frac{r}{2}} I_r(2t\sqrt{\lambda\mu})$$

Fill in the modified Bessel function and simplify

$$\begin{aligned}
&= (1-\rho)\rho^n - \frac{1}{\mu t} e^{-(\lambda+\mu)t} \rho^n \sum_{r=-\infty}^{n} r\rho^{-\frac{r}{2}} \sum_{m=0}^{\infty} \frac{(t\sqrt{\lambda\mu})^{r+2m}}{m!(m+r)!} \\
&= (1-\rho)\rho^n - \frac{1}{\mu t} e^{-(\lambda+\mu)t} \rho^n \sum_{r=-\infty}^{n} \sum_{m=0}^{\infty} r \frac{(\lambda t)^m}{m!} \frac{(\mu t)^{m+r}}{(m+r)!}
\end{aligned}$$

The last step is to substitute $k = m + r$ and invert the order of summation:

$$\pi_t^{(0)}(n) = (1-\rho)\rho^n + e^{-(\lambda+\mu)t} \rho^n \sum_{m=0}^{\infty} \frac{(\lambda t)^m}{m!} \sum_{k=0}^{m+n} (m-k) \frac{(\mu t)^{k-1}}{k!}$$

Hence we have derived Sharma's formula out of Bailey's formula. □

Sharma's formula (2) only holds when starting in an empty system. Conolly generalized Sharma's formula to an arbitrary initial state as can be found in [5]:

$$\begin{aligned}
\pi_t^{(i)}(n) = &(1-\rho)\rho^n + e^{-(\lambda+\mu)t} \rho^n \sum_{m=0}^{\infty} \frac{(\lambda t)^m}{m!} \sum_{k=0}^{m+n+i+1} (m-k) \frac{(\mu t)^{k-1}}{k!} + \\
&e^{-(\lambda+\mu)t} \rho^n \sum_{m=0}^{\infty} \frac{(\mu t)^{m+1} (\mu t)^{m+\max(i,n)}}{m!} \left\{ \frac{(\lambda t)^{-\min(i,n)-1}}{(m+|i-n|)!} - \frac{(\mu t)^{\min(i,n)+1}}{(m+n+i+2)!} \right\}
\end{aligned} \tag{4}$$

3.2.1 Time-dependent arrival rates

These formulas use fixed arrival rates, so a solution has to be found to include time-dependent arrival rates. Suppose that the arrival rate changes every $\triangle T$ minutes and the distribution at time $i\triangle T$ is known, then the distribution at time $(i+1)\triangle T$ can be calculated as follows:

$$\pi_{(i+1)\Delta T}(n) = \sum_{j=0}^{N} \pi_{i\Delta T}(j) \pi_{\Delta T}^{(j)}(n) \text{ for } \forall n \in \{0, 1, \ldots, N\}. \tag{5}$$

A drawback of this method is that the summation creates a discrete process out of the continuous formula. The state distribution can still be calculated every moment, for example if we want to know the distribution at time $i\Delta T + \tau$ with $0 \leq \tau < \Delta T$, this can be calculated by:

$$\pi_{i\Delta T+\tau}(n) = \sum_{j=0}^{N} \pi_{i\Delta T}(j) \pi_{\tau}^{(j)}(n) \text{ for } \forall n \in \{0, 1, \ldots, N\}.$$

For convenience we calculate the state distribution every Δt minutes, so ΔT should be a multiple of Δt: $\Delta T = c\Delta t$ with $c \in \mathbb{N}_{>0}$.

Computationally Conolly's formula (4) is superior above Bailey's formula (1). The computational time for fixed λ-value to calculate the probabilities for the states $\chi = \{0, 1, \ldots, 20\}$ at time t is around the 42 seconds with Conolly's formula. For two intervals with two different λ-values the distribution as given in Formula (5) has even a computational time of 14.30 minutes. Another major drawback of Conolly's formula is that it does not work if $\rho \geq 1$. The formula exists of a constant part and a time-dependent part, the constant term is the steady-state limit of $\pi_t(n)$ as $t \to \infty$ and $\rho < 1$. If $\rho \geq 1$, $\pi_t(n)$ should go to zero, so the constant term must be canceled. In Conolly's formula $\pi_t(n)$ converges to the constant part $(1-\rho)\rho^n$ if t is large and thus gives negative values for $\rho \geq 1$ as we will show in Subsection 4.3.

Conolly has developed a new formula which works for every value of ρ. This formula is more complex, but for completeness it is given below:

$$\pi_t(n) = \left(1 - \frac{(\lambda+\mu) - |\lambda-\mu|}{2\mu}\right)\rho^n + e^{-(\lambda+\mu)t}\sum_{m\geq 0} c_m^{(n)} t^m$$

where $c_m^{(n)}$ can be obtained from the differential equations:

$$\begin{aligned} c_m^{(1)} &= \frac{m+1}{\mu}c_{m+1}^{(0)} - c_m^{(0)} \\ c_m^{(n)} &= \frac{m+1}{\mu}c_{m+1}^{(n-1)} - \rho c_m^{(n-2)} \quad \text{for} \quad n = 2, 3, \ldots \end{aligned}$$

where $c_m^{(0)}$ is the coefficient of t^m in $\frac{\lambda+\mu}{2\mu}\sum_{m\geq 1}\binom{1/2}{m}\left(\frac{4\lambda\mu}{(\lambda+\mu)^2}\right)m\sum_{k=0}^{2m-2}\frac{((\lambda+\mu)t)^k}{k!}$ for details see [2].

This formula isolates the constant term when $\rho < 1$ and cancels it when $\rho \geq 1$.

Due to the long calculation time, we will not use these theoretical formulas. In the next section we will develop a new method to calculate the distribution in a faster way.

4 Time-dependent queue length distribution

As discussed in the previous section, the formulas found in the literature do not fit the requirements of our model. In this section we will develop and discuss different methods to calculate the time-dependent queue length distribution. In the first subsection the queue length distribution is calculated using a transition matrix. With this transition matrix an approximation of the queue length distribution will be presented in the third subsection. In the last subsection the computational results of the different methods are compared and discussed.

4.1 Transition matrix

The time-dependent queue length distribution can be calculated with transition probabilities. The transition rates of an M/M/1/N queue are given by the black arrows in Figure 10.

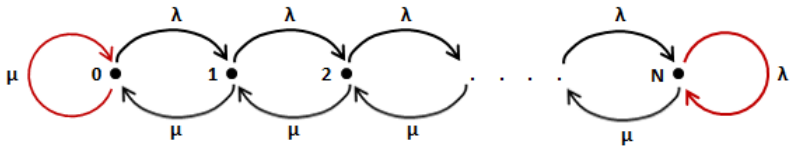


Figure 10: The black arrows are the transition rates of an M/M/1/N queue, the red arrows are added by uniformization.

We assume that the maximum number of cars in the queue is N. This is a reasonable assumption since a road is finite and a queue cannot become infinitely long. Therefore let N be the number of vehicles that fit on the road or be an upper limit of the queue length.

The transition rates out of a state are $\lambda + \mu$ for states $1 \leq x \leq N-1$, λ for $x = 0$ and μ for $x = N$. Therefore the time to move from one state to the next is not the same for every state, but equal to some random variable $T(x)$. Since the inter arrival and service times are both exponentially distributed the time in a state $T(x)$ is exponentially distributed for all states $x \in \chi = \{0, 1, .., N\}$. Let $\sigma(x, y)$ be the transition rates, so $\sigma(x, x+1) = \lambda$ for $0 \leq x \leq N-1$ and $\sigma(x, x-1) = \mu$ for $1 \leq x \leq N$ otherwise $\sigma(x, y) = 0$. By standard properties of the exponential distribution, $T(x)$ is exponentially distributed with rate $\sum_{y \in \chi} \sigma(x, y)$. The probability that the state moves from x to y is given by $p(x, y) = \frac{\sigma(x,y)}{\sum_{y \in \chi} \sigma(x,y)}$.

To calculate the state distribution at a specific time it is convenient if $T(x)$ is equally distributed for all $x \in \chi$. Therefore dummy transitions are added to make the rate out of a state equal for all $x \in \chi$, this is called uniformization. Let $\gamma = \max_{x \in \chi} \sum_{y \in \chi} \sigma(x, y)$, if for a state $x \in \chi$, $\sum_{y \in \chi} \sigma(x, y) < \gamma$ then we add the dummy variable $\sigma'(x, x) = \gamma - \sum_{y \in \chi} \sigma(x, y)$. By including these dummy variables the expected transition time is $1/\gamma$ for all states $x \in \chi$.
In the case of the M/M/1/N queue, only two dummy variables have to be added: $\sigma'(0, 0) = \mu$ and $\sigma'(N, N) = \lambda$, the red arrows in Figure 10. We will continue with this uniformized process.

Let $\varphi_i(n)$ be the probability that at transition i the process is in state n. Then by forward recursion the probabilities of the next transition can be calculated by:

$$\varphi_{i+1}(0) = \frac{\mu}{\lambda+\mu}\varphi_i(1) + \frac{\mu}{\lambda+\mu}\varphi_i(0)$$
$$\varphi_{i+1}(x) = \frac{\mu}{\lambda+\mu}\varphi_i(x+1) + \frac{\lambda}{\lambda+\mu}\varphi_i(x-1) \text{ if } 1 \leq x \leq N-1$$
$$\varphi_{i+1}(N) = \frac{\lambda}{\lambda+\mu}\varphi_i(N-1) + \frac{\lambda}{\lambda+\mu}\varphi_i(N)$$

Therefore the probability to move from state x to state y with $x, y \in \mathbb{N}$ is given by:

$$p(x,y) = \begin{cases} \frac{\lambda}{\lambda+\mu} & \\ \frac{\mu}{\lambda+\mu} & \text{if} \quad y = x-1 \text{ or } y = x = 0 \\ 0 & \text{otherwise} \end{cases}$$

In matrix form this is given by $P_{xy} = p(x,y)$. At a traffic light the inflow λ is not fixed but will change over time, therefore the distribution is dependent on λ. We will continue to write $P(\lambda)$, to show the dependence on λ.

$$P(\lambda) = \begin{pmatrix} \frac{\mu}{\lambda+\mu} & \frac{\lambda}{\lambda+\mu} & 0 & 0 & \dots & 0 \\ \frac{\mu}{\lambda+\mu} & 0 & \frac{\lambda}{\lambda+\mu} & 0 & \dots & 0 \\ 0 & \frac{\mu}{\lambda+\mu} & 0 & \frac{\lambda}{\lambda+\mu} & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & & 0 & \frac{\mu}{\lambda+\mu} & 0 & \frac{\lambda}{\lambda+\mu} \\ 0 & \dots & \dots & 0 & \frac{\mu}{\lambda+\mu} & \frac{\lambda}{\lambda+\mu} \end{pmatrix}$$

4.1.1 Distribution with transition matrix

The queue length distribution can also be calculated with the transition matrix $P(\lambda)$. The matrix gives the probability that a specific state change occurs. Let λ and μ be the transition rates per interval of length $t = 1$, for example each minute. Due to uniformization the time to move from one state to the next is $Exponential(\lambda + \mu)$ distributed for all states. Therefore the number of steps that have to be taken in a time interval of length t is $Poisson((\lambda+\mu)t)$ distributed. The probability that there are m steps in time interval t is $Poisson(m, (\lambda+\mu)t)$ and the probability to move from one state to another in m steps is given by the matrix multiplication $P^m(\lambda)$. Hence if π_0 is the initial distribution at time 0 then the distribution at time t is given by:

$$\begin{aligned} \pi_t =& \pi_0 \sum_{m=0}^{\infty} Poisson(m, (\lambda+\mu)t) P^m(\lambda) \\ =& \pi_0 \sum_{m=0}^{\infty} \frac{(\lambda+\mu)^m t^m}{m!} e^{-(\lambda+\mu)t} P^m(\lambda) \\ =& \pi_0 P_{large}(\lambda, t) \text{ with } P_{large}(\lambda, t) = \sum_{m=0}^{\infty} \frac{(\lambda+\mu)^m t^m}{m!} e^{-(\lambda+\mu)t} P^m(\lambda) \end{aligned} \tag{6}$$

With $\pi_t = (\pi_t(0), \pi_t(1), ..., \pi_t(N))$ the distribution at time t. The distributions given by this formula are similar to the distributions found by Conolly's formula (4), but this method also holds for arrival rates higher than the departure rate ($\lambda > \mu$). Another important advantage of this method is that $P_{large}(\lambda, t)$ only has to be recalculated when the λ value changes. Suppose that the inflow changes every ΔT minutes, then:

$\pi_{\Delta T} = \pi_0 P_{large}(\lambda_1, \Delta T)$
$\pi_{2\Delta T} = \pi_{\Delta T} P_{large}(\lambda_2, \Delta T) = \pi_0 P_{large}(\lambda_1, \Delta T) P_{large}(\lambda_2, \Delta T)$ etc.

In general the state distribution is calculated in discrete time steps of Δt minutes, so for convenience ΔT should be a multiple of Δt. Then we get:

$\pi_{\Delta t} = \pi_0 P_{large}(\lambda, \Delta t)$
$\pi_{2\Delta t} = \pi_{\Delta t} P_{large}(\lambda, \Delta t) = \pi_0 P^2_{large}(\lambda, \Delta t)$ when $0 \leq 2\Delta t \leq \Delta T$ etc.

The calculation time of this method is long since the sum in formula (6) goes to infinity. For a fixed λ value the calculation time of the distribution is around 48 seconds, so the same as for Conolly's formula. When the λ value changes, only $P_{large}(\lambda, \Delta t)$ has to be recalculated. This saves a lot of calculation time compared to Conolly's formula. For two intervals with different λ-rates the calculation time is 107 seconds, this is a major improvement.

Formula (6) gives exactly the same distribution as Conolly's formula, but in a lower calculation time. Since this formula is also directly applicable for $\lambda \geq \mu$ and can easily deal with changing arrival rates, this method is preferable over Conolly's formula. Unfortunately the calculation time is still too long, therefore we will search for an approximation of the queue length distribution.

4.2 Approximation queue length distribution

The time that the process stays in one state is $Exponential(\lambda + \mu)$ distributed for all states in the uniformized case. Hence the expected time that the process stays in one state is $\frac{1}{\lambda+\mu}$. In this approximation we take $\frac{1}{\lambda+\mu} = \frac{1}{\gamma}$ as the fixed discrete time step at which a transition occurs. Let $\widehat{\varphi}_i(n)$ be the expected probability that the queue has length n at time step $\frac{i}{\gamma}$, then the expected distribution at time step $\frac{i+1}{\gamma}$ can be calculated by $\widehat{\varphi}_{i+1} = \widehat{\varphi}_i P(\lambda)$ with $\widehat{\varphi}_i = (\widehat{\varphi}_i(0), \widehat{\varphi}_i(1), ..., \widehat{\varphi}_i(N))$. This is a discrete-time process with time step $\frac{i}{\gamma}$. In general the state distribution is calculated every Δt minutes, then the expected distribution at time $(k+1)\Delta t$ is calculated by $\widehat{\pi}_{(k+1)\Delta t} = \widehat{\pi}_{k\Delta t} P^{\gamma \Delta t}(\lambda) = \widehat{\pi}_{k\Delta t} P^{(\lambda+\mu)\Delta t}(\lambda)$.

Just as in the previous method with the $P_{large}(\lambda)$ matrix, this approximation is very useful when dealing with time-dependent arrival rates. If the arrival rate changes, only the λ value in the matrix $P(\lambda)$ has to be adapted. Suppose that the arrival rate changes every $\Delta T = 10$ minutes and $\Delta t = 1$, then we have to adjust the $P(\lambda)$ matrix every 10 iterations.

4.2.1 Improvement

The approximation in the previous section is very rough and can be improved by decreasing the length of the fixed time step. In the previous section the time step is $\frac{1}{\lambda+\mu}$, this time step can decrease by adding dummy transitions. The rate out of the state stays the same, but every time step the system stays in the current state with rate δ. Hence the time in one state is $Exponential(\lambda+\mu+\delta)$ distributed until a transition occurs, note that a transition can also be to the current state. In Figure 11 the transition diagram is shown.

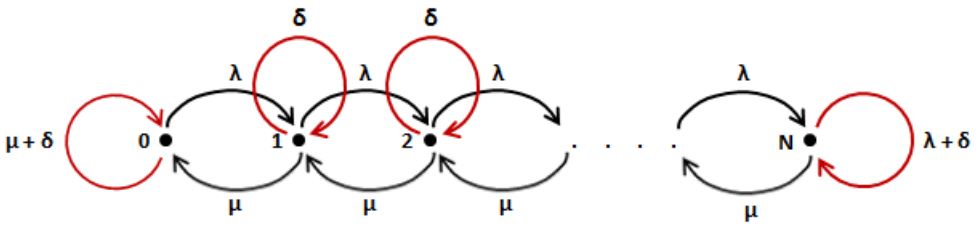


Figure 11: The black arrows are the transition rates of an M/M/1/N queue, the red arrows are added dummy variables.

Let $\widehat{\varphi}_i(n)$ be the expected probability that the process is in state n at time step i, where a time step has length $\frac{1}{\lambda+\mu+\delta}$. Then by forward recursion the probabilities of the next time step can be calculated by:

$$\begin{aligned}
\widehat{\varphi}_{i+1}(0) &= \frac{\mu}{\lambda+\mu+\delta}\widehat{\varphi}_i(1) + \frac{\mu+\delta}{\lambda+\mu+\delta}\widehat{\varphi}_i(0) \\
\widehat{\varphi}_{i+1}(x) &= \frac{\mu}{\lambda+\mu+\delta}\widehat{\varphi}_i(x+1) + \frac{\delta}{\lambda+\mu+\delta}\widehat{\varphi}_i(x) + \frac{\lambda}{\lambda+\mu+\delta}\widehat{\varphi}_i(x-1) \quad \text{if } 1 \leq x \leq N-1 \\
\widehat{\varphi}_{i+1}(N) &= \frac{\lambda}{\lambda+\mu+\delta}\widehat{\varphi}_i(N-1) + \frac{\lambda+\delta}{\lambda+\mu+\delta}\widehat{\varphi}_i(N)
\end{aligned}$$

Let $C = \lambda+\mu+\delta$, the transition matrix will look as follows:

$$\overline{P}(\lambda) = \frac{1}{C}\begin{pmatrix}
\mu+\delta & \lambda & 0 & 0 & \dots & 0 \\
\mu & \delta & \lambda & 0 & \dots & 0 \\
0 & \mu & \delta & \lambda & \ddots & 0 \\
\vdots & \ddots & \ddots & \ddots & \ddots & \\
0 & & 0 & \mu & \delta & \lambda \\
0 & & & 0 & \mu & \lambda+\delta
\end{pmatrix}$$

Hence $\widehat{\varphi}_{i+1}$ can be calculated by $\widehat{\varphi}_i$ by the forward recurrence $\widehat{\varphi}_{i+1} = \widehat{\varphi}_i\overline{P}(\lambda)$. In reality the state distribution is calculated every Δt minutes, so

$$\widehat{\pi}_{(k+1)\Delta t} = \widehat{\pi}_{k\Delta t}\overline{P}^{C\Delta t}(\lambda) = \widehat{\pi}_{k\Delta t}\overline{P}^{(\lambda+\mu+\delta)\Delta t}(\lambda) \text{ for } \forall k \in \mathbb{N}. \tag{7}$$

Hence for $\Delta t = 1$ the formula will be $\widehat{\pi}_{k+1} = \widehat{\pi}_k \overline{P}^C(\lambda) = \widehat{\pi}_k \overline{P}^{\lambda+\mu+\delta}(\lambda)$ for all $k \in \mathbb{N}$. In this general approximation method the matrix $\overline{P}(\lambda)$ only has to be adapted when the inflow changes. Therefore the matrix $\overline{P}^{(\lambda+\mu+\delta)\Delta t}(\lambda)$ can be fixed if we take time steps of length Δt and only has to be adjusted when the λ value changes.

The larger δ, the more precise the approximation is. When $\delta \to \infty$ then the time steps go to zero and the approximation converges to the real distribution. The previous method uses no dummy variable $\delta = 0$ and is a special version of this general approach.

4.3 Computational results

In this section the different methods to calculate the queue length distribution are discussed. We will compare the theoretical matrix method (6) with the approximation method (7), but first we will compare the two theoretical functions.

Conolly's formula (4) and the matrix formula (6) give exactly the same time-dependent queue length distribution if $\rho < 1$. Only the calculation time of the matrix approach is much lower, especially when the λ value is time-dependent. Next to the long calculation time Conolly's formula has another major disadvantage: it does not work for $\rho \geq 1$. If t is large $\pi_t(n)$ converges to the constant part $(1-\rho)\rho^n$ of the formula, but this only holds if $\rho < 1$ and gives negative values if $\rho \geq 1$. In Figure 12 the probability that the queue is empty is shown over time for $\rho = 12/10$. For $1 \leq t \leq 6$ Conolly's formula gives the same probabilities as the matrix formula, but from $t \geq 7$ Conolly's probability converges to $1 - \rho$. It depends on the value of ρ at which time Conolly's formula starts converging to the constant term, if ρ is higher the convergence is faster and vice versa.

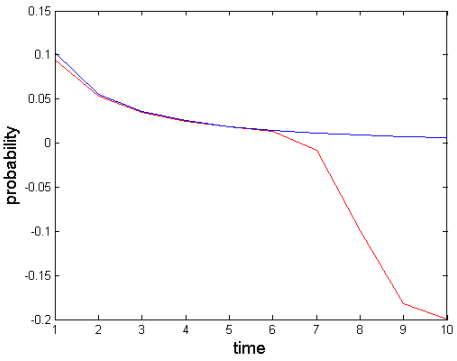


Figure 12: Probability that the queue length is 0 calculated with Conolly's formula (red graph) and the matrix formula (blue graph), with initial state 0, $\lambda = 12$ and $\mu = 10$ so $\rho = 1.2$.

In the morning and evening rush hours there are moments that the inflow at the traffic lights will be higher than the outflow, therefore Conolly's formula is not applicable for our problem. Since the calculation time of this formula is very long and it can not easily deal with time-dependent arrival rates, we will not look at the new developed formula of Conolly that can handle processes with $\rho \geq 1$.

Hence in the remaining of this paragraph the performance of the approximation distribution (7) will be compared to the theoretical distribution (6).

In the approximation method the state changes every $\frac{1}{\lambda+\mu+\delta}$ minute, so this is a discrete-time process, in contrary to the theoretical formula that can be calculated continuously. In Figure 13 the theoretical and approximated distribution are shown at times $t = 1, 2, 10$.

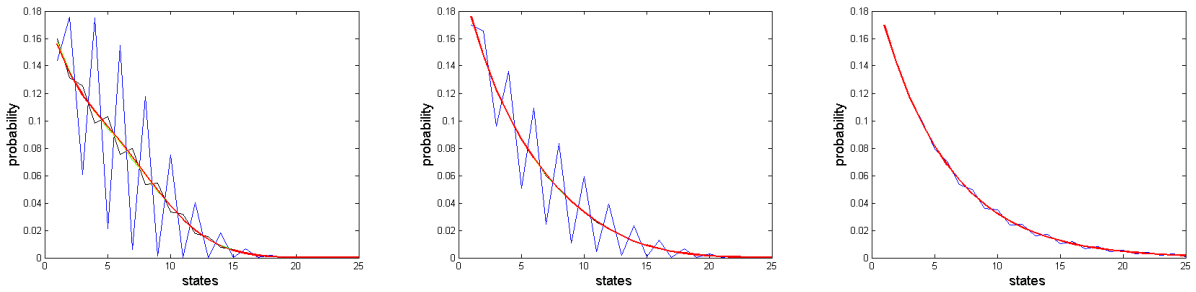


Figure 13: For $\lambda = 10$, $\mu = 12$ and $i = 5$ the state distributions are calculated with the theoretical formula (red graph) and with the approximation approach with different values of δ ($\delta = 0$ blue graph, $\delta = 1$ black graph, $\delta = 2$ green graph, $\delta = 5$ yellow graph), at time $t = 1$ left, $t = 2$ middle and $t = 10$ right.

In these figures can be seen that the state distributions given by the theoretical formula represent a fluent graph for all values of t. The most rough approximation of $\delta = 0$ gives alternately high and low probabilities for the queue length n. This behavior is due to the transition probabilities, $p(i, j) > 0$ only if $|i - j| = 1$. In Figure 13 the initial queue length is 5, so at time step $\frac{1}{\lambda+\mu}$ only the states $n = 4$ and $n = 6$ have a positive probability ($\pi_{\frac{1}{\lambda+\mu}} = \pi_0 P$). At time $\frac{2}{\lambda+\mu}$ only $n = 3$, $n = 5$ and $n = 7$ have a positive probability, $\pi_{\frac{2}{\lambda+\mu}} = \pi_0 P^2$ and so on. In this way the state distribution is fluctuating between even and odd values until the boundary states $n = 0$ and $n = N$ are reached. If the dummy variable δ increases, then $p(i, i) = \frac{\delta}{\lambda+\mu+\delta}$ increases and the time step decreases, therefore the fluctuations of the distributions decrease. As can be seen in Figure 13 the approximation converges really fast to the theoretical formula when $\delta > 0$. At $t = 1$ only $\delta = 0$ and $\delta = 1$ deviate significantly of the theoretical formula, at $t = 2$ this is only $\delta = 0$.

The probability that a queue is in state n over time is graphically shown in Figure 14. The probabilities are shown every half minute, this is to show the fluctuations. If only the minute probabilities where plotted, we would see a fluent line through the upper points ($n = 1$) or through the lower points ($n = 2$ and $n = 10$) in the case of the rough approximation $\delta = 0$. Also here the approximation converges really fast if δ increases. If $\delta > 0$ there are only small fluctuations from the theoretical formula for low values of t.

The fluctuations of the rough approximation $\delta = 0$ are there because the process can only return to the same state in an even number of steps, except when the process hits state 0 or N. Hence if $\Delta t = 1$ and $\lambda + \mu$ is even then the matrix calculations result in a fluent line and when $\lambda + \mu$ is odd a fluctuating graph is shown. In the case of $\lambda + \mu$ even and an odd initial state, the points in the peaks are calculated if n is odd and if n is even only the bottom points are shown. In

the first case the probability that the queue is in state n is overestimated and in the second case this probability is underestimated.

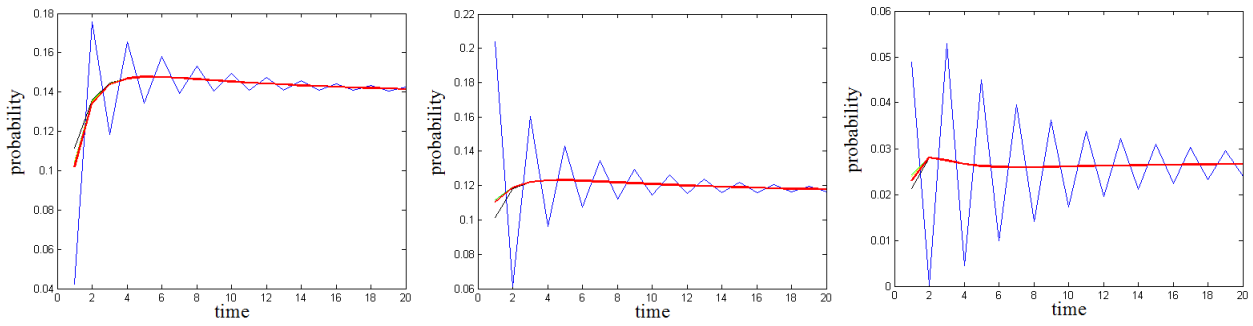


Figure 14: For $\lambda = 10$, $\mu = 12$ and $i = 5$ the probability that the queue is in state $n = 1$(left), $n = 2$(middle) and $n = 10$(right) over time is shown with time step $\Delta t = 0.5$. The red graph is the probability calculated by the theoretical formula and the probabilities calculated by the approximation matrix method are $\delta = 0$ blue graph, $\delta = 1$ black graph, $\delta = 2$ green graph, $\delta = 5$ yellow graph. The latter probabilities coincide with the theoretical graph on large scale and is therefore most of the time not visible.

Since the odd states number are overestimated and the even states are underestimated, this estimation effect is canceled by calculating the mean queue length. Therefore the mean queue length calculated by the approximation method is almost the same as for the theoretical formula for all values of δ, as can be seen in the left graph of Figure 15. In the right figure the absolute difference between the approximated and theoretical value is given for different δ values. In this figure we also see that when δ increases the differences will decrease, so the mean value will converge to the theoretical mean value if $\delta \to \infty$.

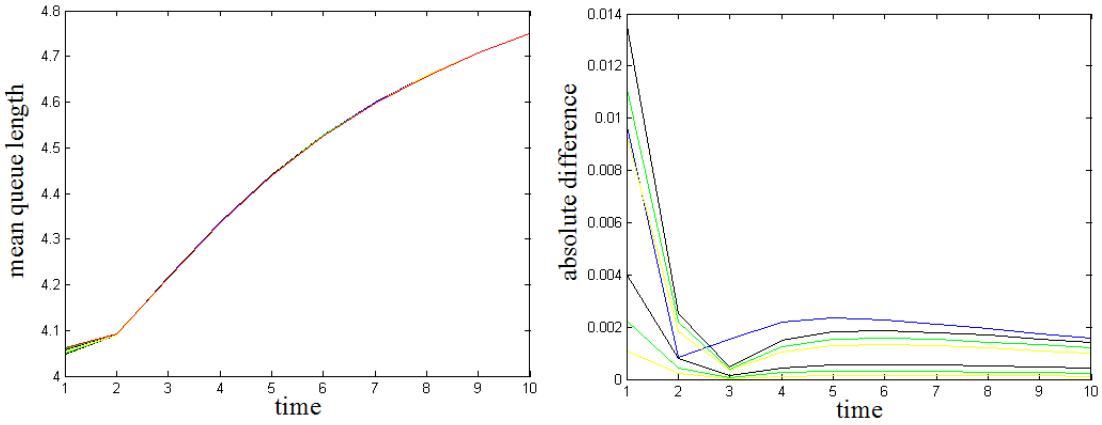


Figure 15: For $\lambda = 10$, $\mu = 12$ and $i = 5$ the mean queue length over time by the theoretical formula (red graph) and the approximation method are shown in the left figure with $\delta = 0$ (blue graph), $\delta = 1$ (black graph), $\delta = 5$ (green graph), $\delta = 10$ (yellow graph), $\delta = 50$ (black graph), $\delta = 100$ (green graph), $\delta = 200$ (yellow graph). The right figure gives the absolute difference of the approximated and theoretical value. In both graphs the time step is $\Delta t = 1$.

Note that for $t < 3$ the rough approximation $\delta = 0$ is closer to the theoretical mean value than the approximations with $\delta = 1, 5, 10$. This is because $\lambda + \mu$ is an even number so the fluctuations of the mean distribution, as shown in the above figures, are not visible. In Figure 16 the absolute differences between the approximation and the theoretical formula are shown in time steps of half a minute.

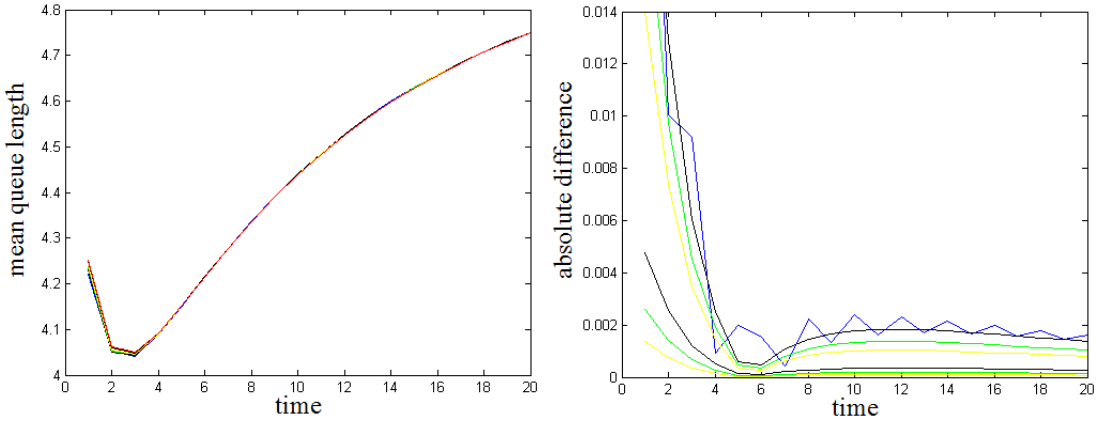


Figure 16: For $\lambda = 10$, $\mu = 12$ and $i = 5$ the mean queue length over time by the theoretical formula (red graph) and the approximation method are shown in the left figure with $\delta = 0$ (blue graph), $\delta = 1$ (black graph), $\delta = 5$ (green graph), $\delta = 10$ (yellow graph), $\delta = 50$ (black graph), $\delta = 100$ (green graph), $\delta = 200$ (yellow graph). The right figure gives the absolute difference of the approximated and theoretical value. In both graphs the time step is $\Delta t = 0.5$.

Calculating the state distribution with states $\chi = \{0, 1, \ldots, 150\}$ by the approximation approach every minute for half an hour has a calculation time of around 1 second. Time-dependent arrival rates will not alter the calculation time of the approximation a lot, since only the λ in the $\overline{P}(\lambda)$ matrix has to be adjusted when λ changes. Therefore the calculation time is slightly higher when we take time-dependent arrival rates, but the computation time of the matrix approach is still really fast. For the same calculation the theoretical Formula (6) has a calculation time of around 20 seconds. Since the approximation method converges quickly to the theoretical distribution when δ increases and the calculation time is much lower the approximation method will be used to calculate the time-dependent queue length distribution.

In order to find the optimal δ value, the distribution calculated by the approximation method is compared to the theoretical distribution for several different values of δ. In Table 3 the total absolute differences between the approximation and theoretical distribution are given for these δ values. The differences are calculated for the distributions at times $t = 1$ and $t = 5$. In Table 4 the total absolute differences between the mean value of the theoretical and approximation method are given. This is calculated by summing the absolute differences between the mean values of the theoretical and approximated method at time $t = 1 - 20$.

In Table 3 we see that the approximation improves extremely by adding $\delta = 1$ as dummy variable. The distribution at time $t = 1$ also improves significantly by increasing $\delta > 1$. If we look at the distribution at a later time, e.g., $t = 5$, the results for increasing δ values still improve but not so

extremely as for $t = 1$. This results match with the figures at the beginning of this paragraph. The differences presented in Table 4 also confirm the results in Figure 15 and 16. The mean is a bit over- or underestimated for small values of δ and t, due to this effect the difference of $\delta = 0$ can be smaller than the difference of $\delta = 1$. The results in Table 4 confirm that if δ increases the approximated mean converges to the theoretical mean.

Distribution	$t = 1$	δ							
μ	λ	0	1	2	5	10	50	100	200
6	2	0.5553	0.0866	0.0773	0.0575	0.0404	0.0119	0.0063	0.0033
6	4	0.7770	0.0826	0.0321	0.0262	0.0195	0.0065	0.0035	0.0019
6	6	0.8858	0.1000	0,0244	0.0210	0.0162	0,0057	0.0031	0.0017
6	8	0.9434	0.1117	0.0298	0.0245	0.0193	0.0072	0.0040	0.0021
12	4	0.1304	0.0393	0.0376	0.0322	0.0262	0.0104	0.0059	0.0032
12	8	0.4648	0.0551	0.0176	0.0150	0.0125	0.0054	0.0031	0.0017
12	12	0.7356	0.0899	0.0126	0.0097	0.0083	0.0038	0.0023	0.0013
12	14	0.8239	0.1026	0.0147	0.0110	0.0095	0.0045	0.0027	0.0015
12	16	0.8856	0.1121	0.0195	0.0158	0.0137	0.0066	0.0040	0.0023
Distribution	$t = 5$								
6	2	0.0022	0.0010	0.0009	0.0007	0.0005	0.0002	0.0001	0.0001
6	4	0.1280	0.0027	0.0025	0.0020	0.0015	0.0005	0.0003	0.0003
6	6	0.5211	0.0022	0.0021	0.0017	0.0013	0.0005	0.0003	0.0001
6	8	0.8253	0.0131	0.0122	0.0103	0.0081	0.0030	0.0017	0.0009
12	4	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
12	8	0.0233	0.0003	0.0003	0.0003	0.0002	0.0001	0.0001	0.0000
12	12	0.3839	0.0017	0.0016	0.0014	0.0012	0.0006	0.0003	0.0002
12	14	0.6344	0.0053	0.0051	0.0046	0.0040	0.0019	0.0011	0.0006
12	16	0.8037	0.0120	0.0116	0.0106	0.0092	0.0044	0.0027	0.0015

Table 3: Total absolute differences between the distribution of the approximation and the theoretical formula for different values of δ at time $t = 1$ and $t = 5$.

For a good approximation the total absolute difference of the distribution and mean should not be too large, but also the calculation time shown in the last row of Table 4 should not be too long. Based on these values we will take $\delta = 50$ in further calculations.

The upper state value, N, also has to be set. The calculation time of the approximation has a computational complexity of $o(N^3)$, so we want to keep N as small as possible. The value of N will depend on the situation, if $\lambda >> \mu$ then N should be larger, since the state N should never be reached. In most cases ρ will be just above 1 and only for a short period. In most calculations in this thesis we will take $N = 150$.

Definition 4.1. *In this thesis the time-dependent queue length distribution will be calculated by the matrix approximation with* $\delta = 50$ *and if not stated otherwise* $N = 150$.

Mean	$t = 1 - 20$	δ							
μ	λ	0	1	2	5	10	50	100	200
6	2	0.2619	0.2624	0.2334	0.1780	0.1275	0.0390	0.0209	0.0108
6	4	0.1245	0.1366	0.1242	0.0991	0.0741	0.0246	0.0134	0.0068
6	6	0.1140	0.1036	0.0958	0.0788	0,0608	0.0215	0.0119	0.0063
6	8	0.0289	0.0265	0.0247	0.0208	0.0164	0.0061	0.0034	0.0018
12	4	0.0779	0.0803	0.0755	0.0651	0.0529	0.0212	0.0121	0.0065
12	8	0.0491	0.0563	0.0531	0.0468	0.0390	0.0168	0.0098	0.0053
12	12	0.1002	0.0945	0.0906	0.0812	0.0693	0.0318	0.0190	0.0105
12	14	0.0517	0.0465	0.0450	0.0406	0.0350	0.0165	0.0100	0.0056
12	16	0.0268	0.0246	0.0239	0.0218	0.0190	0.0093	0.0057	0.0032
Calculation	Time	0.7366	0.5573	0.4702	0.5907	0.5525	0.6718	0.7432	2.2719

Table 4: Total absolute differences between the mean queue length of the approximated and theoretical formula for $t = 1 - 20$ and different values of δ.

5 Departure advice for a single app user

Using the approximation from the previous section the time-dependent queue length distribution is known, so now the waiting time for an arriving customer can be calculated. Since the 'service' time is $Exponential(\mu)$ distributed, the mean 'service' time of a traveller is $\frac{1}{\mu}$. Let $L^a(t)$ be the queue length on arrival at time t, then the expected waiting time is given by $\frac{E(L^a(t))+1}{\mu}$. Because the arrival process is Poisson distributed, the PASTA property[5] can be applied. Hence the state probabilities of the queue length on arrival are the same as the state probabilities, $P(L^a(t) = n) = \pi_t(n)$ for $\forall\ 0 \leq n \leq N$.

Suppose that at arrival there are n vehicles in the queue, then the waiting time is $Erlang(n+1, \mu)$ distributed, since the service times are independent $Exponential(\mu)$ distributed. Let B_k be the exponential service time of vehicle k in the queue, then [6]

$$P\left(\sum_{k=1}^{n+1} B_k > \tau\right) = \sum_{k=0}^{n} \left(\frac{(\mu\tau)^k}{k!} e^{-\mu\tau}\right)$$

Using this the waiting time distribution $W(t)$ can be calculated by:

$$P(W(t) > \tau) = P\left(\sum_{k=1}^{L^a(t)+1} B_k > \tau\right) = \sum_{n=0}^{N} P\left(\sum_{k=1}^{n+1} B_k > \tau\right) P(L^a(t) = n) = \sum_{n=0}^{N}\sum_{k=0}^{n} \left(\frac{(\mu\tau)^k}{k!} e^{-\mu\tau}\right) \pi_t(n)$$

If $P(W(t) > \tau) < \alpha$ then at time t the waiting time is smaller or equal to τ with probability $1 - \alpha$. Let $\tau_\alpha(t)$ be the $1 - \alpha$ upper confidence limit of the waiting time at time t, i.e., $\tau_\alpha(t) = \min(\tau : P(W(t) > \tau) < \alpha)$. Let D be the deadline to be out of the queue, such that the user arrives on time at the destination. Then the latest arrival time such that the app user is on time out of the queue with confidence level $1 - \alpha$ is given by $T^a = \max(t : t + \tau_\alpha(t) \leq D)$. Hence the app user should arrive before T^a at the traffic light.

There is also an easier and faster way to calculate the latest arrival time which gives almost the same result. In reality the outflow of cars from a queue at a traffic light is approximately deterministic with a maximum of μ cars per minute. Let $n_\alpha(t) \in \mathbb{N}$ be the minimum number of vehicles in the queue at time t such that $P(L^a(t) > n_\alpha(t)) < \alpha$. Then with probability $1 - \alpha$ the number of vehicles in the queue at time t is lower than $n_\alpha(t)$ and the expected waiting time of an arriving vehicle is lower than $\frac{1}{\mu}(n_\alpha(t) + 1)$. At every time interval Δt the queue length distribution is calculated, from where we find $n_\alpha(t)$. The points $n_\alpha(t) + 1$ are drawn in Figure 17 for two different λ-values.

To calculate the latest arrival time, such that the app user is with a pre-set confidence level on time out of the queue, we have to solve the following equation:

$$t + \frac{1}{\mu}(n_\alpha(t) + 1) = D \Rightarrow \mu t + (n_\alpha(t) + 1) = \mu D \Rightarrow n_\alpha(t) + 1 = -\mu t + \mu D$$

[5] The fraction of travellers that find the system on arrival in state X is exactly the same as the fraction of time the system is in state X. This property only holds for Poisson arrivals.[1]

[6] Since at time τ maximum n users may depart from the traffic light this can also be seen by $P\left(\sum_{k=1}^{n+1} B_k > \tau\right) = \sum_{k=0}^{n} Pois(k, \mu\tau) = \sum_{k=0}^{n} \left(\frac{(\mu\tau)^k}{k!} e^{-\mu\tau}\right)$.

Hence the latest arrival time $\overline{T}^a$ can be calculated by the intersection of the graph $n_\alpha(t)+1$ and $y=-\mu t+\mu D$, as shown in Figure 17. This is a very fast method since we only have to calculate the intersection.

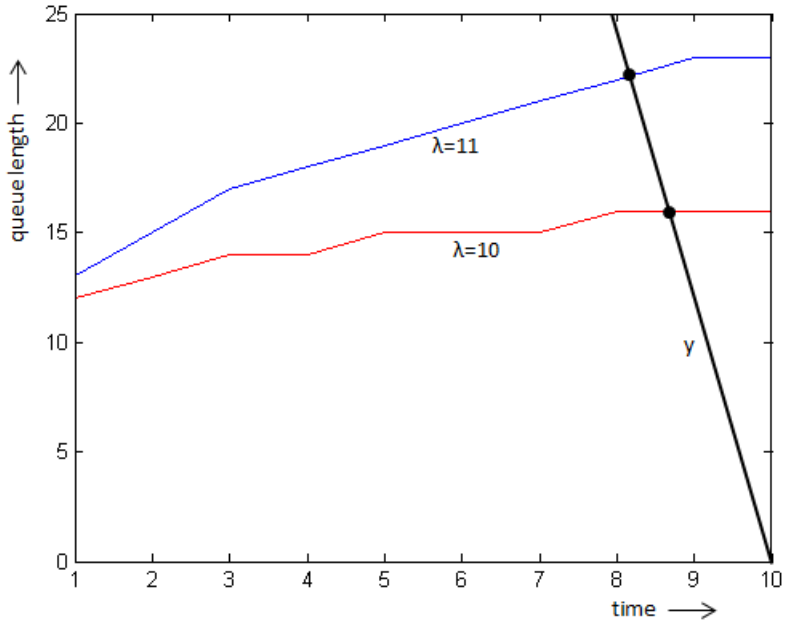


Figure 17: $n_\alpha(t)+1$ is shown for $\lambda=10$ (red) and $\lambda=11$ (blue) with $\mu=12$ and 5 vehicles in the initial state. The intersection with y gives the latest arrival time. Since $n_\alpha(t)=\min(n_\alpha\in\mathbb{N}: P(L^a(t)>n_\alpha)<\alpha)$ is an integer the graph is not fluent.

The solution space of $T^a=\max(t: t+\tau_\alpha(t)<D)$ with $\tau_\alpha(t)=\min(\tau: P(W(t)>\tau)<\alpha)$ is a subset of $\mathbb{R}$. Since the state distribution is calculated at discrete times we only have $\pi_t(n)$ at $t\in\mathbb{N}$. It is hard to compare the latest arrival time T^a with the intersection latest arrival time $\overline{T}^a$ when T^a can only be an integer. Therefore $t+\tau_\alpha(t)$ is calculated for all t values with time step 0.1. The state distributions $\pi_{t,i}(n)$ for $1\le i\le 9$ are calculated by interpolation of the two neighboring state distributions: $\pi_{t,i}=\frac{i}{10}\pi_t+\frac{10-i}{10}\pi_{t+1}$ for $1\le i\le 9$.

	Arrival time		Computational time (sec)	
λ	T^a	$\overline{T}^a$	T^a	$\overline{T}^a$
4	9.6	9.7500	47.50	0.3168
8	9.2	9.3333	71.14	0.3409
12	7.4	7.5714	126.44	0.3723
16	5.6	5.8333	196.85	0.2908
20	4.4	4.6957	415.98	0.2758

Table 5: The latest arrival times and computation times of the theoretical formula (T^a) and the intersection approach ($\overline{T}^a$) for different λ values, with $\mu=12$, $D=10$ and initial state 5.

As can be seen in Table 5 the latest arrival times of both approaches are very close to each other. The calculation of $P(W(t) > \tau)$ is computational intensive as shown in the fourth column of Table 5. In contrary, the calculation time of the intersection approach is really low, since $n_\alpha(t)$ follows directly from π_t. The computational time to calculate $n_\alpha(t) + 1$ and the intersection with y is less than 0.4 second, of which most is due to the calculation of π_t, since the intersection between $n_\alpha(t) + 1$ and y is calculated in less than 0.01 second. Therefore the fast intersection method will be used to calculate the latest arrival time. For notational purposes we will, from now on, omit the bar and write T^a for the latest arrival time calculated by the intersection method.

5.1 Route and departure advice

The only step left is to decide which route the app user has to take, via traffic light $1, 2, \ldots, S-1$ or S. The app should advise the user to take the route with the latest departure time at the origin. Therefore the advised departure time at the origin will be

$$\text{Departure Time} = \max(T_1^a - d_1, T_2^a - d_2, \ldots, T_S^a - d_S) \tag{8}$$

with T_i^a the latest arrival time at the queue at traffic light i and d_i the travel time from the origin to the queue at traffic light i.

The model for a single app user works as follows:
The app distracts the preferred arrival time at the destination from the agenda. The travel time from the traffic light to the destination is calculated using a travel time forecasting model. The departure time D from the intersection is based on the desired arrival time at the destination. This departure time is the deadline at the traffic lights, i.e., the time that the user has to be out of the queue. With this deadline the latest arrival time T^a can be calculated by the intersection method. The travel time from the origin to the queue is given by d, hence $T^a - d$ is the latest departure time from the origin.
These calculations can be done for the routes via traffic lights $1, 2, \ldots, S$. With (8) a decision is made about the route and departure time of the user, which will correspond to the route with the shortest travel time.

6 Schedule multiple app users

In the previous section the optimal arrival time for a single app user was calculated. When only a few travellers use the app, this method works well. Hence, in the starting period of the app in IJburg these simple calculations can be used. When many citizens use the app, the app can not only give the optimal departure time but the users can also be distributed in a smart way over the time slots. In this section we will focus on this last aspect of distributing the users over time.

6.1 Multiple app users with the same deadline

In the previous section a fast approach is given to calculate the latest arrival time at the queue such that the user is with a pre-set confidence level on time out of the queue. This method can only be applied to one or a few app users, because scheduling a second app user with the same deadline will influence the waiting time of the first scheduled app user etc. However, the general idea still holds: in the case of multiple app users we also want that the app users are with a preset confidence level of $1-\alpha$ on time out of the queue and that the total waiting time is as small as possible.

The calculation of the time-dependent state distribution is described in Section 4.2.1. With this distribution the $1-\alpha$ upper confidence limit of the queue length is calculated at every time step Δt. With the intersection approach as shown in Figure 17, the latest arrival time of the app users can be calculated. With some small adjustments this method can still be used in the general setting of multiple app users to ensure that all users will be on time with confidence level $1-\alpha$. In Section 8.1 we will focus on this aspect.

Now that the approach to find the latest arrival time is known we can focus on the main problem: how to schedule the app users?
It is not reasonable to assume that an app user will arrive at exactly the advised time. Therefore we assume that an app user is able to arrive in a time interval of length Δx. For simplicity we will take $\Delta x = \Delta t$, so equal to the time steps in which the queue length distribution is calculated. For every interval of length $\Delta x = \Delta t$, the state distribution is calculated by the mean of the two boundary distributions.

Let ξ_i be the fixed inflow rate in time in vehicles per minute, i.e., the rate of inflow of the vehicles that cannot be scheduled, that do not use the app. The value of ξ_i changes every ΔT minutes, in most traffic management books the flow changes every 10 or 15 minutes. In IJburg the flows are available per quarter, so we set $\Delta T = 15$.
App users are scheduled in intervals of length Δx, let η_i be the number of app users that are scheduled in interval i. Then the inflow of app users per minute in an interval is given by $\frac{\eta_1}{\Delta x}$ and the total inflow rate at interval i is $\lambda_i = \xi_i + \frac{\eta_i}{\Delta x}$ vehicles per minute. Hence the value of λ_i is different for all intervals, let $\lambda = (\lambda_1, \lambda_2, ..., \lambda_n)$ where n is the number of intervals.

Since it is our goal to spread the inflow over the capacity, we do not want to put all app users in the same interval. Therefore we take an interval of length ΔL to schedule the app users with the same deadline, with $\Delta L = l\Delta x$ and $l \in \mathbb{N}_{>0}$. l can be made inflow dependent: l should be larger when the inflow is high and smaller for a low inflow. For now $\Delta L = l\Delta x$ will be a fixed value.

Let M_D be the number of app users which have D as their deadline, corresponding to deadline D there is a latest arrival time T_D^a. Let A_D be the latest arrival interval corresponding to this latest arrival time. The latest arrival interval A_D is calculated by $A_D = \lfloor \frac{T_D^a}{\Delta x} \rfloor$, all users with deadline D that are scheduled before or at this interval will be with a pre-set confidence level on time out of the queue. The M_D app users with latest arrival interval A_D should be divided in l groups, which can be scheduled in the intervals $\{A_D - l + 1, A_D - l + 2, \ldots, A_D\}$.

Our goal is to make the total waiting time of the app users as small as possible. The value function V of a schedule with $\eta_1, \eta_2, \ldots, \eta_n$ users in the intervals $1, 2, \ldots, n$, is given below. Here $L_i(x, \lambda)$ is the mean queue length at time $i\Delta x$ when starting in initial state x, with $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n)$ and $\lambda_i = \xi_i + \frac{\eta_i}{\Delta x}$. Therefore $\frac{1}{2}(L_{i-1}(x, \lambda) + L_i(x, \lambda))$ is the mean queue length in interval i. Hence the value function is the sum of the waiting times of all app users. The goal is to find the schedule $(\eta_1, \eta_2, \ldots, \eta_n)$ that minimizes this value function.

$$V(x, \eta_1, \eta_2, \ldots, \eta_n) = \sum_{i=1}^{n} \eta_i \frac{1}{\mu} \frac{L_{i-1}(x, \lambda) + L_i(x, \lambda)}{2}$$

$$\text{with} \sum_{i=A_D+l-1}^{A_D} \eta_i = M_D$$

$$\eta_i \geq 0 \text{ for } i = 1, 2, \ldots, l$$

The first constraint ensures that all app users with latest arrival interval A_D are scheduled in the intervals $\{A_D - l + 1, A_D - l + 2, \ldots, A_D\}$.

Minimizing the value function is just a linear programming problem which can be solved easily. In Table 6 the optimal schedules are given for 20 app users with latest arrival interval $A_D = 4$, $l = 4$ and for different ξ values.

$\mu = 12$					Mean queue length				
ξ	η_1	η_2	η_3	η_4	$t = 1$	$t = 2$	$t = 3$	$t = 4$	Value
0	5	5	4	6	0.69	0.71	0.51	0.95	1.00
2	6	4	4	6	1.56	1.07	1.01	1.69	1.85
4	6	4	3	7	2.39	2.07	1.59	3.42	3.26
6	6	3	3	8	3.61	3.14	3.03	6.43	5.92
8	6	3	2	9	5.35	5.63	5.26	10.88	10.17
10	4	3	2	11	7.39	8.97	9.60	18.76	18.65
11	0	4	1	15	10.04	13.22	13.47	27.51	30.60
12	20	0	0	0	30.00	30.00	30.00	30.01	33.33
14	20	0	0	0	32.00	34.00	36.00	38.00	35.00

Table 6: The optimal schedule is given for 20 users with $A_D = l = 4$ for different values of ξ. The corresponding mean queue lengths at the four measure points and the minimum values are also shown. The initial value is $\frac{\rho}{1-\rho}$ with $\rho = \xi/\mu$ if $\mu > \xi$ otherwise the initial value is 10.

If ξ is small compared to μ the app users are almost evenly divided over the intervals. In general more users are scheduled in the last interval. This is optimal since users in the last interval do not influence the waiting times of users in previous intervals. If more users are scheduled in for example the first interval this increases the queue length in the following intervals. Note that more people are scheduled in the first interval due to the relative low value of the initial state.
All the users are scheduled in the first interval if $\xi \geq 12$, this is due to the fact that $\rho \geq 1$ even without the 20 app users. In this case the queue length increases over time, so it is optimal to join the queue as fast as possible.

In the case of one deadline the scheduling problem is easily solved by linear programming, but when the number of intervals or the number of app users increase the computational time will increase as well. Due to the long computation time LP is not an option in the case of multiple users and multiple deadlines, so we need a different approach to find an optimal schedule. In Subsection 6.3 we will show that the value function is convex for an M/M/1 queue, so a local search method will give the optimal schedule. In Figure 18 the value function is shown for $\xi = 8$, $\mu = 12$ and $l = A_D = 3$.

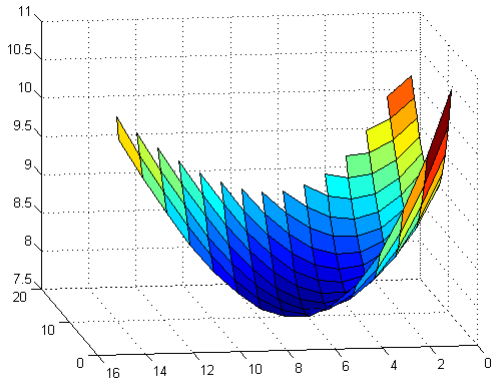


Figure 18: The value function for (η_1, η_2, η_3) is shown with η_1 on the x axis and η_2 on the y axis, with $\mu = 12$, $\xi = 8$ and $M_D = 15$.

Before we will prove that the value function is convex, the value function for multiple users with multiple deadlines is presented.

6.2 Multiple app users with different deadlines

The case of multiple deadlines, is very similar to the case of multiple users with the same deadline. All the parameters defined in the previous section also hold in this case.

The goal is to make the total waiting time of the app users as small as possible. Suppose that $(\eta_1, \eta_2, \ldots, \eta_n)$ is a schedule of the app users in the intervals $1, 2, \ldots, n$ respectively. The value function is the sum of the expected waiting times of all app users:

$$V(x, \eta_1, \eta_2, \ldots, \eta_n) = \sum_{i=1}^{n} \eta_i \frac{L_{i-1}(x, \lambda) + L_i(x, \lambda)}{2\mu}$$

The goal is to find the schedule $(\eta_1, \eta_2, \ldots, \eta_n)$ that minimizes this value function. The optimal schedule must satisfy a couple of constraints that are specified below. A user with deadline D, must be scheduled before or at latest arrival interval $A_D = \lfloor \frac{T_D^a}{\Delta x} \rfloor$ such that the user is on time out of the queue. Since a user does not want to arrive too early at the destination, a user must not be scheduled too early. Therefore a vehicle can be scheduled in the l intervals $\{A_D - l + 1, A_D - l + 2, \ldots, A_D\}$. Let N_i be the number of app users with latest arrival interval i. $N_i = 0$ for $i < l$ since there are no l intervals before i to schedule the user in.[7] Therefore the schedule must satisfy the following constraints:

$$\sum_{j=1}^{i} \eta_j \geq \sum_{k=1}^{i} N_k \text{ for } \forall i \in \{l, \ldots, n\}$$
$$\sum_{j=1}^{i-l+1} \eta_j \leq \sum_{k=1}^{i} N_k \text{ for } \forall i \in \{l, \ldots, n\}$$

The first constraint ensures that enough users are scheduled in the intervals before or at the latest arrival intervals, i.e., that all users are on time. We do not want to schedule persons too early, so by adding the second constraint all users are scheduled in the l intervals before or at their latest arrival interval. The last constraint can be dropped, if we do not mind if users are scheduled earlier than l intervals before their arrival deadline.

$$\left.\begin{aligned} \eta_1 + \ldots + \eta_i &\geq N_l + \ldots + N_i \\ \eta_1 + \ldots + \eta_{i-l} &\leq N_l + \ldots + N_{i-1} \end{aligned}\right\} \Rightarrow \eta_{i-l+1} + \ldots + \eta_i \geq N_i$$

Hence, the two constraints ensure that there are enough users scheduled in the intervals $i-l+1, \ldots, i$.

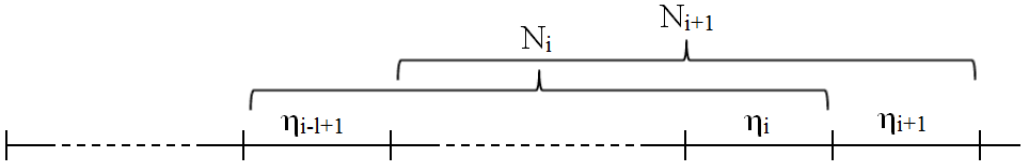


Figure 19: Graphical representation of the intervals in which users with latest arrival interval i and $i + 1$ can be scheduled.

[7] Of course a user can have an arrival interval $i < l$, but then the user would have been scheduled in previous iterations, so we do not take them into account anymore. This aspect will be discussed in Section 8

Definition 6.1. *The optimization problem that has to be solved for multiple app users and multiple deadlines is given by:*

$$\begin{aligned} \underset{\eta}{\textit{minimize}} \quad & V(x,\eta) = \sum_{i=1}^{n} \eta_i \frac{L_{i-1}(x,\lambda) + L_i(x,\lambda)}{2\mu} \\ \textit{subject to} \quad & \sum_{j=1}^{i} \eta_j \geq \sum_{k=1}^{i} N_k \textit{ for } \forall i \in \{l,\ldots,n\} \\ & \sum_{j=1}^{i-l+1} \eta_j \leq \sum_{k=1}^{i} N_k \textit{ for } \forall i \in \{l,\ldots,n\} \\ & \eta_i \geq 0 \textit{ for } i = 1,2,\ldots,n. \end{aligned}$$

With $\eta = (\eta_1, \eta_2, \ldots, \eta_n)$ *and* $\lambda = \xi + \frac{\eta}{\Delta x}$.

The constraints are linear in η_i for $\forall\ 1 \leq i \leq n$, so the solution space is convex in $\eta = (\eta_1, \eta_2, \ldots, \eta_n)$. To prove that the optimization problem is convex, we only have to show that the value function is convex in η_i for $\forall\ 1 \leq i \leq n$. Note that $L_i(x,\lambda)$ depends on η, since $\lambda = \xi + \frac{\eta}{\Delta x}$.

6.3 Proof convexity value function

In this subsection it will be proven that the value function given in Definition 6.1 is convex in $\eta = (\eta_1, \eta_2, \ldots, \eta_n)$. First, it will be shown that $L(x,\lambda)$ is convex in λ, because then convexity of $L(x,\lambda)$ in η is trivial since $\lambda = \xi + \frac{\eta}{\Delta x}$.

First, we have to look at the process as described by the approximation method in Subsection 4.2.1. Every time step the state of the process increases by one with rate λ and decreases by one with rate μ, the process stays in the current state with rate δ. To show convexity the λ value will be adjusted. The time steps in the approximation method are given by $\frac{1}{\lambda+\mu+\delta} = \frac{1}{C}$, hence if λ increases the time step will decrease and vice versa. To calculate the distribution at time interval Δt more iterations (time steps) are needed when the λ value increases. For convenience we want that the number of iterations are fixed for all λ values, i.e., the time step $\frac{1}{\lambda+\mu+\delta}$ should be fixed for $\forall\lambda$. The dummy variable δ can be used to accomplish the fixed time steps: If $\lambda \to \lambda + 1$ then $\delta \to \delta - 1$ and vice versa if $\lambda \to \lambda - 1$ then $\delta \to \delta + 1$. This can only be done if $\delta > \lambda$ for all values of λ. It is plausible that λ has an upper value λ_{max}, so the only assumption we have to make is that $\delta > \lambda_{max}$. Since we have chosen $\delta = 50$ the assumption is satisfied in the IJburg case, $\lambda_{max} < 50$ in IJburg [6], otherwise δ should be chosen such that this condition holds.

We will show that for an M/M/1 queue the mean queue length in time is convex in $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n)$. Let $L_t(x,\lambda)$ be the mean queue length at time step t starting with queue length x at time $t = 0$, here the time step is $\frac{1}{\lambda+\mu+\delta} = \frac{1}{C}$. Since users are scheduled in intervals of length Δx, every time slot of length $\Delta x = 1$ has its own λ value, i.e., every C time steps the λ value changes[8]. Therefore we will often write $t = kC + \tau$ with $k \in \mathbb{N}$ and $0 \leq \tau < C$.

[8] In the proof we take $\Delta x = 1$, but for $\Delta x > 1$ the proof is almost exactly the same.

An example of the λ values for $X = \alpha C + \beta$ with $0 \leq \beta < C$ and $\alpha = \lfloor \frac{X}{C} \rfloor$ is given in Figure 20. Note that the steps shown in this picture consist of C time steps of length $\frac{1}{C}$, except the first step from 0 to $X - \alpha C$ which consists of β time steps.

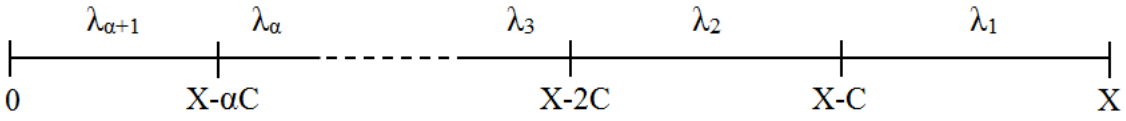


Figure 20: Intervals with corresponding λ values for $X = \alpha C + \beta$.

Corollary 6.2. *$L_{kC+\tau}(x, \lambda)$ with $1 \leq \tau < C$ and $k, x \in \mathbb{N}$ depends on all λ_r with $1 \leq r \leq k+1$ and is independent of λ_r with $r > k+1$. $L_{kC}(x, \lambda)$ with $k, x \in \mathbb{N}$ depends only on λ_r with $1 \leq r \leq k$.*

This can be easily seen by Figure 20.

The value of $L_{t+1}(x, \lambda)$ can recursively be found from $L_t(x, \lambda)$ in the following way:

$$\begin{aligned}
&L_0(x, \lambda) = x \\
&L_{kC}(0, \lambda) = \lambda_k L_{kC-1}(1, \lambda) + (C - \lambda_k) L_{kC-1}(0, \lambda) \\
&L_{kC}(x, \lambda) = \lambda_k L_{kC-1}(x+1, \lambda) + (C - \lambda_k - \mu) L_{kC-1}(x, \lambda) + \mu L_{kC-1}(x-1, \lambda) \\
&L_{kC+\tau}(0, \lambda) = \lambda_{k+1} L_{kC+\tau-1}(1, \lambda) + (C - \lambda_{k+1}) L_{kC+\tau-1}(0, \lambda) \\
&L_{kC+\tau}(x, \lambda) = \lambda_{k+1} L_{kC+\tau-1}(x+1, \lambda) + (C - \lambda_{k+1} - \mu) L_{kC+\tau-1}(x, \lambda) + \mu L_{kC+\tau-1}(x-1, \lambda)
\end{aligned} \tag{9}$$

with $k \in \mathbb{N}$, $x \in \mathbb{N}_{>0}$ and $1 \leq \tau < C$.
Note that the process $L_t(x, \lambda)$ can never be larger than $L_t(x+1, \lambda)$, i.e., the same process with higher initial state has a higher mean queue length, therefore

$$L_t(x+1, \lambda) - L_t(x, \lambda) > 0 \text{ for } \forall x, t \in \mathbb{N} \tag{10}$$

To prove that $L_t(x, \lambda)$ is convex in λ, we need the following two Lemmas 6.3 and 6.4. The first lemma states that the mean queue length in time is convex in the initial state x.

Lemma 6.3. *$L_t(x+1, \lambda) - 2L_t(x, \lambda) + L_t(x-1, \lambda) \geq 0$ for $\forall t \in \mathbb{N}$ and $\forall x \in \mathbb{N}_{>0}$.*

Proof. We will prove this lemma by induction on t. For $t = 0$ and $t = 1$ the lemma holds:

For $t = 0$ and $\forall x \geq 1$

$$L_0(x+1, \lambda) - 2L_0(x, \lambda) + L_0(x-1, \lambda) = x + 1 - 2x + x - 1 = 0$$

For $t = 1$ and $\forall x > 1$, by the recursive formulas (9) we find:

$$\left.\begin{aligned}
&L_1(0, \lambda) = \lambda_1 L_0(1, \lambda) + (C - \lambda_1) L_0(0, \lambda) = \lambda_1 \\
&L_1(x, \lambda) = \lambda_1 L_0(x+1, \lambda) + (C - \lambda_1 - \mu) L_0(x, \lambda) + \mu L_0(x-1, \lambda) = Cx - \mu + \lambda_1
\end{aligned}\right\} \Rightarrow$$

$$\begin{aligned}
&L_1(2, \lambda) - 2L_1(1, \lambda) + L_1(0, \lambda) = \mu. \\
&L_1(x+1, \lambda) - 2L_1(x, \lambda) + L_1(x-1, \lambda) = 0
\end{aligned}$$

Suppose the lemma holds for $t = kC + \tau$ with $0 \leq \tau < C$. We will now show that it also holds for $t + 1$, by using the recursive formulas (9) an rearrangement of the terms.

For $\forall x > 1$

$$\begin{aligned}
&L_{t+1}(x+1,\lambda) - 2L_{t+1}(x,\lambda) + L_{t+1}(x-1,\lambda) = \\
&\quad = \lambda_{k+1}L_t(x+2,\lambda) + (C - \lambda_{k+1} - \mu)L_t(x+1,\lambda) + \mu L_t(x,\lambda) \\
&\quad - 2(\lambda_{k+1}L_t(x+1,\lambda) - 2(C - \lambda_{k+1} - \mu)L_t(x,\lambda) - 2\mu L_t(x-1,\lambda) \\
&\quad + \lambda_{k+1}L_t(x,\lambda) + (C - \lambda_{k+1} - \mu)L_t(x-1,\lambda) + \mu L_t(x-2,\lambda) \\
&\quad = \lambda_{k+1}(L_t(x+2,\lambda) - 2L_t(x+1,\lambda) + L_t(x,\lambda)) \\
&\quad + (C - \lambda_{k+1} - \mu)(L_t(x+1,\lambda) - 2L_t(x,\lambda) + L_t(x-1,\lambda)) \\
&\quad + \mu(L_t(x,\lambda) - 2L_t(x-1,\lambda) + L_t(x-2,\lambda)) \geq 0
\end{aligned}$$

Since we assumed that the lemma holds for t and $\forall x \in N_{>0}$, the lemma is also true for $t + 1$ and $\forall x \in N_{>1}$. The proof for $x = 1$ is almost identical to the proof for $x > 1$, therefore it is omitted but for completeness it can be found in Appendix A. Hence we have proven that the lemma holds for $\forall x \in \mathbb{N}_{>0}$ at $t + 1$, so by induction on t the lemma is true for $\forall t \in \mathbb{N}$ and $\forall x \in \mathbb{N}_{>0}$. □

With Lemma 6.3 we have proven that the mean queue length in time is convex in the initial value x. We will use this lemma to prove that the partial derivative of the mean queue length $L_t(x, \lambda)$ to λ is increasing in the initial state x.

Lemma 6.4. $\frac{\partial}{\partial\lambda_r}L_t(x+1,\lambda) - \frac{\partial}{\partial\lambda_r}L_t(x,\lambda) \geq 0$ *for* $\forall x, t \in \mathbb{N}$ *and* $\forall 1 \leq r \leq n$.

Proof. We will prove this lemma by induction on t. For $t = 0$ and $t = 1$ the lemma is true, since L_0 does not depend on λ and $L_1(x, \lambda)$ only depends on λ_1, so $\forall x \in \mathbb{N}$:

$$\frac{\partial}{\partial\lambda_r}L_0(x,\lambda) = 0 \text{ for } \forall 1 \leq r \leq n \tag{11}$$

$$\frac{\partial}{\partial\lambda_r}L_1(x,\lambda) = \begin{cases} 1 & \text{if } r = 1 \\ 0 & \text{otherwise} \end{cases}$$

Suppose that the lemma holds for $t = kC + \tau$ with $k \in \mathbb{N}$ and $0 \leq \tau < C$:

$$\frac{\partial}{\partial\lambda_r}L_t(x+1,\lambda) - \frac{\partial}{\partial\lambda_r}L_t(x,\lambda) \geq 0 \text{ for } \forall x \in \mathbb{N} \text{ and } \forall 1 \leq r \leq n \tag{12}$$

We will show that the theorem also holds for $t + 1$ and $\forall x > 0$. The proof for initial state $x = 0$ is similar to the proof for $x > 0$ and can be found in Appendix A.

By Corollary 6.2 $L_{t+1}(x, \lambda)$ does not depend on λ_r with $r > k + 1$, i.e., $\frac{\partial}{\partial\lambda_r}L_{t+1}(x,\lambda) = 0$ for $r > k + 1$, so the lemma holds for $r > k + 1$. We now only have to show that the lemma also holds for $r \leq k + 1$.

First we will look at $r = k + 1$ for which two cases are distinguished:
Case 1: $1 \leq \tau < C$ then $1 < \tau + 1 \leq C$, so $L_{kC+\tau} = L_t$ and $L_{kC+\tau+1} = L_{t+1}$ depend on the same λ values, λ_r with $r \leq k + 1$ (see Corollary 6.2).
Case 2: $\tau = 0$ then $\tau + 1 = 1$, here $L_{kC} = L_t$ depends on $\lambda_1, \ldots, \lambda_k$ and $L_{kC+1} = L_{t+1}$ depends on $\lambda_1, \ldots, \lambda_k, \lambda_{k+1}$, so the derivative of L_t to λ_{k+1} is zero. In this case the computations can be simplified.

We will first prove the lemma for case 1 of $r = k + 1$ by filling in the recursive formulas (9).

For $\forall x > 0$

Case 1: $t = kC + \tau$ with $1 \leq \tau < C$ and $k \in \mathbb{N}$

$$\begin{aligned}
&\frac{\partial}{\partial \lambda_{k+1}} L_{t+1}(x+1,\lambda) - \frac{\partial}{\partial \lambda_{k+1}} L_{t+1}(x,\lambda) = \\
&\quad = \frac{\partial}{\partial \lambda_{k+1}} (\lambda_{k+1} L_t(x+2,\lambda) + (C - \lambda_{k+1} - \mu) L_t(x+1,\lambda) + \mu L_t(x,\lambda)) \\
&\quad - \frac{\partial}{\partial \lambda_{k+1}} (\lambda_{k+1} L_t(x+1,\lambda) + (C - \lambda_{k+1} - \mu) L_t(x,\lambda) + \mu L_t(x-1,\lambda))
\end{aligned}$$

By working out the derivatives we get:

$$\begin{aligned}
&= \lambda_{k+1} \frac{\partial}{\partial \lambda_{k+1}} L_t(x+2,\lambda) + (C - \lambda_{k+1} - \mu) \frac{\partial}{\partial \lambda_{k+1}} L_t(x+1,\lambda) + \mu \frac{\partial}{\partial \lambda_{k+1}} L_t(x,\lambda) \\
&+ L_t(x+2,\lambda) - L_t(x+1,\lambda) - \Bigg(\lambda_{k+1} \frac{\partial}{\partial \lambda_{k+1}} L_t(x+1,\lambda) + (C - \lambda_{k+1} - \mu) \frac{\partial}{\partial \lambda_{k+1}} L_t(x,\lambda) \\
&+ \mu \frac{\partial}{\partial \lambda_{k+1}} L_t(x-1,\lambda) + L_t(x+1,\lambda) - L_t(x,\lambda) \Bigg)
\end{aligned}$$

Rearrangement of the terms gives

$$\begin{aligned}
&= \lambda_{k+1} \left(\frac{\partial}{\partial \lambda_{k+1}} L_t(x+2,\lambda) - \frac{\partial}{\partial \lambda_{k+1}} L_t(x+1,\lambda_k) \right) \\
&+ (C - \lambda_{k+1} - \mu) \left(\frac{\partial}{\partial \lambda_{k+1}} L_t(x+1,\lambda_k) - \frac{\partial}{\partial \lambda_{k+1}} L_t(x,\lambda_k) \right) \\
&+ \mu \left(\frac{\partial}{\partial \lambda_{k+1}} L_t(x,\lambda) - \frac{\partial}{\partial \lambda_{k+1}} L_t(x-1,\lambda) \right) + L_t(x+2,\lambda) - 2L_t(x+1,\lambda) + L_t(x,\lambda) \\
&\geq 0
\end{aligned}$$

By the induction hypothesis (12) and Lemma 6.3 the lemma holds for case 1 of $r = k + 1$. For case 2 a similar proof is given below.

Case 2: $t = kC$ with $k \in \mathbb{N}$

$$\begin{aligned}
&\frac{\partial}{\partial\lambda_{k+1}}L_{t+1}(x+1,\lambda) - \frac{\partial}{\partial\lambda_{k+1}}L_{t+1}(x,\lambda) = \\
&\quad = \lambda_{k+1}\frac{\partial}{\partial\lambda_{k+1}}L_t(x+2,\lambda) + (C-\lambda_{k+1}-\mu)\frac{\partial}{\partial\lambda_{k+1}}L_t(x+1,\lambda) + \mu\frac{\partial}{\partial\lambda_{k+1}}L_t(x,\lambda) \\
&\quad + L_t(x+2,\lambda) - L_t(x+1,\lambda) - \Bigg(\lambda_{k+1}\frac{\partial}{\partial\lambda_{k+1}}L_t(x+1,\lambda) + (C-\lambda_{k+1}-\mu)\frac{\partial}{\partial\lambda_{k+1}}L_t(x,\lambda) \\
&\quad + \mu\frac{\partial}{\partial\lambda_{k+1}}L_t(x-1,\lambda) + L_t(x+1,\lambda) - L_t(x,\lambda)\Bigg) \\
&\quad = L_t(x+2,\lambda) - 2L_t(x+1,\lambda) + L_t(x,\lambda) \geq 0
\end{aligned}$$

Since $L_t(x,\lambda) = L_{kC}(x,\lambda)$ does not depend on $\lambda_{k+1} \Rightarrow \frac{\partial}{\partial\lambda_{k+1}}L_t(x,\lambda) = 0$ for $\forall x \in \mathbb{N}$ and by Lemma 6.3 the last inequality holds. Hence the lemma is true for case 2 of $r = k+1$, therefore the lemma holds for $t+1$ with $r = k+1$.

The only step left is to show that the lemma also holds for $r < k+1$:

For $\forall x > 0$

$$\begin{aligned}
&\frac{\partial}{\partial\lambda_{k+1}}L_{t+1}(x+1,\lambda) - \frac{\partial}{\partial\lambda_{k+1}}L_{t+1}(x,\lambda) = \\
&\quad = \lambda_{k+1}\frac{\partial}{\partial\lambda_r}L_t(x+2,\lambda) + (C-\lambda_{k+1}-\mu)\frac{\partial}{\partial\lambda_r}L_t(x+1,\lambda) + \mu\frac{\partial}{\partial\lambda_r}L_t(x,\lambda) \\
&\quad - \left(\lambda_{k+1}\frac{\partial}{\partial\lambda_r}L_t(x+1,\lambda) + (C-\lambda_{k+1}-\mu)\frac{\partial}{\partial\lambda_r}L_t(x,\lambda) + \mu\frac{\partial}{\partial\lambda_r}L_t(x-1,\lambda)\right)
\end{aligned}$$

Rearrangement of the terms gives

$$\begin{aligned}
&= \lambda_{k+1}\left(\frac{\partial}{\partial\lambda_{k+1}}L_t(x+2,\lambda) - \frac{\partial}{\partial\lambda_{k+1}}L_t(x+1,\lambda_k)\right) \\
&+ (C-\lambda_{k+1}-\mu)\left(\frac{\partial}{\partial\lambda_{k+1}}L_t(x+1,\lambda_k) - \frac{\partial}{\partial\lambda_{k+1}}L_t(x,\lambda_k)\right) \\
&+ \mu\left(\frac{\partial}{\partial\lambda_{k+1}}L_t(x,\lambda) - \frac{\partial}{\partial\lambda_{k+1}}L_t(x-1,\lambda)\right)
\end{aligned}$$

Hence by induction hypothesis (12) the lemma is also true for $r < k+1$, therefore the lemma holds for $t+1$. By induction on t Lemma 6.4 holds for all $t, x \in \mathbb{N}$ and $\forall 1 \leq r \leq n$. □

We are now ready to prove that the mean queue length in time is convex in λ. As an addition we will also prove that the queue length is increasing in λ.

Theorem 6.5. *$L_t(x,\lambda)$ is increasing and convex in $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n)$ for $\forall t, x \in \mathbb{N}$*

Proof. By induction on t we will show that this lemma is true. By (11) the theorem holds for $t = 0$ and $t = 1$. Suppose that the theorem holds for $t = kC + \tau$ with $k \in \mathbb{N}$ and $0 \leq \tau < C$:

$$\frac{\partial}{\partial \lambda_r} L_t(x,\lambda) \geq 0 \text{ and } \frac{\partial^2}{\partial \lambda_r^2} L_t(x,\lambda) \geq 0 \text{ for } \forall 1 \leq r \leq n \text{ and } \forall x \in \mathbb{N} \tag{13}$$

We will show that the theorem is also true for $t+1$. The proof is only given for $x \geq 1$, the proof for $x = 0$ is similar and can be found in Appendix A.

By Corollary 6.2, $L_{t+1}(x,\lambda)$ does not depend on λ_r with $r > k+1$, i.e., $\frac{\partial}{\partial \lambda_r} L_{t+1}(x,\lambda) = 0$ for $\forall r > k+1$, so the lemma holds for λ_r with $r > k+1$. Now we only have to show that the lemma holds for $t+1$ with $r \leq k+1$.

For $r = k+1$ the two cases defined in the proof of Lemma 6.4 are again distinguished. We will prove that the lemma holds in both cases by using the recursive formulas (9).

For $x > 0$

Case 1: $t = kC + \tau$ with $1 \leq \tau < C$ and $k \in \mathbb{N}$

$$\begin{aligned}
L_{t+1}(x,\lambda) =& \lambda_{k+1} L_t(x+1,\lambda) + (C - \lambda_{k+1} - \mu) L_t(x,\lambda) + \mu L_t(x-1,\lambda) \\
\frac{\partial}{\partial \lambda_{k+1}} L_{t+1}(x,\lambda) =& \lambda_{k+1} \frac{\partial}{\partial \lambda_{k+1}} L_t(x+1,\lambda) + (C - \lambda_{k+1} - \mu) \frac{\partial}{\partial \lambda_{k+1}} L_t(x,\lambda) + \mu \frac{\partial}{\partial \lambda_{k+1}} L_t(x-1,\lambda) + \\
& L_t(x+1,\lambda) - L_t(x,\lambda) \\
\frac{\partial^2}{\partial \lambda_{k+1}^2} L_{t+1}(x,\lambda) =& \lambda_{k+1} \frac{\partial^2}{\partial \lambda_{k+1}^2} L_t(x+1,\lambda) + (C - \lambda_{k+1} - \mu) \frac{\partial^2}{\partial \lambda_{k+1}^2} L_t(x,\lambda) + \mu \frac{\partial^2}{\partial \lambda_{k+1}^2} L_t(x-1,\lambda) + \\
& 2\frac{\partial}{\partial \lambda_{k+1}} L_t(x+1,\lambda) - 2\frac{\partial}{\partial \lambda_{k+1}} L_t(x,\lambda)
\end{aligned}$$

By the induction hypothesis (13) and inequality (10) $\Rightarrow \frac{\partial}{\partial \lambda_{k+1}} L_{t+1}(x,\lambda) > 0$ for $\forall x \in \mathbb{N}$. Due to assumption (13) and Lemma 6.4 $\Rightarrow \frac{\partial^2}{\partial \lambda_{k+1}^2} L_{t+1}(x,\lambda) \geq 0$ for $\forall x \in \mathbb{N}$. Hence for case 1 of $r = k+1$ the lemma is true. The proof of case 2 is given below.

Case 2 : $t = kC$ with $k \in \mathbb{N}$

$$\begin{aligned}
L_{t+1}(x,\lambda) =& \lambda_{k+1} L_t(x+1,\lambda) + (C - \lambda_{k+1} - \mu) L_t(x,\lambda) + \mu L_t(x-1,\lambda) \\
\frac{\partial}{\partial \lambda_{k+1}} L_{t+1}(x,\lambda) =& L_t(x+1,\lambda) - L_t(x,\lambda) \\
\frac{\partial^2}{\partial \lambda_{k+1}^2} L_{t+1}(x,\lambda) =& 0
\end{aligned}$$

By (10) $L_t(x+1,\lambda) - L_t(x,\lambda) > 0$, hence the lemma also holds for case 2 of $r = k+1$. The only thing left is to show that the theorem also holds for $t+1$ and λ_r with $r < k+1$.

For $r < k+1$ and $x > 0$

$$\begin{aligned}
L_{t+1}(x,\lambda) &= \lambda_{k+1} L_t(x+1,\lambda) + (C - \lambda_{k+1} - \mu) L_t(x,\lambda) + \mu L_t(x-1,\lambda) \\
\frac{\partial}{\partial \lambda_r} L_{t+1}(x,\lambda) &= \lambda_{k+1} \frac{\partial}{\partial \lambda_r} L_t(x+1,\lambda) + (C - \lambda_{k+1} - \mu) \frac{\partial}{\partial \lambda_r} L_t(x,\lambda) + \mu \frac{\partial}{\partial \lambda_r} L_t(x-1,\lambda) \\
\frac{\partial^2}{\partial \lambda_r^2} L_{t+1}(x,\lambda) &= \lambda_{k+1} \frac{\partial^2}{\partial \lambda_r^2} L_t(x+1,\lambda) + (C - \lambda_{k+1} - \mu) \frac{\partial^2}{\partial \lambda_r^2} L_t(x,\lambda) + \mu \frac{\partial^2}{\partial \lambda_r^2} L_t(x-1,\lambda)
\end{aligned}$$

By assumption (13) $\Rightarrow \frac{\partial}{\partial \lambda_r} L_{t+1}(x,\lambda) \geq 0$ and $\frac{\partial^2}{\partial \lambda_r^2} L_{t+1}(x,\lambda) \geq 0$ for $\forall 1 \leq r \leq n$. Therefore the theorem also holds for $t+1$ and λ_r with $r < k+1$.
Hence the theorem is true for $t+1$ and by induction Theorem 6.5 holds for $\forall t, x \in \mathbb{N}$. □

Now we are ready to prove that the mean queue length is convex in $\eta = (\eta_1, \eta_2, \ldots, \eta_n)$. Since by Theorem 6.5 the mean queue length is convex in λ and $\lambda_k = \xi_k + \frac{\eta_k}{\Delta x}$ is linear in η_k for $\forall k \in \mathbb{N}$, the following lemma proves that $L_t(x,\lambda)$ is convex in η.

Lemma 6.6. *If g is convex and h a linear function then $g(h(x))$ is convex.*

Proof. Suppose g is convex, i.e., $\alpha g(x) + (1-\alpha) g(y) \geq g(\alpha x + (1-\alpha) y)$
and h is linear, i.e., $\alpha h(x) + (1-\alpha) h(y) = h(\alpha x + (1-\alpha) y)$,
then $\alpha g(h(x)) + (1-\alpha) g(h(y)) \geq g(\alpha h(x) + (1-\alpha) h(y)) = g(h(\alpha x + (1-\alpha) y))$ □

In the same way it can be shown that since $L_t(x,\lambda)$ is increasing in λ and $\lambda = \xi + \frac{\eta}{\Delta x}$ is increasing in η, $L_t(x,\lambda)$ is increasing in η.

Theorem 6.7. *The mean queue length $L_t(x,\lambda)$ is convex and increasing in $\eta = (\eta_1, \eta_2, \ldots, \eta_n)$.*

We will now focus on the value function given in Definition 6.1. For simplicity the lambda indices as given in the proof above are reversed in the value function. The situation is sketched in Figure 21.

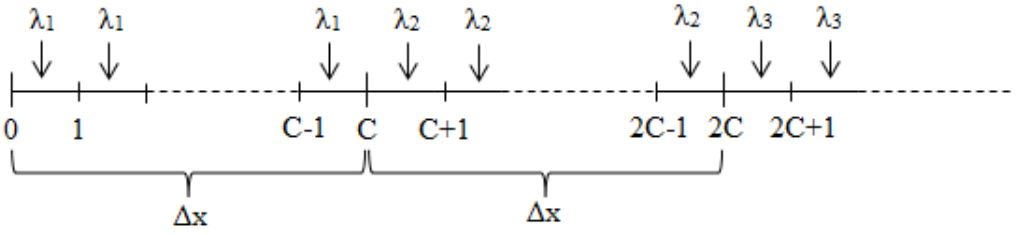


Figure 21: The intervals with corresponding λ values.

In the value function we do not look at the small time steps $\frac{1}{C}$, but only at time intervals of length $\Delta x = 1$. Therefore, every C steps the mean queue length is calculated, so from now on $L_{kC} = L_k$ for $\forall k \in \mathbb{N}$. The lemmas and theorems given in this section hold for $t = kC$, so they can also be used in this case. The value function in Definition 6.1 can be rewritten as:

$$V_n(x,\eta) = \sum_{i=1}^{n-1} \frac{1}{2\mu} (\eta_i + \eta_{i+1}) L_i(x,\lambda) + \frac{1}{2\mu} \eta_1 L_0(x,\lambda) + \frac{1}{2\mu} \eta_n L_n(x,\lambda)$$

As we have shown, $L_k(x,\lambda)$ depends only on λ_r with $1 \leq r \leq k$. Therefore the partial derivatives of the value function are given by:

For $\forall x \in \mathbb{N}$ and $\forall k \in \{1,\ldots,n-1\}$

$$\begin{aligned}
\frac{\partial}{\partial \eta_k} V_n(x,\eta) &= \sum_{i=1}^{n-1} \frac{1}{2\mu}(\eta_i + \eta_{i+1}) \frac{\partial}{\partial \eta_k} L_i(x,\lambda) + \frac{1}{2\mu} L_k(x,\lambda) + \frac{1}{2\mu} L_{k-1}(x,\lambda) + \frac{1}{2\mu}\eta_n \frac{\partial}{\partial \eta_k} L_n(x,\lambda) \\
&= \sum_{i=k}^{n-1} \frac{1}{2\mu}(\eta_i + \eta_{i+1}) \frac{\partial}{\partial \eta_k} L_i(x,\lambda) + \frac{1}{2\mu} L_k(x,\lambda) + \frac{1}{2\mu} L_{k-1}(x,\lambda) + \frac{1}{2\mu}\eta_n \frac{\partial}{\partial \eta_k} L_n(x,\lambda) \\
\frac{\partial^2}{\partial \eta_k^2} V_n(x,\eta) &= \sum_{i=k}^{n-1} \frac{1}{2\mu}(\eta_i + \eta_{i+1}) \frac{\partial^2}{\partial \eta_k^2} L_i(x,\lambda) + \frac{1}{\mu} \frac{\partial}{\partial \eta_k} L_k(x,\lambda) + \frac{1}{2\mu}\eta_n \frac{\partial^2}{\partial \eta_k^2} L_n(x,\lambda)
\end{aligned}$$

For $\forall x \in \mathbb{N}$

$$\begin{aligned}
\frac{\partial}{\partial \eta_n} V_n(x,\eta) &= \frac{1}{2\mu}\eta_n \frac{\partial}{\partial \eta_n} L_n(x,\lambda) + \frac{1}{2\mu} L_n(x,\lambda) + \frac{1}{2\mu} L_{n-1}(x,\lambda) \\
\frac{\partial^2}{\partial \eta_n^2} V_n(x,\eta) &= \frac{1}{2\mu}\eta_n \frac{\partial^2}{\partial \eta_n^2} L_n(x,\lambda) + \frac{1}{\mu} \frac{\partial}{\partial \eta_n} L_n(x,\lambda)
\end{aligned}$$

By Theorem 6.7 $\frac{\partial}{\partial \eta_k} L_i(x,\lambda) \geq 0$ and $\frac{\partial^2}{\partial \eta_k^2} L_i(x,\lambda) \geq 0$ for all $1 \leq i,k \leq n$. Hence the value function is convex and increasing in η.

Theorem 6.8. *For an M/M/1 queue the value function defined in Definition 6.1 is convex and increasing in η.*

6.3.1 M/M/1/N queue

For an M/M/1 queue the mean queue length is convex in λ, as has been shown in the previous paragraph. With some easy calculations given in Appendix B, it can be shown that this does not hold for an M/M/1/N queue.

The mean queue length of an M/M/1/N queue is convex in the neighbourhood of the left border zero and concave at the right border N. In the value function, L_t is multiplied with η, this reduces the concave property of L_t a little bit. Unfortunately the value function is still not convex in the neighbourhood of N as can be seen in Figure 22.

Hence for an M/M/1/N queue the value function is not convex in η. When the road is long enough, such that the road will never be fully occupied, there is no problem. There are some areas where a junction is near to another junction, such that the road connecting these junctions can be totally occupied during rush hours. In this case the queue will not stop growing, but develops further at the roads connected to the last junction. Another argument for dealing with an M/M/1 queue is that the actions we will take do not change whether we are dealing with an M/M/1/N or M/M/1 queue. When the queue is longer than a certain number of vehicles, X (smaller than the the size of the road), the system will ask for longer green times. Therefore we will assume an M/M/1 queue. Of course we do not have infinitely many states, but we will take N large enough

such that this upper state will never be reached. In this thesis N=150 if not stated otherwise, which is large enough in most cases.

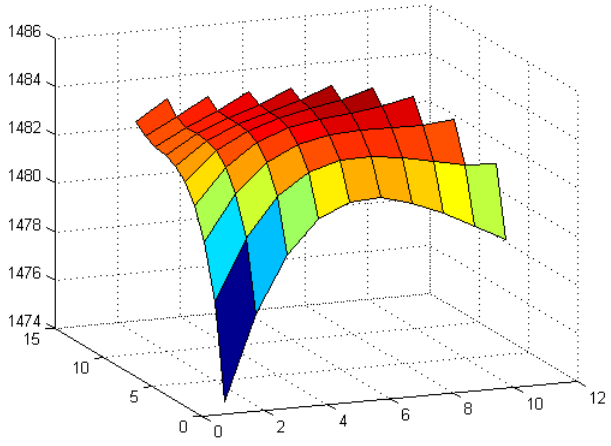


Figure 22: Value function for distributing 10 persons over 3 intervals with $N = 150$ and initial state 149.

7 Local Search

In the previous section is shown that the value function in Definition 6.1 is convex in η, therefore a local search algorithm will find the optimal schedule of the users. In this section the local search algorithm is defined and results are discussed. The results will show that the calculation times of the local search algorithm using the matrix approach can be long. Therefore some rough approximations of the mean queue length will be tested in this section, but first the value function will be slightly modified such that it is better useable in reality.

The value function given in Definition 6.1 schedules relatively more people on the last time interval as can be seen in Table 6. This is optimal since users on the last interval do not effect the waiting times of the previous users. Therefore the value function works well if the time period ends with low inflow rates, if the time period is a whole day from 5 am till 1 am for example. In this case the relative high queue length at the end of the period does not matter, since there is very little traffic on the road.

In practice only time ranges shorter or equal than an hour are used, since the predictions for a time span longer than an hour are very uncertain and the calculation times will increase. In the case of shorter time periods, putting more users on the last interval will effect the waiting times of the users in the next period. This is especially the case for the time periods during rush hours. Hence the initial queue length of the next period has to be taken into account in the value function, therefore the term $\frac{1}{\mu}N_nL_n$ is added. In case of a longer interval the value function would have continued with $\frac{1}{2\mu}\eta_{n+1}L_n$, since the value of η_{n+1} is unknown, the number of users with latest arrival interval n is used instead. The last queue length is important in the next time period since also L_{n+1} etc. depends on it, therefore $\frac{1}{\mu}$ is used instead of $\frac{1}{2\mu}$.

This additional term includes the importance of a low queue length at the last interval to the value function, i.e., the importance of a low initial state of the next period. With this adjustment the value function and corresponding optimization problem becomes:

Definition 7.1. *The optimization problem for multiple app users with multiple deadlines is given by:*

$$
\begin{aligned}
\underset{\eta}{\textit{minimize}} \quad & V(x,\eta) = \sum_{i=1}^{n} \eta_i \frac{L_{i-1}(x,\lambda) + L_i(x,\lambda)}{2\mu} + \frac{1}{\mu} N_n L_n(x,\lambda) \\
\textit{subject to} \quad & \sum_{j=1}^{i} \eta_j \geq \sum_{k=1}^{i} N_k \ \textit{for} \ \forall i \in \{l,\ldots,n\} \\
& \sum_{j=1}^{i-l+1} \eta_j \leq \sum_{k=1}^{i} N_k \ \textit{for} \ \forall i \in \{l,\ldots,n\} \\
& \eta_i \geq 0 \ \textit{for} \ i = 1,2,\ldots,n.
\end{aligned}
$$

This value function is convex, since the addition $L_n(x,\lambda)$ is convex in η (Theorem 6.7). Therefore a local search algorithm is used to find the optimal schedule.

The local search algorithm in pseudo-code:

```
Take η an initial schedule
for i=1:n
    if η_i > 0
        for j=1:n and i ≠ j
            W=zeros(1,n)
            W(i)=-1
            W(j)=1
            η' = η + W
            if η' in solution space, then calculate the value of schedule η'
                if Value η'< Value η then η = η' and start again
                else continue
                end
            end
        end
    end
end
```

The calculation time of the local search algorithm depends on the initial schedule. The initial schedule can make a big difference, since the calculation of the value of a schedule over 12 intervals of 2 minutes takes around 0.3 second (this is due to the matrix multiplications in the matrix approach). The local search over this period with 10 deadlines and $l = 3$ with as initial solution the optimal schedule, will already take around 20 seconds.[9]

Therefore, a good initial schedule should be chosen. In general the inflow should be spread as much as possible, so an initial schedule should take this into account. The function $\sum_{i=1}^{n} \lambda_i^2$ can be used to find a schedule with spread inflow rates, with $\lambda_i = \xi_i + \frac{\eta_i}{\Delta x}$. This function is also convex in λ, so the optimal schedule of this function will be used as initial schedule in the local search. Note that this initial distribution does not take the initial value into account.

Although the local search is much faster with this initial distribution it is still too slow, in Section 7.1 will be shown that the calculation time can be larger than 10 minutes. Therefore, we will consider some rough approximations of the mean queue length which are much faster than the matrix approach described in Section 4.2.1. There are two fast approximations which will be used to calculate the mean queue length:

- If the mean queue length is high enough, such that it will not hit zero in one time step, L_{t+1} can be calculated by $L_{t+1} = L_t + \lambda_{t+1} - \mu$.
- If $\lambda_t < \mu$ the mean queue length in steady-state is given by $\frac{\lambda_t}{\mu - \lambda_t}$. If L_t is small enough and λ does not fluctuate much over time, the steady-state formula can be used for L_{t+1}.

Between these two approximations there is a phase for which no fast approximation is available, the matrix approach can be used in that case. Different combinations of the two approximations

[9] The calculations in this thesis are done in Matlab, so the calculation times can be different in other programming languages.

and the matrix approach have been tested. Three combinations given by methods 2, 3 and 4 below will be discussed in this section and compared to Method 1, the matrix approach. We will see that the performance of the approximation will improve if the matrix approach is used on larger scale, but then also the calculation time will increase.

Method 1: Matrix Approach for $\forall t$

Method 2: $L_t = \begin{cases} \max(L_{t-1} + \lambda_t - \mu, 0) & \text{if } L_{t-1} > \frac{\lambda_{t-1}}{\mu - \lambda_{t-1}} + 1 \bigvee \lambda_{t-1} \geq \mu \bigvee \lambda_t \geq \mu - 1 \bigwedge L_{t-1} \geq 2\frac{1}{2}\mu \\ \text{Matrix Approach} & \text{if } L_{t-1} > \frac{\lambda_{t-1}}{\mu - \lambda_{t-1}} + 1 \bigvee \lambda_{t-1} \geq \mu \bigvee \lambda_t \geq \mu - 1 \bigwedge L_{t-1} < 2\frac{1}{2}\mu \\ \frac{\lambda_t}{\mu - \lambda_t} & \text{otherwise} \end{cases}$

Method 3: $L_t = \begin{cases} \max(L_{t-1} + \lambda_t - \mu, 0) & \text{if } L_{t-1} > \mu \bigvee \lambda_t > \mu \forall t \\ L_{t-1} + 0.4 & \text{if } \lambda_t \in \langle \mu - 1, \mu - \frac{1}{2}] \bigwedge L_{t-1} \leq \mu \\ L_{t-1} + 0.45 & \text{if } \lambda_t \in \langle \mu - \frac{1}{2}, \mu] \bigwedge L_{t-1} \leq \mu \\ \frac{\lambda_t}{\mu - \lambda_t} & \text{otherwise} \end{cases}$

Method 4: $L_t = \max(L_{t-1} + \lambda_t - \mu, 0)$ for $\forall t$

Method 2 uses the fast approximations in the best way, $L_t = \max(L_{t-1} + \lambda_t - \mu, 0)$ is used when the current queue length is large enough, $L_{t-1} > 2.5\mu$. $L_t = \frac{\lambda_t}{\mu - \lambda_t}$ is used when the current queue length is small enough and the previous and current arrival rates are small enough, i.e., $L_{t-1} \leq \frac{\lambda_{t-1}}{\mu - \lambda_{t-1}} + 1$ and $\lambda_{t-1} < \mu$ and $\lambda_t < \mu - 1$. In the other cases the matrix approach is used.

Method 3 does not use the matrix approach, only the fast approximations and two extra formulas are used. The extra formulas are added to improve the connection between the two approximations. Without these two functions $\lambda_t \in \langle \mu - 1, \mu \rangle$ will lead to higher queue lengths than $\lambda_t = \mu$ if $L_{t-1} < \mu$. Since in that case L_t is calculated by $\frac{\lambda_t}{\mu - \lambda_t} > \mu$ for $\lambda_t \in \langle \mu - 1, \mu \rangle$ and by $L_t = L_{t-1} < \mu$ for $\lambda_t = \mu$. With the two extra functions the queue length increases, when λ_t increases.
Since we are especially interested in overloaded systems only $L_t = \max(L_{t-1} + \lambda_t - \mu, 0)$ is used in Method 4.

In Figure 23 the mean queue lengths calculated by Method 1, 2, 3 and 4 are shown for the four scenarios defined in Section 2.3. For scenario high N has to be increased to 200, such that the upper limit is not reached with a large probability in the matrix approach.

As can be seen in Figure 23, Method 2 (red graph) is the best approximation, but Method 3 performs also really well. As expected Method 4 is the worst approximation, in the low scenario it stays zero all the time since $\lambda_t < \mu$ for all t. Based on these results we can conclude that the performance of the approximation increases when the matrix approach is more often used. However, due to more use of the matrix approach the calculation time will also increase, as can be seen in Table 7. The calculation times of Method 3 and 4 are steady and very low, since they do not use the matrix approach.

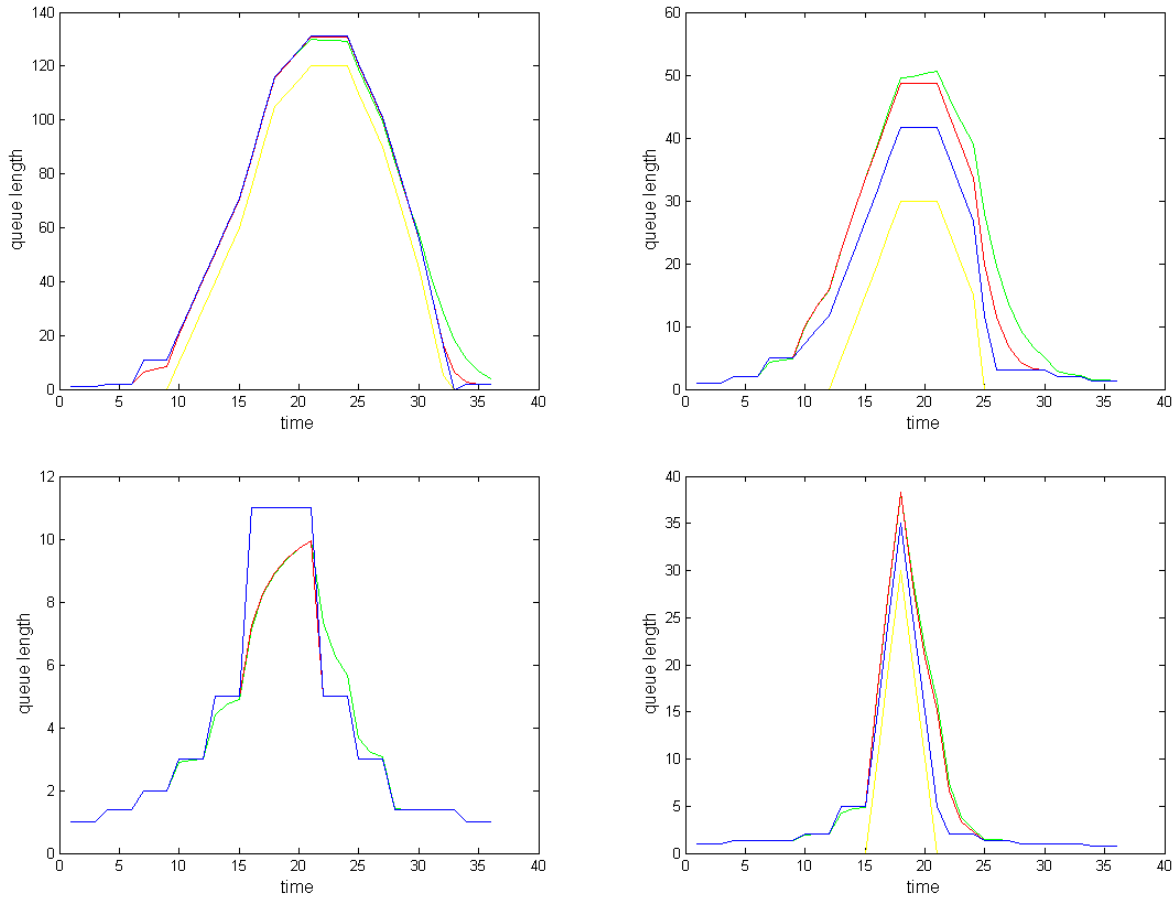


Figure 23: The mean queue lengths of Method 1 (green), Method 2 (red), Method 3 (blue) and Method 4 (yellow) over time for the four scenarios described in Section 2.3. From left to right from top to bottom scenarios: high, medium, low and peak. The mean queue lengths are calculated every 5 minutes.

Scenario	Method 1	Method 2	Method 3	Method 4
high	5.38864	1.28901	0.00004	0.00001
medium	2.83328	0.66927	0.00004	0.00001
low	2.25079	0.47554	0.00004	0.00001
peak	1.86006	0.58491	0.00004	0.00001

Table 7: Calculation times of the four methods for the four scenarios, where the mean queue length is calculated in intervals of 5 minutes.

7.1 Local search results of the four methods

The two rough approximations $L_{t-1} + \lambda_t - \mu$ and $\frac{\lambda_t}{\mu - \lambda_t}$ are both convex in λ_t, but the combination of these approximations is not convex. A simple example that support this:

Take $\mu = 12$ and $x < \mu = 12$ the queue length at time t. Let $\lambda_1 = 10$ and $\lambda_2 = 12$ then according to Method 3 $\frac{1}{2}(L_{t+1}(x, \lambda_1) + L_{t+1}(x, \lambda_2)) = \frac{1}{2}(5 + x + 0.45) \leq L_{t+1}(x, \frac{1}{2}(\lambda_1 + \lambda_2)) = 11$.

Therefore Method 3 is not convex, it can also be shown that Method 2 is not convex. However, Method 4 is convex, since it is not a combination of the approximations, but only uses $L_t = L_{t-1} + \lambda_t - \mu$. For Method 2 and 3 a different search method could be used such that the algorithm does not get trapped in a local minimum. For Method 2 this will not be useful, since the calculation time of for example a genetic algorithm will often be longer than the calculation time of Method 1. For Method 3 the solutions found by the local search algorithm are already really good. Since we are especially interested in overloaded systems ($L_t > \mu$) and in that case Method 3 is convex, we will also use the local search algorithm for Method 3.

The optimal schedules found by the local search algorithm will be shown for the four different methods. Some results are given in the tables below, more optimal schedules can be found in Appendix C. In these results 10 consecutive arrival intervals have been used with $l = 3$ and the interval length to schedule a user is $\Delta x = \Delta t = 2$. The length of the fixed arrival rates is $\Delta T = 10$, so for the fixed total inflow $\Lambda = (\Lambda_1, \Lambda_2, \Lambda_3)$ the situation is as sketched below:

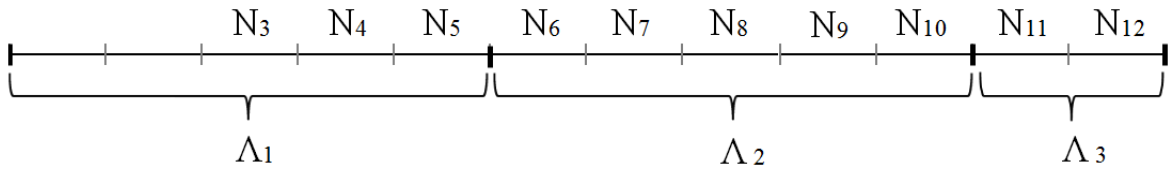


Figure 24: Graphical representation of the inflow rates per interval

Let σ be the fraction of citizens that use the app. The inflow of non app users ξ should be adapted to the value of σ, for example if the fixed total inflow is $\Lambda = (12, 10, 8)$ and $\sigma = 1/4$, then $\xi = (1 - \sigma)\Lambda = (9, 7.5, 6)$[10]. The new total inflow per interval in vehicles per minute used in the Methods is given by $\lambda = \xi + \frac{\eta}{\Delta x}$.

In the tables the number of users per latest arrival interval N_i are given. For the four different methods the optimal schedules η found by the local search algorithm and the calculation times are presented. The value of the schedules given in the tables are calculated using only the matrix approach, Method 1.

$i = 10$, $\Lambda = (12, 10, 8)$ and $\sigma = 1/2$

N_i	0	0	13	12	10	11	11	10	9	9	7	8		
Method	Schedule												Time	Value
1	3	10	8	8	8	9	11	10	9	9	7	8	129.94	39.16
2	7	8	8	8	8	9	9	10	9	9	8	7	1.39	40.63
3	7	8	8	8	8	9	9	10	9	9	8	7	0.24	40.63
4	1	15	7	7	7	10	10	10	9	9	7	8	0.15	40.28

[10]Note that $\xi = (9, 7.5, 6)$ is a short notation for $\xi = (9, \ldots, 9, 7.5, \ldots, 7.5, 6, \ldots, 6)$, the inflow of non app users per interval.

$i = 20$, $\Lambda = (14, 10, 8)$ and $\sigma = 1$

N_i	0	0	28	28	26	21	21	20	20	20	17	16		
Method	Schedule												Time	Value
1	3	23	19	20	19	20	20	20	20	20	17	16	74.99	81.99
2	3	24	23	19	19	19	19	19	19	20	17	16	44.32	85.47
3	4	28	19	19	19	19	19	19	19	19	17	16	0.65	88.30
4	3	35	18	18	18	18	18	18	19	19	17	16	0.62	103.84

$i = 5$, $\Lambda = (10, 10, 10)$ and $\sigma = 1/4$

N_i	0	0	6	5	4	5	4	4	3	5	5	7		
Method	Schedule												Time	Value
1	3	5	4	4	4	4	4	4	4	4	5	3	138.52	17.28
2	4	4	4	4	4	4	4	4	4	4	5	3	0.70	17.32
3	4	4	4	4	4	4	4	4	4	4	5	3	0.26	17.32
4	0	8	4	4	4	4	4	4	4	4	4	4	0.32	17.67

$i = 20$, $\Lambda = (12, 12, 12)$ and $\sigma = 1/2$

N_i	0	0	11	12	13	13	12	10	11	14	12	12		
Method	Schedule												Time	Value
1	0	9	11	10	10	10	11	10	11	14	12	12	633.81	120.42
2	1	11	8	11	6	20	11	6	13	9	12	12	1047.99	129.25
3	0	13	13	4	19	0	12	11	11	13	12	12	0.62	126.24
4	0	3	19	12	11	11	12	11	11	10	10	10	0.41	138.71

Calculation time

The calculation times of Method 3 and 4 are very low, as can be seen in the tables. When the mean queue length is low, the matrix approach is not or little used in Method 2, then the calculation time of Method 2 is also low. If the mean queue length increases the calculation time of Method 2 is significantly longer than the calculation time of Method 3 and 4. In most cases the calculation time of Method 2 is lower than the calculation time of Method 1, but their are exceptions as can be seen in the last table. This is probably due to non-convexity of Method 2.

Performance

As expected we see that the worst schedules are usually found by Method 4, while Method 2 and 3 give the best schedules overall. In most cases the values of Method 3 are very good compared to the optimal values of Method 1. In many parameter settings Method 3 give the same or a better schedule than Method 2. (Due to non-convexity of Method 2 and 3 it is possible that Method 3 gives a better schedule than Method 2.) It can also happen that Method 4 gives the best schedule, as shown in the first table. (This is very rare and is also due to non-convexity of Method 2 and 3.)

Conclusion
To summarize, Method 3 has an optimal ratio between performance and calculation time. The schedules do not differ much from the optimal schedules found by Method 1 and the calculation times are very low. Therefore Method 3 is a good method and will be used in the remaining of this thesis.

If there is more time available to find a schedule, a combination of Method 3 with the other methods can also be used.

- The optimal schedules found by Method 3 and Method 4 are compared and the best schedule is used.
- Another option is to include Method 1 and 2. If the calculation time can be at most T, then the schedules found by method 1 and 2 at time T can also be taken into account. The best schedule found by Method 1, 2, 3 or 4 is used.

7.2 Mean queue length for different fractions of users

When few residents use the app, distributing the users over time has little effect on the queue lengths at traffic lights. The more residents use the app, the more influence the app will have on the queue lengths. In this section different fractions of app users will be distributed over time to see the effect on the mean queue length. Since the users are distributed with as goal to minimize the waiting times of the users, we suspect that the total mean queue length will decrease.

The optimal schedules are calculated for the scenarios described in Section 2.3, Method 3 is used for the calculations. In Figure 25 the mean queue lengths for the optimal schedules are shown for scenarios high for different σ values. As can be seen the mean queue length decreases when more people use the app. Furthermore the number of intervals l in which a user can be scheduled has influence on the mean queue length. If the users can be distributed over a larger time period the mean queue length will decrease, as can be seen in Figure 25.

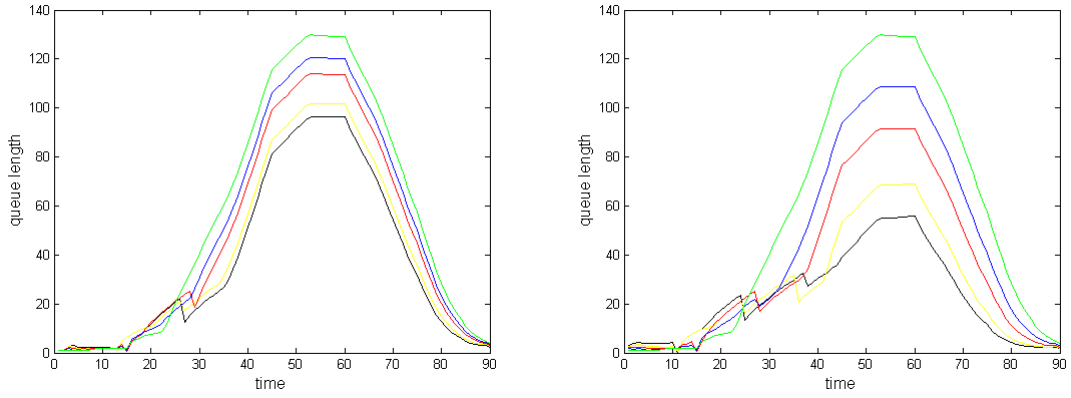


Figure 25: Mean queue lengths for optimal schedules of scenario high with σ values 0 (green graph), 1/4 (blue graph), 1/2 (red graph), 3/4 (yellow graph) and 1 (black graph). In the left figure users can be scheduled in 3 intervals of 2 minutes and in the right figure in 5 intervals.

When the users can be scheduled in many intervals, the inflow will be spread such that the inflow is evenly distributed over time. In reality many citizens want to depart in the same period, to be on time for work for example. Due to the constraints of scheduling all users on time but not too late, a peak in the inflow will still exist during these rush hours. This can also be seen in scenario high, the peak in the mean queue length will decrease but does not disappear entirely.

In Figure 26 the queue length in time is shown for scenario peak for different values of σ and l. Since the peak of the queue length is much lower in this scenario, the peak disappears when 3/4 of the citizens use the app and $l = 5$.

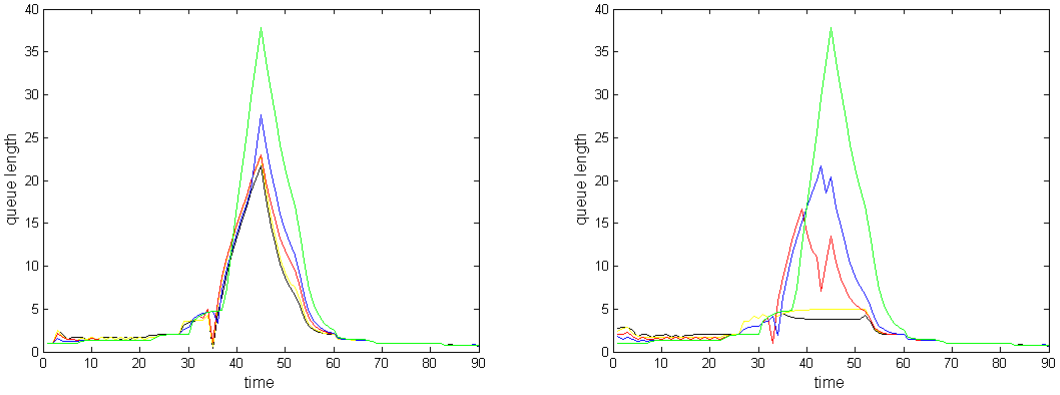


Figure 26: Mean queue lengths for the optimal schedule of scenario peak with σ values 0 (green graph), 1/4 (blue graph), 1/2 (red graph), 3/4 (yellow graph) and 1 (black graph). In the left figure users can be scheduled in 3 intervals of length 2 and in 5 intervals in the right picture.

For $l = 3$ the mean queue length does not improve by increasing $\sigma > 2/4$, this is due to the fact that all users have to be on time out of the queue. The latest arrival intervals of the deadlines during the peak are all within a small period, as shown in Figure 27. Since all users have to be on time, more users have to be scheduled in this small time period when σ increases. By increasing the number of distribution intervals l, we increase the period in which users can be scheduled which solves the problem.

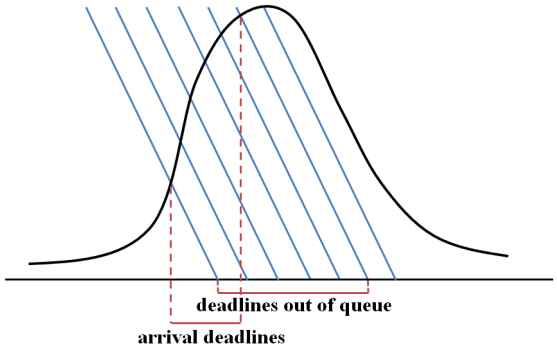


Figure 27: Latest arrival intervals of the deadlines during the peak are all within a small interval.

8 Departure advice for multiple app users

To make a schedule as done in the previous section, we need the latest arrival interval per user. Furthermore a decision have to be made about which route a user has to take, via traffic light $1, 2, \ldots, S$. This section is about all the calculations that have to be done before the users can be distributed over the time slots at a traffic light. In the last paragraph all calculations will be put together into one algorithm.

8.1 Latest arrival intervals

To be able to schedule the users as have been done in the previous section we have to know the latest arrival interval A_D per deadline D. Users which are scheduled before or at interval A_D will be with a pre-set confidence level on time out of the queue, before D. We define $A = (A_1, A_2, \ldots, A_n)$ with A_i the latest arrival interval of deadline i and $N = (N_1, \ldots, N_n)$ with N_i the number of users with latest arrival interval i. N_i can be calculated by summing the users of the deadlines which have i as latest arrival interval.

It is difficult to fix the latest arrival intervals, since they will change when the schedule changes. Therefore we have to recalculate the latest arrival intervals when a new schedule is found. The algorithm to find the optimal schedule, which includes the method to find the latest arrival intervals, consist of the following steps:

1. Calculate the latest arrival interval with the intersection method for all deadlines A without users scheduled, $N = 0$.
2. Add every user to the right latest arrival interval, this gives a new N.
3. Calculate the optimal schedule with the local search algorithm, use N as in step 2.
4. Calculate the latest arrival intervals A with the intersection method, by taking into account the schedule found in step 3.
5. Repeat step 2, 3 and 4 until the latest arrival intervals, A, are the same as in the previous iteration.

To add every user to the right latest arrival interval, we will need the deadline per user. These are stored in the vector *Deadline* with on element i the deadline of user i. Next to the non app users ξ and the app users we have to schedule, there are also fixed scheduled app users γ. These users already got a departure sign, but still have to arrive at the queue. $\gamma = (\gamma_1, \ldots, \gamma_n)$ is the schedule of fixed scheduled app users, with γ_i the number of fixed scheduled users that will arrive during interval i.

With the intersection method as shown in Figure 17, the latest arrival intervals per deadline A are calculated. The confidence bound used in this methods is calculated by the matrix approach, therefore we need the total inflow λ in vehicles per minute per interval. This inflow λ is calculated by the sum of the inflow of the non app users ξ, the fixed scheduled users γ, and the scheduled users η. Since γ and η are the number of scheduled users per interval, they should be divided by the interval length Δx to get the number of vehicles per minute.

The pseudo code of the algorithm to find the optimal schedule is given by:

```
N'=0
repeat
    N=N'
    Calculate the optimal schedule with N
    Output: Schedule η

    for i=1:n
        λ(i)=ξ(i)+γ(i)/Δx+η(i)/Δx
    end
    Calculate A by the intersection method with λ as defined above

    for i=Users
        N'(A(Deadline(i,1)))=N'(A(Deadline(i,1)))+1
    end

until N'=N
```

The pseudo code to calculate the optimal schedule is left out, since this is discussed in detail in the previous sections. (The pseudo code of the local search algorithm is given in Section 7.) If $N = N'$, then the latest arrival intervals do not change anymore and the optimal schedule is found.

Note that the algorithm can be trapped in an infinite loop of repeating A values. In that case the vector A, of the repeating vectors, with in total the lowest arrival intervals is chosen. After ten iterations the algorithm is stopped, since the A values will not change that much anymore and to fasten the algorithm. From the last five vectors, the vector A with in total the lowest arrival intervals is chosen.

8.2 Distribution over the routes

In most neighbourhoods there are multiple routes out of the residential area, the user could take the route via traffic light $1, 2, \ldots, S$. For example in IJburg there are two routes, so the last step is to divide the users over the traffic lights. The users are divided in almost the same way as in the single user case, based on the latest arrival intervals and the travel times. Due to the increase in calculation time we do not want to include the distribution over the traffic lights into the local search algorithm. Therefore, the distribution of the users over the traffic lights has to be done before the users are scheduled in a time slot at a traffic light.

8.2.1 Algorithm

The algorithm to distribute all users which have a deadline in interval Y over the traffic lights will be described in this subsection. The length of prediction interval Y will be set in the next paragraph. First some parameters and input variables are defined.

To distribute the users over the traffic lights the deadlines of every user at every traffic light are needed. These deadlines are stored in the matrix *Deadline* with the users in the rows and in the first column the deadlines of queue 1, in the second column the deadlines of queue 2 etc. Hence $Deadline(i, 1)$ gives the deadline of user i at queue 1.
We also have to know the travel times from the origin to the traffic lights. Since small roads do not contain loops, there is no data available to measure the travel times in most residential areas.[11] Therefore the travel time is calculated by the free-flow travel time, for more information see Appendix D2. These fixed travel times from the origin to the queues are stored in the matrix t in the same way as the deadlines, so $t(i, 1)$ is the travel time from the origin to queue 1 for user i.

There is an upper bound on the waiting time at a traffic light. Most of the time this upper bound can be derived from historical data, if this data is not available the state limit N can be used. N is chosen such that it will never be reached, therefore the upper waiting time can be calculated by N/μ. The maximum waiting time can be calculated for all traffic light $1, 2, \ldots, S$ and the maximum waiting time of the system is $W^{max} = \max(W_1^{max}, \ldots, W_S^{max})$.

Some users are scheduled immediately at a traffic light, due to one of the following causes:

1. For some users there is a route which has obviously the shortest travel time. If the detours of the other routes are large enough this route is automatically chosen. The difference in travel time between the optimal route and the other routes should be more than W^{max} minutes, then the user is immediately scheduled at the route with the shortest travel time.

2. Users which are already scheduled in a previous calculation and the departure messages for the not chosen routes should already be sent. In the current calculation there is no choice anymore the user is automatically scheduled on the same route and traffic light as in the previous calculation.

The users which are not immediately scheduled will be distributed based on the latest arrival intervals per queue and the travel times t. The latest arrival intervals per deadline of queue 1 are stored in the vector $A1$. The departure time for user i for the route via traffic light 1 is calculated by $A1(Deadline(i, 1)) - t(i, 1)$. The same calculations can be done for the routes via traffic light $2, 3, \ldots, S$. The route with the latest departure time will be advised by the app and the user will be assigned to the traffic light corresponding to this route.

The latest arrival intervals have to be calculated in the same way as described in the previous section, only there are now multiple queues instead of one. If the number of users that are scheduled at in an interval changes the latest arrival intervals can also change. Therefore we have to update the latest arrival intervals if the distribution of the users over the traffic lights changes. When the latest arrival intervals do not change anymore, the departure times from the origin also stay the same, so then the optimal distribution over the traffic lights has been found.

For two traffic lights ($S = 2$) the pseudo code of the algorithm is given below. Users which have a deadline in interval Y are stored in the vector $Users$ and the immediately scheduled users are stored in $UsersScheduled$. The fixed scheduled users per interval at queue 1 and queue 2 are given by $\gamma 1$ and $\gamma 2$. The inflow of non app users at queue 1 and queue 2 are $\xi 1$ and $\xi 2$, respectively.

[11] This is also the case for the neighbourhood IJburg.

```
Define immediately scheduled users due to cause 1
for i= Users
    if |Deadline(i,1)-t(i,1)-Deadline(i,2)+t(i,2)|>W^max
        if Deadline(i,1)-t(i,1)>Deadline(i,2)-t(i,2)
            i ∈ UsersScheduled1
        else i ∈ UsersScheduled2
        end
    end
end

Users=Users \ (UsersSchedule1 ∪ UsersScheduled2)

N1'=0
N2'=0
repeat
    N1=N1'
    N2=N2'

    Calculate latest arrival intervals
    Calculate the optimal schedule at queue 1 with N1
    Output: Schedule η1

    for i=1:n
        λ1(i)=ξ1(i)+γ1(i)/Δx+η1(i)/Δx
    end
    Calculate A1 with the intersection method with λ1 as defined above

    Calculate the optimal schedule at queue 2 with N2
    Output: Schedule η2

    for i=1:n
        λ2(i)=ξ2(i)+γ2(i)/Δx+η2(i)/Δx
    end
    Calculate A2 with the intersection method with λ2 as defined above

    Divide the users over the traffic lights.
    for i=Users
        if A1(Deadline(i,1))-t(i,1)>A2(Deadline(i,2))-t(i,2)
            N1'(A1(Deadline(i,1)))=N1'(A1(Deadline(i,1)))+1
        else N2'(A2(Deadline(i,2)))=N2'(A2(Deadline(i,2)))+1
        end
    end

    for i=UsersScheduled1
        if A1(Deadline(i,1)) in interval Y
```

```
            N1'(A1(Deadline(i,1)))=N1'(A1(Deadline(i,1)))+1
        end
    end

    for i=UsersScheduled2
        if A2(Deadline(i,2)) in interval Y
            N2'(A2(Deadline(i,2)))=N2'(A2(Deadline(i,2)))+1
        end
    end
until N1'=N1 and N2'=N2

Define the users per queue and the immediately scheduled users due to cause 2
for j=Users
    if A1(Deadline(j,1))-t(j,1)>A2(Deadline(j,2))-t(j,2)
        j ∈ UsersQ1
        if A2(Deadline(j,2))-t(j,2)<InfoTime+RunTime
            j ∈ UsersScheduled1
        end
    else j ∈ UsersQ2;
        if A1(Deadline(j,1))-t(j,1)<InfoTime+RunTime
            j ∈ UsersScheduled2
        end
    end
end

UsersQ1= UsersQ1 ∪ UsersScheduled1
UsersQ2= UsersQ2 ∪ UsersScheduled2
```

Here $InfoTime$ is the number of intervals that is needed to inform a user 5 minutes before departure and $RunTime$ is the time between two calculation times.

As can be seen in the code the immediately scheduled users are separated from the normal users before starting the algorithm of distributing the users over the traffic lights. After the algorithm the users which are immediately scheduled (in the next calculation) due to cause 2 are added to $UsersScheduled$. As output of this algorithm the users per queue are known, $UsersQ1$ and $UsersQ2$, and the number of users per latest arrival interval, $N1$ and $N2$. Thus now the optimal schedule per queue can be made.

Just as in the previous paragraph the algorithm can be trapped in an infinite loop of repeating N1 and N2 values. In that case we take the combination of latest arrival intervals of queue 1 and 2 (A1 and A2) with in total the lowest values. To fasten the algorithm the loop is also stopped after ten iterations. In that case the combination with in total the lowest arrival intervals is chosen from the last five pairs of A1 and A2.

8.2.2 Length prediction interval Y

The users that have a deadline in a certain prediction interval Y will be distributed over the traffic lights as shown in the previous section. But how large should this interval Y be?

When a user i has a deadline in the interval and is not immediately scheduled, then $\exists j, k \in \{1, 2, \ldots, S\}$ such that

$$\begin{aligned} &|Deadline(i,j) - t(i,j) - Deadline(i,k) + t(i,k)| < W^{max} \\ &\Rightarrow |Deadline(i,j) - Deadline(i,k)| < W^{max} + t_i^{max} \quad \text{with } t_i^{max} = \max t(i,:). \end{aligned}$$

In this case the latest arrival intervals of both queues j and k have to be known to schedule the user at one queue. Therefore the deadlines of both queues should be together in interval Y at a calculation time. Hence the interval should be larger than $W^{max} + t^{max} + \Delta k$ minutes, where Δk is the time between two calculation times.

The user has to be informed about the approaching departure, therefore the app will give a warning signal 5 minutes before departure. By adding the maximum waiting time and travel time, the information time is at most $5 + W^{max} + t^{max}$ minutes. Therefore interval Y should at least be $5 + W^{max} + t^{max} + W^{max} + t^{max} + \Delta k = 2W^{max} + 2t^{max} + \Delta k + 5$ minutes. It can happen that this is longer than 60 minutes, that is not desirable since predictions for a time span longer than an hour are very uncertain. Therefore the length of Interval Y will be at most 60 minutes.

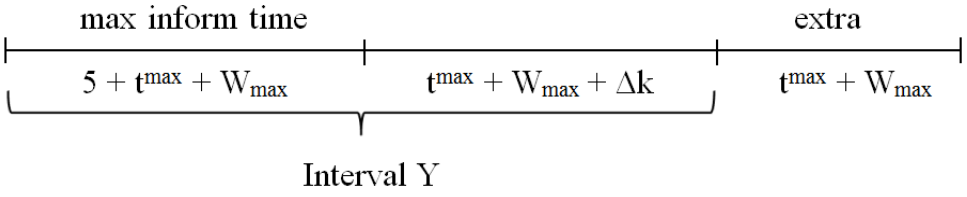


Figure 28: Length of interval Y.

For the users which have a deadline in interval Y but are not immediately scheduled, it can happen that the deadline of one traffic light lies outside interval Y. This deadline will be at most $W^{max} + t^{max}$ minutes later than the deadlines in interval Y. By also calculating the latest arrival intervals of the deadlines in the extra interval in Figure 28, all app users with a deadline in interval Y can be divided. The waiting times outside interval Y can still increase, therefore the division of these users is temporarily.

Hence if the difference between the deadlines is smaller than $W^{max} + t^{max}$ minutes, the deadlines should be together in interval Y at some calculation time, to schedule it definitely. The extra interval is only used to distribute the users in interval Y over the traffic lights, i.e., for this interval only the latest arrival intervals are calculated.

The length of prediction interval Y given in this section is an advice. If the interval is larger fluctuations in inflow rates can be observed earlier and the algorithm can react better on this

change. For high fluctuations in inflow rates it is therefore useful to take a larger prediction interval to schedule the users. Most of the time interval Y will be chosen between 30 and 60 minutes and the extra interval between 15 and 30 minutes.

8.3 Total Algorithm

All separate parts are now developed, the only step left is to put everything together into one algorithm. Before the total algorithm will be given first the input and output variables are discussed.

8.3.1 Input

The app distracts the preferred arrival time at the destination from the agenda of the user. The travel times from the traffic lights to the destination are calculated by a travel time forecasting model and are assumed to be given as input. The deadline at a traffic light is based on the travel time from the traffic light to the destination, such that the user is on time at the destination. These travel times are updated every τ minutes, so every τ minutes the deadlines are also updated by:

Deadline=Prefered Arrival Time - Travel Time (traffic light to destination)

Also the travel times from the origin to the traffic lights have to be known. Most of the time there is no data available of the roads in a residential area, therefore these travel times are fixed calculated by the free-flow travel time (see Appendix D2 for more information).

First we will assume that all travel times are fixed, calculated by the free-flow travel time. With this assumption the deadlines at traffic lights $1, 2, \ldots, S$ are also fixed. In a later stadium of the app the travel times will change every τ minutes, so then the deadlines will be updated at every calculation time.

Because the current time is $t = 0$, the parameters should adapt to this current time. For example, in the case of fixed travel times, at every calculation the deadlines should be adapted to the current time by distracting the number of intervals between two calculations, NIntervalRun. Also the right arrival rates ξ should be selected.

8.3.2 Output

Five minutes before departure the app should give a warning signal and the route the user has to follow. To accomplish this the algorithm has to be determine which users should get a sign in the next period of length Δk. Hence these are the users which departure times are less than $5 + \Delta k$ minutes away. The users which have to be informed, the departure times and corresponding routes are all stored in the matrix $TimeUsers$. This matrix is the output of the algorithm after each calculation, with in the first column the users that should get a sign, in the second column the departure times and in the last column the route via traffic light $1, 2, \ldots, S - 1$ or S.

The informed users are fixed scheduled at an interval at a traffic light, let $\gamma 1_i$ be the number of users which are fixed scheduled at interval i at queue 1 and $\gamma 1 = (\gamma 1_1, \ldots, \gamma 1_n)$. These users will arrive at traffic light 1, but we can not change the arrival intervals any more. For $\gamma 2, \ldots \gamma S$ the same definition hold for queue $1, \ldots, S$, respectively.

Let $UsersGone$ be the users which already got a departure sign. These users should not be taken into account in the current calculation.

8.3.3 Algorithm

We distinguish the following parts in our model:

Part

1. First the parameters and deadlines have to be set and the users that have a deadline in prediction interval Y have to be selected.
2. The fixed scheduled users have to be removed and the immediately scheduled users have to be separated from the normal users. Then the users can be distributed over the traffic lights.
3. From Part 2 the users are divided over the traffic lights and per traffic light the latest arrival intervals per deadline are known. Therefore a schedule per traffic light can be made.
4. The users that get a departure sign in the next period are selected, also the corresponding departure times and routes are given.
5. The already scheduled users are updated for the next calculation time.

Below the algorithm is given in pseudo code for the case of two traffic lights. The code of part 2 and 3 are left out, since the code of Part 2 can be found in Section 8.2.1 and the calculation of a schedule by the local search algorithm is discussed in detail in the previous sections and a part of the pseudo code can be found in Section 8.1.

NInterval is the number of intervals in interval Y and NIntervalTotal is the number of intervals in interval Y + extra.

```
Part 1
ξ1=ξTotal1(CurrentInterval:CurrentInterval+NIntervalTotal-1)
ξ2=ξTotal2(CurrentInterval:CurrentInterval+NIntervalTotal-1)

Deadline(:,1)=Deadline(:,1)-NIntervalRun
Deadline(:,2)=Deadline(:,2)-NIntervalRun

Users1=find(Deadline(:,1) ≤ NInterval)
Users2=find(Deadline(:,2) ≤ NInterval)
Users=(Users1 ∪ Users2) \ UsersGone

Part 2: See Section 8.2.1
Users are distributed over the traffic lights
Output: N1, N2, UsersQ1, UsersQ2

Part 3: See Section 8.1
```

```
Find the optimal schedule at queue 1 for UsersQ1 with N1
Output: Schedule η1

Find the optimal schedule at queue 2 for UsersQ2 with N2
Output: Schedule η2

Part 4
% queue 1
SortUsers1=Sort the UsersQ1 in increasing order of their deadline at queue 1

r=0
b=0
for j=find(η1>0)
    for i=1:η1(j)
        b=b+1;
        if j-t(SortUsers1(b),1)-1<=InfoTime+RunTime
            TimeUsers(r,1)=SortUsers1(b)
            TimeUsers(r,2)=CurrentInterval+j-t(SortUsers1(b),1)
            TimeUsers(r,3)=A
            γ1(j)=γ1(j)+1
        end
    end
    if j>t^max+NIntervalInfoTime+NIntervalRun
        break
    end
end

% queue 2
SortUsers2=Sort the UsersQ2 in increasing order of their deadline at queue 2.

b=0;
for j=find(η2>0)
    for i=1:η2(j)
        b=b+1;
        if j-t(SortUsers2(b),2)-1<=InfoTime+RunTime
            TimeUsers(r,1)=SortUsers2(b)
            TimeUsers(r,2)=CurrentInterval-1+j-t(SortUsers2(b),2)
            TimeUsers(r,3)=B
            γ2(j)=γ2(j)+1
        end
    end
    if j>t^max+NIntervalInfoTime+NIntervalRun
        break
    end
end
```

Part 5

```
UsersGone=UsersGone  ∪  TimeUsers(:,1)

γ1'(1:NInterval-NIntervalRun)=γ1(NIntervalRun+1:NInterval)
γ2'(1:NInterval-NIntervalRun)=γ2(NIntervalRun+1:NInterval)
γ1=γ1'
γ2=γ2'

UsersScheduled1= UsersScheduled1 \ UsersGone
UsersScheduled2= UsersScheduled2 \ UsersGone

CurrentInterval=CurrentInterval+NIntervalRun
```

9 Experimental results

Since the pilot has not been started yet when this thesis was finished, there are no results from the pilot to discuss. Therefore the algorithm is tested in a simulation, where of the results are discussed in this section.

In the beginning of the thesis we assumed exponential departure and arrival rates, such that the knowledge of M/M/1 queues could be used. In reality the departure times are not exponential distributed, but more deterministic. Therefore our model is tested in a simulation with deterministic departure times.

First some results of the mean queue length are discussed for both exponential and deterministic departure times.

9.1 Mean queue length

We have developed many different methods to calculate the mean queue length. In this section we will compare the queue lengths calculated by these methods to the mean queue lengths found by simulation.

In the left picture of Figure 29 the queue length is shown for 1000 simulations of scenario medium defined in Section 2.3. In this simulation we assumed exponential departure times. The variation of the queue length per simulation is very large, this is also visible in the standard deviation of the queue length in the right picture of Figure 29.

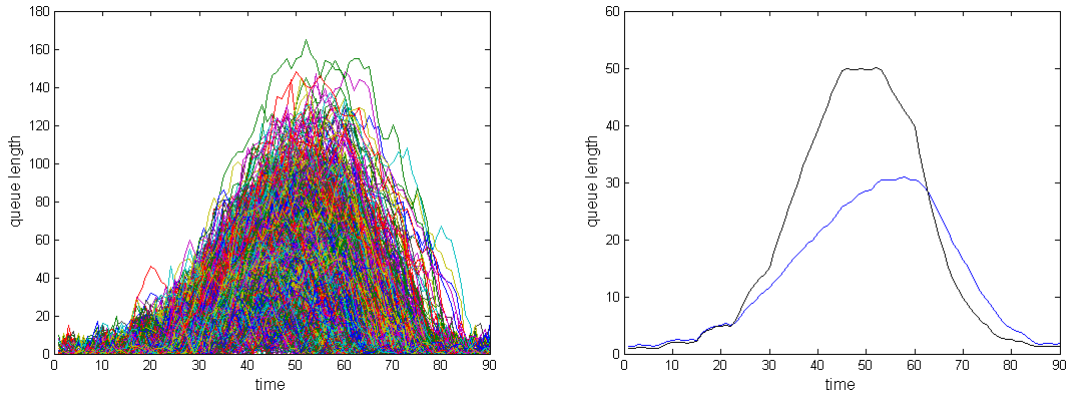


Figure 29: For scenario medium the queue length of 1000 simulations are shown in the left picture and in the right picture the mean queue length (black graph) and standard deviation (blue graph) in time are given. The departure times are exponential distributed in this simulation.

In Figure 30 the same output is shown for 1000 simulations with deterministic departure times. Figure 30 looks similar to Figure 29, only both the mean queue length and standard deviation are a bit lower than in the exponential departure case. These differences are caused by the fluctuations in the number of departures per minute in the exponential case. In the deterministic case every minute μ vehicles will depart, but in the exponential case the departure times fluctuate a lot with

as mean μ departures per minute. If one departure time is very long, this has effect on the queue length.

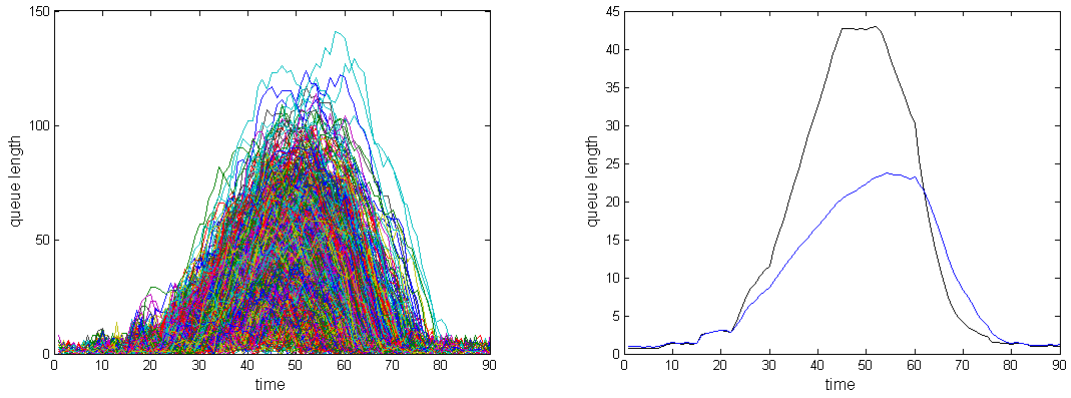


Figure 30: For scenario medium the queue length of 1000 simulations with deterministic departure times are shown in the left picture. In the right picture the mean queue length (black graph) and standard deviation (blue graph) in time are shown.

The developed methods to calculate the time-dependent queue length distribution and the mean queue length are based on M/M/1 queues. Since there is a difference in the mean queue lengths of systems with exponential and deterministic departure times, we have to know how this effects the performance of the developed methods.

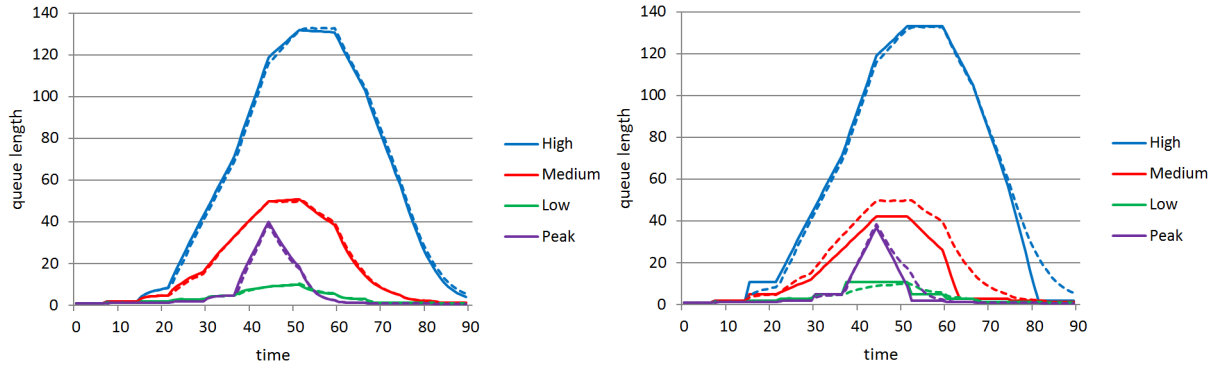


Figure 31: The striped graphs represent the mean queue lengths in time found by simulations with exponential departure times. In the left figure the mean queue lengths in time calculated by the the matrix approach (solid graphs) are shown and in the right figure the mean queue lengths calculated by Method 3 (solid graphs).

In the left graph of Figure 31 can be seen that the mean queue lengths found by the matrix approach are almost equal to the simulated mean queue lengths with exponential departure times. Therefore, the matrix approach is very accurate in forecasting the mean queue length when the departure times are exponentially distributed. Method 3 approximates the mean queue length really well as shown in the right figure, only in the medium scenario the mean queue length is underestimated.

In Figure 32 the simulation results of the deterministic departure times are compared with the mean queue lengths of the matrix approach and Method 3. For all scenarios the mean queue lengths found by the matrix approach are higher than the simulated mean queue lengths, this confirms the results found in Figure 29 and 30. The mean queue lengths of Method 3 are close to the simulated results. Only for the high scenario, there is a small difference between Method 3 and the simulated mean queue length.

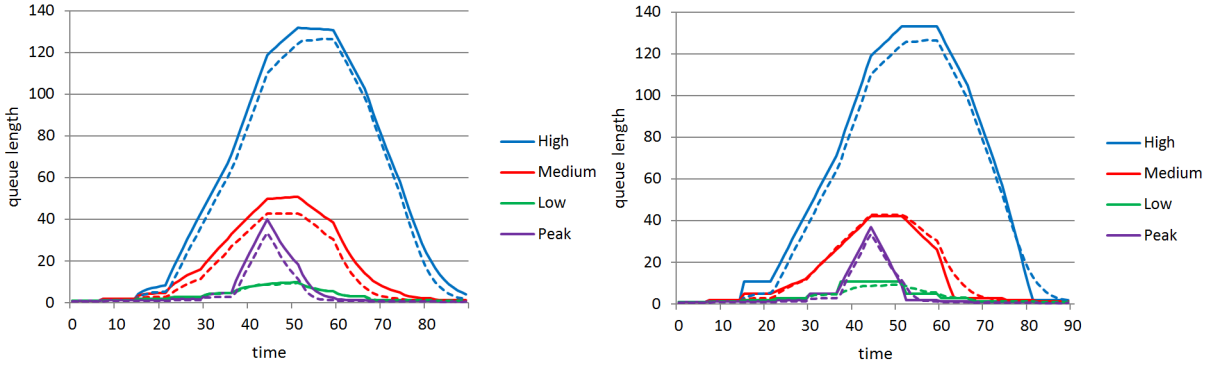


Figure 32: The striped graphs represent the mean queue lengths in time calculated by simulations with deterministic departure times. In the left figure the mean queue length in time calculated by the the matrix approach (solid graphs) is shown and in the right figure the mean queue length calculated by Method 3 (solid graphs).

Hence, Method 3 provides good approximations of the mean queue length of a queue with deterministic departure times. The matrix approach overestimates the mean queue length a little bit. Since the matrix approach is used to calculate the confidence bound, this overestimation ensures that more than $1-\alpha$ fraction of the users will be on time out of the queue. Therefore, both Method 3 and the matrix approach perform well in the real case of deterministic departure times.

9.2 One queue

We will first test the algorithm, given in Section 8.3, on one queue to measure the effect of distributing the users over time. In the next paragraph, the dynamics of distributing the users over the two queues are included.

The algorithm is used on the scenarios described in Section 2.3. We used intervals of length $\Delta x = 2$ and every $\Delta k = 6$ minutes the algorithm is calculated. Every set of parameters is simulated 50 times. The mean queue lengths of the simulations of the different scenarios for different fractions of users are given in Figure 33.

When the fraction of citizens using the app increases, the mean queue length decreases in the medium, low and peak scenario. In the low scenario the queue length is already low without distributing the road users. Hence, distributing a small fraction of users over $l = 3$ intervals can already spread the inflow. Therefore the queue length decreases the most from $\sigma = 0$ to $\sigma = 1/4$.

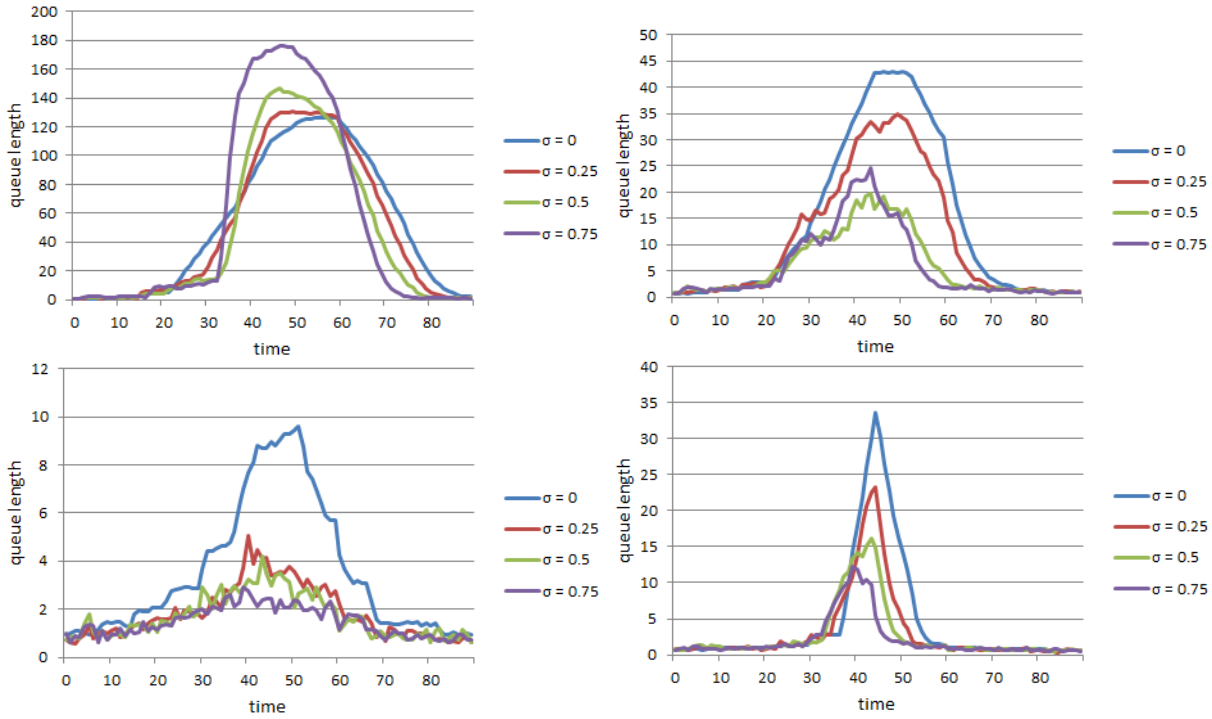


Figure 33: Mean queue lengths for the different scenarios and different σ values. From left to right from top to bottom scenario high ($l = 5$), medium ($l = 5$), low ($l = 3$) and peak ($l = 3$).

For the medium scenario the mean queue length decreases a lot for $\sigma = 1/4$ and $\sigma = 2/4$, but for $\sigma = 3/4$ the mean queue length stays almost the same as can be seen in Figure 33. This is due to the fact that all users should arrive on time, but not too early at the destination. Therefore, the users have to be scheduled in the l intervals before or at their latest arrival interval. The latest arrival intervals of the deadlines during the peak of inflow are all within a small period, as shown in Figure 34. Since all users have to be on time, more users have to be scheduled in this small time period when the σ value increases. (Because for $\sigma = 3/4$ much more users have to be scheduled than for $\sigma = 1/4$.) Therefore the mean queue length will not always improve, when σ increases and l stays the same. By increasing the distributing intervals l we increase the period in which users can be scheduled which solves the problem, as can be seen in Table 9 and Figure 35.

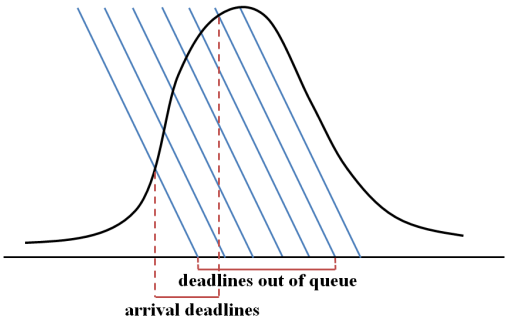


Figure 34: The latest arrival intervals of the deadlines during the peak are within a small interval.

In scenarios high and medium we took $l = 5$ and for scenarios low and peak $l = 3$, since scenarios with a higher queue length need more distributing space to lower the mean queue length. For scenarios peak and medium the optimal schedules are also calculated by distributing the users over more intervals, the results are shown in Figure 35. For all the fractions the mean queue lengths decreased compared to the lower l values in Figure 33. Since there is now enough space to schedule the users, all users will be on time and still the mean queue length will decrease when the fraction of users increases. This shows that when both the mean queue length and the fraction of users is high, more space should be given to schedule the users to ensure that the mean queue length will decrease. Of course the number of intervals can not be too large, otherwise the users will be too early at their destination. Since we have to distribute the users bounded by the constraints of l distributing intervals, not all congestions can be removed completely.

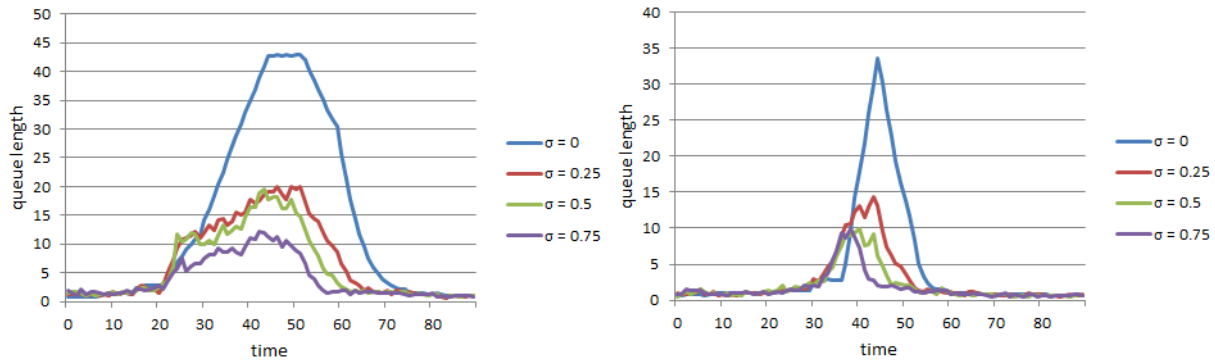


Figure 35: Mean queue lengths for the optimal schedules for the scenario medium ($l = 7$) and peak ($l = 5$) for different fractions of users.

For the high scenario a different pattern is seen in Figure 33. By raising the fractions of users the peak becomes steeper and shifts to the left. This is due to the fact that all user have to be on time, therefore many users have to arrive in a smaller area earlier in time. Increasing the number of distributing intervals l has to eliminate this problem.
Another option is to increase the prediction interval, such that the algorithm can react earlier on the increase in inflow. In Figure 33 the prediction interval of scenarios low and peak is 30 minutes, for medium 40 minutes and for scenario high 50 minutes. If the prediction interval of scenario high increases to 60 minutes the mean queue length will decrease as shown in Figure 36. In this case the waiting time per app user will also decrease from 4.49 ($\sigma = 1/4$) to 4.41 ($\sigma = 2/4$) until 3.83 ($\sigma = 3/4$).

In Table 9 the results of the simulations are summarized. Since the waiting times are related to the mean queue lengths the same results are seen as discussed above. The higher the fractions and the more distributing intervals l, the lower the waiting times. We have seen that for some fractions the mean queue lengths and waiting times do not improve, due to the fact that everyone has to be on time out of the queue. We used a confidence level of $1 - \alpha = 0.90$ to ensure that 90% of the users is on time. The results in Table 9 show that this goal is more than achieved. This is because users are not only scheduled at their latest arrival interval, but also on the $l - 1$ intervals before.

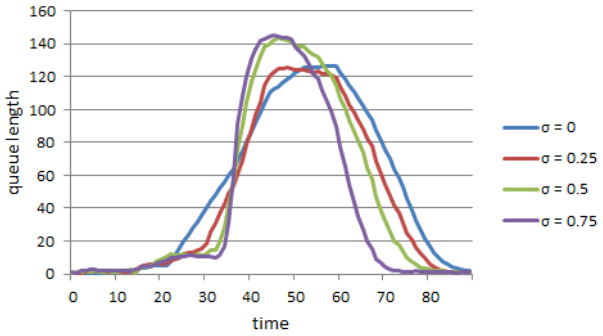


Figure 36: Scenario high with prediction interval of 60 minutes.

Scenario	σ	l	Waiting time per user	number users on time	number users too late
Peak	0.25	3	0.3656	391.1	8.7
	0.50	3	0.3158	691.7	14.4
	0.75	3	0.2687	1055.7	26.2
Peak	0.25	5	0.2930	392.1	8.1
	0.50	5	0.2423	694.1	13.1
	0.75	5	0.2241	1057.5	24.9
Low	0.25	3	0.2122	413.5	12.5
	0.50	3	0.1900	753.1	24.9
	0.75	3	0.1786	1125.7	38.3
Medium	0.25	5	1.0988	468.0	16.1
	0.50	5	0.6129	869.9	23.1
	0.75	5	0.6205	1289.3	27.7
Medium	0.25	7	0.6783	472.2	12.0
	0.50	7	0.6072	870.8	22.2
	0.75	7	0.4114	1289.3	27.7
High	0.25	5	4.8162	499.2	16.8
	0.50	5	4.4267	931.7	23.3
	0.75	5	5.2117	1392.6	44.4

Table 9: Results of the simulations.

In the simulations above the interval length $\Delta x = 2$ is used, but also other values can be chosen. When the interval length decreases, the mean queue length will also decrease, since the users can be distributed more precisely. The users have to be scheduled in intervals of one minute or longer, since more detailed intervals will not work in practice. Increasing the interval length will fasten the algorithm, but the results will be less optimal since the users can be distributed over less intervals. The results for the scenario peak with $\Delta x = 1$ are given in Table 10. For the best balance between calculation time and performance, we advise to schedule the users in intervals of two or three minutes.

Scenario	σ	l	Waiting time per user	number users on time	number users too late
Peak	0.25	6	0.3780	380.9	17.7
	0.5	6	0.2832	679.3	20.9
	0.75	6	0.2738	1043.3	39.9

Table 10: Results peak scenario with interval length $\Delta x = 1$.

For $\sigma = 1/4$ the mean queue length decreases a lot when the users are distributed over the intervals. The question is from which fraction on we see significant results when scheduling the users over time. In Figure 37 the mean queue lengths are given for low fractions of users for scenario medium and high. When 4% of the citizens use the app the queue length will decreases a little bit, so when more than 4% of the citizens use the app, distributing the users over time will be useful. When a lower fraction of citizens uses the app, the app has too little influence to decrease the queue length. In that case the departure advice for a single app user as described in Section 5.1 can be used.

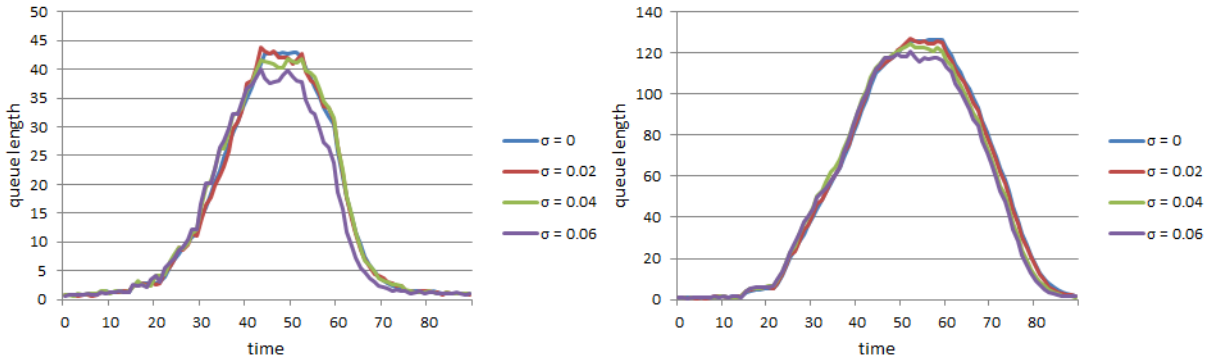


Figure 37: Mean queue lengths for scenario medium ($l = 5$) and high ($l = 7$) for low fractions of users.

9.3 Two queues

In the previous section the algorithm to distribute the users over time at one traffic light was tested. In this paragraph, we include the distribution over the traffic lights into the simulation. We discuss the special case of two traffic lights, $S = 2$, the results for multiple traffic lights will be similar.

Since the scenarios described in Section 2.3 are for one traffic light, some new scenarios are defined in Table 11. The inflow λ, outflow μ and initial queue lengths are given for the two traffic light. Every quarter the inflow rate will change, thus the scenarios are given for an hour.

In the first scenario traffic light 2 is more congested than traffic light 1, but the green times are the same. In the second scenario the inflow rates are the same, but the first traffic light has longer green times. In the last scenario traffic light 1 has increasing inflow rates, while at traffic light 2 the inflow rates are decreasing.

Scenario	λ_1	λ_2	μ_1	μ_2	Initial 1	Initial 2
1	(8,8,8,8)	(12,12,12,12)	12	12	0	0
2	(8,10,10,10)	(8,10,10,10)	12	8	0	0
3	(8,9,11,12)	(12,12,11,9)	12	12	0	40

Table 11: Scenarios of an hour for two traffic lights.

When one traffic light has longer green times or has a lower inflow rates, the occupation rate is lower and more users are sent to this traffic light. Therefore in the scenarios described in Table 11 more users are sent to the first queue, as can be seen in Table 12. In the last scenario more users are sent to traffic light 1 in the beginning of the hour, but at the end more users are sent to the second traffic light. Overall more users are sent to queue 1, since only in the last quarter traffic light 2 has a lower occupation rate.

Scenario	σ	l	Waiting time per user	number users scheduled 1	number users scheduled 2	number users on time	number users too late
1	0.25	3	0.3636	151.8	105.4	254.2	3.0
	0.50	3	0.3200	269.7	236.6	500.2	6.2
	0.75	3	0.2533	387.4	363.7	739.9	11.2
2	0.25	3	1.1449	229.9	82.6	310.3	2.2
	0.50	3	1.0871	352.5	169.2	518,5	3.2
	0.75	3	1.0716	521.6	296.9	816.8	1.7
3	0.25	3	0.6707	176.4	122.5	293.3	5.6
	0.50	3	0.5422	316.2	248.0	554.3	9.9
	0.75	3	0.4868	451.7	373.7	810.0	15.4

Table 12: Results of the simulations of the scenarios described in Table 11.

In total the mean queue lengths of the queues will decrease, when the users are optimal distributed over the traffic lights. As can be seen in Figure 38, the queue lengths of the second queues decrease a lot when the fraction of users increases, while the first queues only increase a little bit. Hence the higher the fraction of users, the lower the waiting times and mean queue lengths. The biggest improvements are made by distributing $\sigma = 1/4$ of the citizens, since in this case the occupation rates of both traffic lights can already be made more similar.

In Figure 39 the mean queue lengths are shown for two queues with on queue 1 scenario high and on queue 2 a steady scenario with inflow $\lambda = 10$ on all intervals and an outflow of 12 vehicles per minute. Compared to the results in the previous paragraph the queue length of scenario high decreases a lot by including the second traffic light. Also the waiting time per person decreases a lot, from 0.75 ($\sigma = 1/4$) to 0.60 ($\sigma = 1/4$) until 0.54 minute ($\sigma = 3/4$).

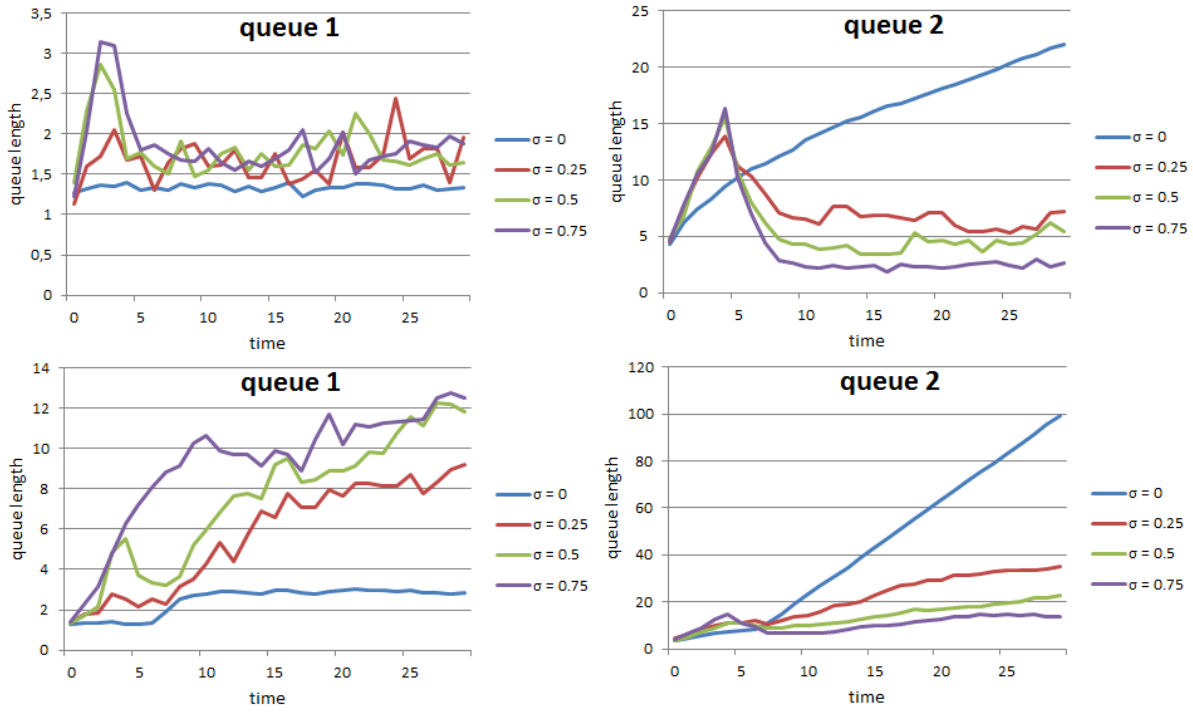


Figure 38: Mean queue lengths of the two queues of scenario 1 (top figures) and scenario 2 (bottom figures) for different fractions of users.

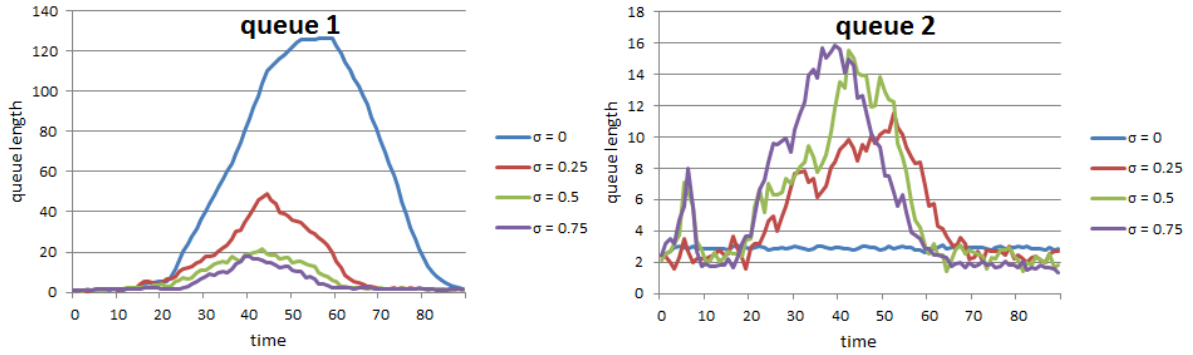


Figure 39: Mean queue lengths of two queues with on queue 1 the high scenario and on queue 2 a steady scenario of $\lambda = 10$ and $\mu = 12$ for different fraction of users ($l = 3$).

10 Conclusion

The matrix approach appears to be the best method to calculate the time-dependent queue length distribution. This method is much faster than the theoretical models found in the literature and can easily deal with time-dependent arrival rates and overloaded systems. If few citizens use the app, the time-dependent queue length distribution is used to calculate the optimal arrival time such that the user is with confidence level $1 - \alpha$ on time out of the queue. The route with the shortest travel time is advised to the user.

In the matrix approach we assumed exponential departure times instead of deterministic departure times. Due to high randomness in the number of departures, this method overestimates the real mean queue length a little bit. For the calculation of the confidence bound this is no problem, since the upper bound is overestimated: even more users will be out of the queue before their deadline.

When a significant fraction of citizens use the app, the users can be distributed over both available traffic lights and time slots. First, the users are distributed over the traffic lights depending on the total travel time of the routes corresponding to the traffic lights. This distribution will be determined by a local search algorithm, since the distributed users influence each other's waiting times at the traffic lights. As a result the users are distributed over the traffic lights such that the waiting times are minimized and more users are sent to the fastest or least congested queue. Therefore the occupation rates of both traffic lights will be more similar, when the fraction of users increases.

Secondly per traffic light the users are scheduled over time by minimizing the value function of the total waiting times of the app users, as defined in Definition 7.1. We proved that the mean queue length calculated by the matrix approach is convex in η, this resulted in a convex value function. Hence, the optimal solution of the value function can be found by a local search algorithm.

The matrix approach used to calculate the mean queue length in the value function, is too slow in the calculation of a real time problem with multiple users. Therefore a new fast method is developed to approximate the mean queue length in time. This method is not convex in η, but in most parameter settings the local search algorithm finds a schedule really close to the optimal schedule. Therefore this method is used in the local search algorithm, since it has an optimal ratio between calculation time and performance.

The optimal schedules result in lower mean queue lengths and shorter waiting times, with users being with confidence level $1 - \alpha$ on time. The higher the fractions and the longer the distributing intervals, the lower the waiting times and queue lengths. Since the user wants to arrive at a certain time at the destination the user can be scheduled in maximum l intervals. Due to this constraint not all congestions will disappear completely, but the mean waiting times and queue lengths will always decrease. Since all users have to be on time, more users have to be scheduled in a smaller time period when the σ value increases. Therefore in overloaded systems the mean queue length will not always improve when σ increases and l stays the same. Increasing the number of distributing intervals l will eliminate this problem.

11 Extensions and Further Research

In this section a couple of extensions of the model will be discussed. Most of the extensions can be easily added to our model, but for some applications further research is needed.

11.1 Flexible users

In general there are two types of users:

- Type 1: users that want to arrive at a certain time.
- Type 2: users that want to arrive during an interval.

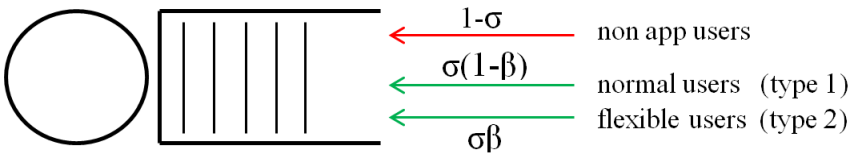


Figure 40: Fractions of the different types of arriving vehicles at a traffic light. Here σ is the fraction app users and a fraction β of these users is flexible.

Until now only the first type of users are discussed. In this section we will show how the second type of users can be included into the model.

In the app the user has the option to define an interval in which he wants to arrive. For example a user with flexible working hours can use this feature to indicate that he wants to arrive between 8 am and 9 am. The app should return a departure advice such that the travel time is as short as possible and the user arrives in the preferred interval at the destination.

A user wants to know beforehand at what time he has to leave, such that the user has the possibility to adapt to the departure time. For example a user must be able to adjust the waking alarm to the departure time. Therefore the app should give a preliminary departure time, such that the user can take this into account. The final advised time should not differ much from the preliminary departure time otherwise the preliminary advice has no use. (Of course it can happen that due to a large congestion it will differ a lot, but in that occasional case the user will understand).

Suppose that a user wants to arrive at the destination between T_1^{dest} and T_2^{dest}, with $T_1^{dest} < T_2^{dest}$. To arrive at the destination during interval $[T_1^{dest}, T_2^{dest}]$, the user should depart in interval $[D_1, D_2]$ from the queue at the traffic light. (This will be calculated by the travel time forecasting model.) Hence the user should be out of the queue before deadline D_2. The latest arrival interval A_2 for this deadline should be calculated such that the user is with confidence level $1 - \alpha$ before D_2 out of the queue. Let $A^{max} = A_2$ be the latest arrival interval in which the user can be scheduled. The user wants to arrive after T_1^{dest}, so calculate A_1 the latest arrival interval such that the user is with a confidence level of 50% before D_1 out of the queue. Let $A^{min} = A_1$ be the earliest interval in which the user can be scheduled. Hence the user has be scheduled in an interval between A^{min} and A^{max}.

It is optimal to schedule a flexible user in the interval with the lowest queue length, call this interval A^{opt}. Note that $A^{min} \leq A^{opt} \leq A^{max}$. If there is only a single flexible user the queue length can be predicted by Method 3 and the interval with the lowest queue length is chosen. If there are multiple flexible users, the flexible users will affect each other's waiting time, therefore an other method is needed. The flexible user can for example be distributed by a local search algorithm to minimize the waiting time of the flexible users. Or by minimizing $(\xi_i + \eta_i^1/\Delta x + \eta_i^2/\Delta x)^2$ over η^2, with η_i^1 the number of users of type 1 which have latest arrival interval i and η_i^2 the number of flexible users with optimal arrival interval i.

The optimal arrival interval A^{opt} can be found for the queues at traffic lights $1, 2, \ldots, S$. The optimal arrival interval corresponding to the route with the shortest travel time will be given as preliminary advice. Twelve hours before departure this preliminary departure advice is given to include as many not flexible η^1 users as possible. Of course the optimal departure time can be recalculated, if demanded by the user.

The final departure time of the flexible user should not deviate too much from the preliminary advice, for example maximum 10 minutes. Therefore the final arrival interval i should lie between the bounds $\max(A^{min}, A^{opt} - \frac{10}{\Delta x})$ and $\min(A^{max}, A^{opt} + \frac{10}{\Delta x})$. These bounds also ensure that the arrival time lies between A^{min} and A^{max}. If for example $\Delta x = 2$, the final arrival interval i lies between $\max(A^{min}, A^{opt} - 5) \leq i \leq \min(A^{max}, A^{opt} + 5)$.

The final departure advices will be calculated in the same way as for the normal users of type 1, by solving an optimization problem with the local search algorithm. The flexible users are easily included in the optimization problem given in Definition 7.1, only the constraints still have to be set.

When few users are flexible the bounds defined above can be used as the constraints for the flexible users. If multiple users are flexible, we want a more general approach. Let NU_i^2 be the number of users which have interval i as final upper bound, and let NL_i^2 be the number of users with interval i as lower bound. This can be translated in the following constraints for the flexible users:

$$\sum_{j=1}^{i} \eta_j^2 \geq \sum_{k=1}^{i} NU_k^2 \text{ for } \forall i \in \{1, \ldots, n\}$$
$$\sum_{j=1}^{i} \eta_j^2 \leq \sum_{k=1}^{i} NL_k^2 \text{ for } \forall i \in \{1, \ldots, n\}$$
$$\eta_i^2 \geq 0 \text{ for } i = 1, 2, \ldots, n.$$

The first constraint ensures that all users are finally scheduled before the upper bound and the second constraints ensures that all users are scheduled after the lower bound.

The constraints for type 1 users are already given in Definition 7.1. Hence, the value function given in Definition 7.1 can be adjusted as follows such that flexible users are inserted.

Definition 11.1. *The value function to find the optimal schedule for multiple users of type 1 and 2 with multiple deadlines is given by:*

$$
\begin{aligned}
\underset{\eta}{\textit{minimize}} \quad & V(x,\eta) = \sum_{i=1}^{n} \eta_i \frac{L_{i-1}(x,\lambda) + L_i(x,\lambda)}{2\mu} + \frac{1}{\mu} N_n L_n(x,\lambda) \\
\textit{subject to} \quad & \sum_{j=1}^{i} \eta_j^1 \geq \sum_{k=1}^{i} N_k^1 \ \textit{for} \ \forall i \in \{l,\ldots,n\} \\
& \sum_{j=1}^{i-l+1} \eta_j^1 \leq \sum_{k=1}^{i} N_k^1 \ \textit{for} \ \forall i \in \{l,\ldots,n\} \\
& \sum_{j=1}^{i} \eta_j^2 \geq \sum_{k=1}^{i} NU_k^2 \ \textit{for} \ \forall i \in \{1,\ldots,n\} \\
& \sum_{i=1}^{i} \eta_j^2 \leq \sum_{k=1}^{i} NL_k^2 \ \textit{for} \ \forall i \in \{1,\ldots,n\} \\
& \eta_i^1, \eta_i^2 \geq 0 \ \textit{for} \ i = 1,2,\ldots,n.
\end{aligned}
$$

With $\eta_i = \eta_i^1 + \eta_i^2$, $\eta = (\eta_1, \eta_2, \ldots, \eta_n)$ *and* $\lambda = \xi + \frac{\eta}{\Delta x}$.

With the results from Section 6.3 it is easily shown that Definition 11.1 is a convex optimization problem.

11.2 Global optimum

In this thesis optimal schedules are found such that the waiting times of the users are minimized. As an additional result we have seen that the mean queue lengths are lower under these optimal schedules, therefore also the waiting times of the non app users decrease.

In some cases it is the goal to find a global optimum, i.e., minimizing the waiting times of all road users. In this case the users are distributed over the time intervals such that the total waiting time of all road users is minimized. The corresponding optimization problem is given below:

Definition 11.2. *The value function to find the global optimum for multiple users with multiple deadlines is given by:*

$$
\begin{aligned}
\underset{\eta}{\textit{minimize}} \quad & V(x,\eta) = \sum_{i=1}^{n} \lambda_i \frac{L_{i-1}(x,\lambda) + L_i(x,\lambda)}{2\mu} + \frac{1}{\mu} \lambda_n L_n(x,\lambda) \\
\textit{subject to} \quad & \sum_{j=1}^{i} \eta_j \geq \sum_{k=1}^{i} N_k \ \textit{for} \ \forall i \in \{l,\ldots,n\} \\
& \sum_{j=1}^{i-l+1} \eta_j \leq \sum_{k=1}^{i} N_k \ \textit{for} \ \forall i \in \{l,\ldots,n\} \\
& \eta_i \geq 0 \ \textit{for} \ i = 1,2,\ldots,n.
\end{aligned}
$$

With $\eta = (\eta_1, \eta_2, \ldots, \eta_n)$ *and* $\lambda = \xi + \frac{\eta}{\Delta x}$.

This is also a convex optimization problem, since the value function is a combination of the value function of Definition 7.1 and additional terms of $L_i(x, \lambda)$. The value function in Definition 7.1 and $L_i(x, \lambda)$ are both convex in η, hence the value function in Definition 11.2 is convex in η.

11.3 Green times

In this thesis fixed green times are used, but in reality this is not true for all traffic lights. There are different kinds of traffic lights, which all have different programs to set their green times. Some traffic lights have only one fixed green time or use a day pattern. For example, in the case of day patterns the headway will get a longer green time during rush hour.

If the changes in green times are known, these can easily be inserted into our model. In that case not only the inflow parameter λ changes over time, but also the departure rate μ is a function in time. Changing the μ values in the calculations, works in general the same as for the λ values. By the calculations of the latest arrival times with the intersection method the outflow is important, since the line has as slope the outflow parameter μ. Hence when the μ value changes there will be a node in this line, but the remainder of the approach is exactly the same.

There are also smart traffic lights, which change the green times based on the current traffic conditions, so in that case μ is a state-dependent variable. For example, when a loop that detects traffic jams is occupied for more than 10 minutes, a request can be send to increase the green time. The state-dependent green times are difficult to simulate accurately, since whether the request is approved also depends on the other routes at the intersection. Further research has to show how state-dependent green times can be inserted in an accurate way.

If this problem of variable green times is solved, an advice about the optimal green times can be given as discussed in the next paragraph.

11.3.1 Advised green times

If a traffic light can handle the demand of arriving vehicles, the green time is enough and no large queue will develop at the traffic light. But in overloaded systems as during rush hours in IJburg, the green time is not enough. In that case the queue length can decrease by giving the optimal green times.

By the parameter $\lambda(t)$ the current and expected inflow of vehicles is known, so the outflow should be higher otherwise a queue will develop. Based on this inflow parameter the minimum outflow per minute can be calculated such that there will be no congestion at the traffic light. This outflow should be a bit higher than $\lambda(t)$. The minimum outflow will be our advice to the traffic center, where it can be translated into a preferred green which is enough to handle the expected demand. If this advice is followed depends also on the other roads connected to the intersection.
If a direction gets more green time than needed, the minimum flow advice can also be handy, since more green time can be given to congested directions.

11.4 Inflow data and initial queue length

Next to the green times also the inflow variable $\lambda(t)$ should be known for our model. This $\lambda(t)$ represents the number of vehicles arriving at the queue per minute. This variable is used in most traffic and queueing models, but it is difficult to determine in practice. There should be a loop further down the road to count the number of arriving vehicles. In large roads this is often the case, but in most residential areas like IJburg these loops are missing, therefore the parameter is difficult to set.

There are two options to approximate the $\lambda(t)$ values without loop information from down the road. The first method uses a loop to measure traffic jams at a traffic light, this loop can be used to determine if there is a jam and for how long. With this information something can be said about the queue length and the inflow in time, but this is not precise. The second method can be used if the outflows from all the roads connected to our main road are known. The inflow at the queue can be predicted by adding the outflow of the incoming roads over time. The disadvantage of this method is that the outflow of the incoming roads are not always known nor given in real time. Sometimes the outflow on a lane is known, but at this lane a vehicle can turn in multiple directions, in that case also turn fractions are needed. Future research must show if one of these methods can be used to predict $\lambda(t)$.

A corresponding problem is that the initial queue length is not always known. If the inflow and outflow in time are known the current queue length can be calculated. But even the outflow at a traffic light is not given in real time, so with the current data it is difficult to calculate the initial queue length. Another option is to use special cameras which can measure the queue length and inflow at a road, but these cameras are expensive and not often used in the Netherlands.

11.5 General situation

This thesis is written for residential areas, but the developed methods can also be used in other situations. As shown in this section it is hard to set the parameters needed for our model. Therefore it might be useful to look at the traffic lights at larger roads, were the parameters can more easily be set.

We looked at journeys with as origin a location in a residential area, this is easier since the travel time to the traffic light is short. If this travel time is long, the prediction has to be done over a longer interval which makes the prediction less accurate. The travel time to the traffic light can be shorter or larger than predicted due to all different kinds of congestions. Due to these uncertainties distributing the users in intervals of length Δx is less useful, since the arrival times will not be accurate. You can also wonder how much influence the traffic light will have in the case of a long journey, since there are so many different factors influencing the travel time.

Of course in all cases the predicted waiting time can be taken into account by calculating the travel time. This can be used on all major intersections, so not only the ones in residential area. Including the waiting times in the travel time forecasting model will improve the prediction especially on congested routes. Also distributing the flexible users still works. They are scheduled long before departure, so the effect of future changes in travel time are the same as for residential areas.

Appedix A: additional proof convex value function

In the proof of the convex value function in Section 6.3 the initial state $x = 0$ is omitted. The proof for this initial state is almost the same as for $x > 1$, but for completeness the lemmas and theorem from Section 6.3 are also proven for $x = 0$. All proofs are with induction on t and the base cases are already shown in Section 6.3.

Lemma 6.3: $L_t(x+1,\lambda) - 2L_t(x,\lambda) + L_t(x-1,\lambda) \geq 0$ for $\forall t \in \mathbb{N}$ and $\forall x \in \mathbb{N}_{>0}$.

Suppose the lemma holds for $t = kC + \tau$ with $0 \leq \tau < C$, we will now show that it also holds for $t+1$ and $x = 0$, by writing out the recursive formulas.

$$
\begin{aligned}
&L_{t+1}(2,\lambda) - 2L_{t+1}(1,\lambda) + L_{t+1}(0,\lambda) = \\
&\quad = \lambda_{k+1}L_t(3,\lambda) + (C - \lambda_{k+1} - \mu)L_t(2,\lambda) + \mu L_t(1,\lambda) - 2\lambda_{k+1}L_t(2,\lambda) - 2(C - \lambda_{k+1} - \mu)L_t(1,\lambda) \\
&\quad - 2\mu L_t(0,\lambda) + \lambda_{k+1}L_t(1,\lambda) + (C - \lambda_{k+1})L_t(0,\lambda) \\
&\quad = \lambda_{k+1}(L_t(3,\lambda) - 2L_t(2,\lambda) + L_t(1,\lambda)) + (C - \lambda_{k+1} - \mu)(L_t(2,\lambda) - 2L_t(1,\lambda) + L_t(0,\lambda)) \\
&\quad + \mu(L_t(1,\lambda) - L_t(0,\lambda)) \geq 0
\end{aligned}
$$

The last inequality holds by (10) and since we assumed that the lemma holds for t and $\forall x \in \mathbb{N}_{>0}$.

Lemma 6.4: $\frac{\partial}{\partial \lambda_r} L_t(x+1,\lambda) - \frac{\partial}{\partial \lambda_r} L_t(x,\lambda) \geq 0$ for $\forall x, t \in \mathbb{N}$ and $\forall 1 \leq r \leq n$.

Suppose that the theorem holds for $t = kC + \tau$ with $k \in \mathbb{N}$ and $0 \leq \tau < C$:

$$
\frac{\partial}{\partial \lambda_r} L_t(x+1,\lambda) - \frac{\partial}{\partial \lambda_r} L_t(x,\lambda) \geq 0 \text{ for } \forall x \in \mathbb{N} \text{ and } \forall 1 \leq r \leq n \tag{14}
$$

We will show that the theorem also holds for $t+1$ and $x = 0$.

By Corollary 6.2 $L_{t+1}(0,\lambda)$ does not depend on λ_r with $r > k+1$, i.e., $\frac{\partial}{\partial \lambda_r} L_{t+1}(0,\lambda) = 0$ for $r > k+1$, so the lemma holds for $r > k+1$. We now have to show that the lemma also holds for $r \leq k+1$.

First we will look at $r = k+1$ for which two cases are distinguished:
Case 1: $1 \leq \tau < C$ then $1 < \tau + 1 \leq C$, so $L_{kC+\tau} = L_t$ and $L_{kC+\tau+1} = L_{t+1}$ depend on the same λ values, λ_r with $r \leq k+1$ (see Corollary 6.2).
Case 2: $\tau = 0$ then $\tau + 1 = 1$, here $L_{kC} = L_t$ depends on $\lambda_1, \ldots, \lambda_k$ and $L_{kC+1} = L_{t+1}$ depends on $\lambda_1, \ldots, \lambda_k, \lambda_{k+1}$, so the derivative of L_t to λ_{k+1} is zero. In this case the computations can be simplified.

We will first prove the lemma for case 1 of $r = k+1$ by filling in the recursive formulas (9).

Case 1: $t = kC + \tau$ with $1 \leq \tau < C$ and $k \in \mathbb{N}$

$$\begin{aligned}
&\frac{\partial}{\partial\lambda_{k+1}} L_{t+1}(1,\lambda) - \frac{\partial}{\partial\lambda_{k+1}} L_{t+1}(0,\lambda) = \\
&\quad = \frac{\partial}{\partial\lambda_{k+1}} (\lambda_{k+1} L_t(2,\lambda) + (C - \lambda_{k+1} - \mu) L_t(1,\lambda) + \mu L_t(0,\lambda)) \\
&\quad - \frac{\partial}{\partial\lambda_{k+1}} (\lambda_{k+1} L_t(1,\lambda) + (C - \lambda_{k+1}) L_t(0,\lambda)) \\
&\quad = \lambda_{k+1} \frac{\partial}{\partial\lambda_{k+1}} L_t(2,\lambda) + (C - \lambda_{k+1} - \mu) \frac{\partial}{\partial\lambda_{k+1}} L_t(1,\lambda) + \mu \frac{\partial}{\partial\lambda_{k+1}} L_t(0,\lambda) + L_t(2,\lambda_1) - L_t(1,\lambda_1) \\
&\quad - \left(\lambda_{k+1} \frac{\partial}{\partial\lambda_{k+1}} L_t(1,\lambda) + (C - \lambda_{k+1}) \frac{\partial}{\partial\lambda_{k+1}} L_t(0,\lambda) + L_t(1,\lambda_1) - L_t(0,\lambda_1) \right) \\
&\quad = \lambda_{k+1} \left(\frac{\partial}{\partial\lambda_{k+1}} L_t(2,\lambda) - \frac{\partial}{\partial\lambda_{k+1}} L_t(1,\lambda) \right) + (C - \lambda_{k+1} - \mu) \left(\frac{\partial}{\partial\lambda_{k+1}} L_t(1,\lambda) - \frac{\partial}{\partial\lambda_{k+1}} L_t(0,\lambda) \right) \\
&\quad + L_t(2,\lambda) - 2L_t(1,\lambda) + L_t(0,\lambda) \geq 0
\end{aligned}$$

By the induction hypothesis (14) and Lemma 6.3 the lemma holds for case 1 of $r = k+1$ and $x = 0$. For case 2 a similar proof is given below.

Case 2: $t = kC$ with $k \in \mathbb{N}$

$$\begin{aligned}
&\frac{\partial}{\partial\lambda_{k+1}} L_{t+1}(1,\lambda) - \frac{\partial}{\partial\lambda_{k+1}} L_{t+1}(0,\lambda) = \\
&\quad = \lambda_{k+1} \frac{\partial}{\partial\lambda_{k+1}} L_t(2,\lambda) + (C - \lambda_{k+1} - \mu) \frac{\partial}{\partial\lambda_{k+1}} L_t(1,\lambda) + \mu \frac{\partial}{\partial\lambda_{k+1}} L_t(0,\lambda) \\
&\quad + L_t(2,\lambda) - L_t(1,\lambda) \\
&\quad - \left(\lambda_{k+1} \frac{\partial}{\partial\lambda_{k+1}} L_t(1,\lambda) + (C - \lambda_{k+1}) \frac{\partial}{\partial\lambda_{k+1}} L_t(0,\lambda) + L_t(1,\lambda) - L_t(0,\lambda) \right) \\
&\quad = L_t(2,\lambda) - 2L_t(1,\lambda) + L_t(0,\lambda) \geq 0
\end{aligned}$$

Since $L_{kC}(0,\lambda)$ does not depend on $\lambda_{k+1} \Rightarrow \frac{\partial}{\partial\lambda_{k+1}} L_t(0,\lambda) = 0$ and by Lemma 6.3 the last inequality holds. Hence, Lemma 6.4 is also true for case 2 of $r = k+1$, therefore the lemma holds for $t+1$ with $r = k+1$ and $x = 0$.

The only step left is to show that the lemma also holds for $r < k+1$:

$$\begin{aligned}
&\frac{\partial}{\partial\lambda_r}L_{t+1}(1,\lambda) - \frac{\partial}{\partial\lambda_r}L_{t+1}(0,\lambda) = \\
&\quad = \lambda_{k+1}\frac{\partial}{\partial\lambda_r}L_t(2,\lambda) + (C-\lambda_{k+1}-\mu)\frac{\partial}{\partial\lambda_r}L_t(1,\lambda) + \mu\frac{\partial}{\partial\lambda_r}L_t(0,\lambda) \\
&\quad - \left(\lambda_{k+1}\frac{\partial}{\partial\lambda_r}L_t(1,\lambda) + (C-\lambda_{k+1})\frac{\partial}{\partial\lambda_r}L_t(0,\lambda)\right) \\
&\quad = \lambda_{k+1}\left(\frac{\partial}{\partial\lambda_{k+1}}L_t(2,\lambda) - \frac{\partial}{\partial\lambda_{k+1}}L_t(1,\lambda)\right) + (C-\lambda_{k+1}-\mu)\left(\frac{\partial}{\partial\lambda_{k+1}}L_t(1,\lambda) - \frac{\partial}{\partial\lambda_{k+1}}L_t(0,\lambda)\right) \geq 0
\end{aligned}$$

Hence by induction hypothesis (14) the lemma is also true for $r < k+1$, therefore the lemma holds for $t+1$. By induction on t, Lemma 6.4 holds for $x = 0$ and $\forall t \in \mathbb{N}$, $\forall 1 \leq r \leq n$.

Theorem 6.5: $L_t(x,\lambda)$ is increasing and convex in $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n)$ for $\forall t \in \mathbb{N}$

Suppose that the theorem holds for $t = kC + \tau$ with $k \in \mathbb{N}$ and $0 \leq \tau < C$:

$$\frac{\partial}{\partial\lambda_r}L_t(x,\lambda) \geq 0 \text{ and } \frac{\partial^2}{\partial\lambda_r^2}L_t(x,\lambda) \geq 0 \text{ for } 1 \leq r \leq n \text{ and } \forall x \in \mathbb{N}. \tag{15}$$

We will show that the theorem is also true for $t+1$ and $x = 0$.
By Corollary 6.2, $L_{t+1}(0,\lambda)$ does not depend on λ_r with $r > k+1$, i.e., $\frac{\partial}{\partial\lambda_r}L_{t+1}(0,\lambda) = 0$ for $\forall r > k+1$, so the lemma holds for $r > k+1$. Now we have to show that the lemma also holds for $r \leq k+1$.

For $r = k+1$ the two cases defined in the proof of Lemma 6.4 are again distinguished. We will prove that the lemma holds in both cases by using the recursive formulas.

Case 1: $t = kC + \tau$ with $1 \leq \tau < C$ and $k \in \mathbb{N}$

$$\begin{aligned}
L_{t+1}(0,\lambda) =& \lambda_{k+1}L_t(1,\lambda) + (C-\lambda_{k+1})L_t(0,\lambda) \\
\frac{\partial}{\partial\lambda_{k+1}}L_{t+1}(0,\lambda) =& \lambda_{k+1}\frac{\partial}{\partial\lambda_{k+1}}L_t(1,\lambda) + (C-\lambda_{k+1})\frac{\partial}{\partial\lambda_{k+1}}L_t(0,\lambda) + L_t(1,\lambda) - L_t(0,\lambda) \\
\frac{\partial^2}{\partial\lambda_{k+1}^2}L_{t+1}(0,\lambda) =& \lambda_{k+1}\frac{\partial^2}{\partial\lambda_{k+1}^2}L_t(1,\lambda) + (C-\lambda_{k+1})\frac{\partial^2}{\partial\lambda_{k+1}^2}L_t(0,\lambda) + 2\frac{\partial}{\partial\lambda_{k+1}}L_t(1,\lambda) - 2\frac{\partial}{\partial\lambda_{k+1}}L_t(0,\lambda)
\end{aligned}$$

By the induction hypothesis (15) and inequality (10) $\Rightarrow \frac{\partial}{\partial\lambda_{k+1}}L_{t+1}(0,\lambda) > 0$ for $\forall x \in \mathbb{N}$. Due to assumption (15) and Lemma 6.4 $\Rightarrow \frac{\partial^2}{\partial\lambda_{k+1}^2}L_{t+1}(0,\lambda) \geq 0$ for $\forall x \in \mathbb{N}$. Hence for case 1 of $r = k+1$ the lemma is true. The proof of case 2 is given below.

Case 2: $t = kC$ with $k \in \mathbb{N}$

$$\begin{aligned} L_{t+1}(0,\lambda) =& \lambda_{k+1} L_t(1,\lambda) + (C-\lambda) L_t(0,\lambda) \\ \frac{\partial}{\partial \lambda_{k+1}} L_{t+1}(0,\lambda) =& L_t(1,\lambda) - L_t(0,\lambda) \\ \frac{\partial^2}{\partial \lambda_{k+1}^2} L_{t+1}(0,\lambda) =& 0 \end{aligned}$$

By (10) $L_t(1,\lambda) - L_t(0,\lambda) > 0$, hence the lemma also holds for case 2 of $r = k+1$ and $x = 0$. The only thing left is to show that the theorem also holds for $t+1$ and λ_r with $r < k+1$.

For $r < k+1$

$$\begin{aligned} L_{t+1}(0,\lambda) &= \lambda_{k+1} L_t(1,\lambda_k) + (C-\lambda_{k+1}) L_t(0,\lambda) \\ \frac{\partial}{\partial \lambda_r} L_{t+1}(0,\lambda) &= \lambda_{k+1} \frac{\partial}{\partial \lambda_r} L_t(1,\lambda) + (C-\lambda_{k+1}) \frac{\partial}{\partial \lambda_r} L_t(0,\lambda) \\ \frac{\partial^2}{\partial \lambda_r^2} L_{t+1}(0,\lambda) &= \lambda_{k+1} \frac{\partial^2}{\partial \lambda_r^2} L_t(1,\lambda) + (C-\lambda_{k+1}) \frac{\partial^2}{\partial \lambda_r^2} L_t(0,\lambda) \end{aligned}$$

By assumption (15) $\Rightarrow \frac{\partial}{\partial \lambda_r} L_{t+1}(x,\lambda) \geq 0$ and $\frac{\partial^2}{\partial \lambda_r^2} L_{t+1}(x,\lambda) \geq 0$ for $\forall 1 \leq r \leq n$. Therefore the theorem also holds for $t+1$ with $x = 0$ and λ_r with $r < k+1$. Hence the theorem is true for $t+1$ and by induction on t the theorem holds for $\forall t \in \mathbb{N}$.

Appendix B: mean queue length of an M/M/1/N queue

In this appendix is shown that the mean queue length is not convex for an M/M/1/N queue. The first time steps with time interval $\frac{1}{C}$ are given below.

For $\forall x \in \{1, \ldots, N\}$

$$L_0(x, \lambda) = x$$

$$L_1(N, \lambda) = (C - \mu)L_0(N, \lambda) + \mu L_0(N - 1, \lambda) = CN - \mu$$

For $\forall x \in \{2, \ldots, N - 1\}$

$$L_1(x, \lambda) = \lambda_1 L_0(x + 1, \lambda) + \mu L_0(x - 1, \lambda) + (C - \lambda_1 - \mu)L_0(x, \lambda) = Cx - \mu + \lambda$$

$$\begin{aligned} L_2(N, \lambda) &= (C - \mu)L_1(N, \lambda) + \mu L_1(N - 1, \lambda) = (C - \mu)(CN - \mu) + \mu(CN - C - \mu + \lambda) \\ &= C^2N - 2C\mu + \lambda\mu \\ \frac{\partial}{\partial \lambda_1} L_2(N, \lambda) &= \mu \\ \frac{\partial^2}{\partial \lambda_1^2} L_2(N, \lambda) &= 0 \end{aligned}$$

$$\begin{aligned} L_2(N - 1, \lambda) &= \lambda_1 L_1(N, \lambda) + \mu L_1(N - 2, \lambda) + (C - \lambda_1 - \mu)L_1(N - 1, \lambda) \\ \frac{\partial}{\partial \lambda_1} L_2(N - 1, \lambda) &= \lambda_1 \frac{\partial}{\partial \lambda_1} L_1(N, \lambda) + \mu \frac{\partial}{\partial \lambda_1} L_1(N - 2, \lambda) + (C - \lambda_1 - \mu)\frac{\partial}{\partial \lambda_1} L_1(N - 1, \lambda) \\ &\quad + L_1(N, \lambda) - L_1(N - 1, \lambda) = 2C - C\lambda \\ \frac{\partial^2}{\partial \lambda_1^2} L_2(N - 1, \lambda) &= 0 \end{aligned}$$

$$\begin{aligned} L_3(N, \lambda) &= (c - \mu)L_2(N, \lambda) + \mu L_2(N - 1, \lambda) \\ \frac{\partial}{\partial \lambda_1} L_3(N, \lambda) &= (c - \mu)\frac{\partial}{\partial \lambda_1} L(N, \lambda) + \mu \frac{\partial}{\partial \lambda_1} L_2(N - 1, \lambda) = 3C\mu - \mu^2 - 2\lambda\mu \\ \frac{\partial^2}{\partial \lambda_1^2} L_3(N, \lambda) &= -2\mu \end{aligned}$$

Hence $\frac{\partial^2}{\partial \lambda_1^2} L_t(x, \lambda) \leq 0$ for some values of x and t, therefore $L_t(x, \lambda)$ is not convex in λ.

Appendix C: schedules different methods

In this appendix some optimal schedules found by the local search algorithm described in Section 7 are presented. For the four different methods described in Section 7.1 the optimal schedules are given in the tables below for different parameter settings.

In these results 10 consecutive arrival intervals have been used with $l = 3$ and the interval length to schedule a user is $\Delta x = \Delta t = 2$. The length of the fixed arrival rates is $\Delta T = 10$ and let σ be the fraction of citizens that use the app. The inflow of non app users ξ should be adapted to the value of σ, for example if the fixed total inflow is $\Lambda = (12, 10, 8)$ and $\sigma = 1/4$, then $\xi = (1 - \sigma)\Lambda = (9, 7.5, 6)^{12}$. The new total inflow per interval in vehicles per minute used in the Methods is given by $\lambda = \xi + \frac{\eta}{\Delta x}$.

$i = 5$, $\Lambda = (10, 11, 12)$ and $\sigma = 1/4$

N_i	0	0	4	5	5	6	5	6	6	5	6	7		
Method	Schedule												Time	Value
1	4	5	5	6	5	5	4	4	4	5	4	4	177.90	28.97
2	4	5	3	8	2	6	6	6	7	2	5	1	202.89	30.80
3	3	4	4	5	0	8	8	8	8	4	3	0	0.84	36.95
4	0	9	5	6	5	5	5	5	5	4	3	3	0.44	30.28

$i = 5$, $\Lambda = (10, 11, 12)$ and $\sigma = 1/2$

N_i	0	0	9	10	10	11	10	11	12	12	12	13		
Method	Schedule												Time	Value
1	9	10	10	9	10	9	9	9	9	9	9	8	56.57	47.55
2	9	10	9	10	8	12	12	8	9	9	8	6	39.14	49.71
3	9	10	10	10	10	9	9	9	9	9	9	7	0.85	47.67
4	4	15	10	10	10	9	9	9	9	9	8	8	0.61	50.11

$i = 2$, $\Lambda = (8, 10, 12)$ and $\sigma = 1/4$

N_i	0	0	3	4	4	5	4	5	5	7	6	7		
Method	Schedule												Time	Value
1	3	4	4	5	4	5	5	5	5	5	4	1	337.69	16.70
2	3	4	4	5	4	5	5	5	5	5	0	5	39.11	17.21
3	3	4	4	5	4	5	5	5	5	6	3	1	0.61	16.73
4	0	7	4	5	4	5	5	5	5	5	2	3	0.89	17.17

$i = 2$, $\Lambda = (8, 10, 12)$ and $\sigma = 1/2$

N_i	0	0	8	7	9	9	10	11	11	10	12	13		
Method	Schedule												Time	Value
1	8	7	9	9	10	10	8	9	8	9	8	5	273.81	27.39
2	8	7	9	9	10	9	9	9	9	9	7	5	2.23	27.46
3	8	7	9	9	10	9	9	9	9	9	7	5	0.82	27.46
4	0	15	9	9	10	9	9	9	9	8	6	7	1.13	30.57

$i = 10$, $\Lambda = (12, 10, 8)$ and $\sigma = 1/4$

N_i	0	0	7	6	7	5	5	4	5	4	4	3		
Method	Schedule												Time	Value
1	0	7	3	4	6	5	5	4	5	4	4	3	92.29	29.33
2	0	8	2	5	5	5	5	4	5	4	4	3	25.20	29.38
3	0	7	7	7	5	2	4	3	4	4	4	3	0.14	33.85
4	0	1	9	6	4	5	5	4	5	4	4	3	0.24	30.98

$i = 10$, $\Lambda = (12, 10, 8)$ and $\sigma = 1/2$

N_i	0	0	13	12	10	11	11	10	9	9	7	8		
Method	Schedule												Time	Value
1	3	10	8	8	8	9	11	10	9	9	7	8	129.94	39.16
2	7	8	8	8	8	9	9	10	9	9	8	7	1.39	40.63
3	7	8	8	8	8	9	9	10	9	9	8	7	0.24	40.63
4	1	15	7	7	7	10	10	10	9	9	7	8	0.15	40.28

$i = 20$, $\Lambda = (14, 10, 8)$ and $\sigma = 1/2$

N_i	0	0	14	14	12	11	11	10	10	10	9	8		
Method	Schedule												Time	Value
1	0	8	10	10	12	11	11	10	10	10	9	8	78.13	98.36
2	0	6	9	17	9	10	13	9	9	10	9	8	69.19	103.29
3	0	11	11	11	9	10	10	10	10	10	9	8	0.16	103,28
4	1	15	8	8	8	11	11	10	10	10	9	8	0.11	104.77

$i = 20$, $\Lambda = (14, 10, 8)$ and $\sigma = 1$

N_i	0	0	28	28	26	21	21	20	20	20	17	16		
Method	Schedule												Time	Value
1	3	23	19	20	19	20	20	20	20	20	17	16	74.99	81.99
2	3	24	23	19	19	19	19	19	19	20	17	16	44.32	85.47
3	4	28	19	19	19	19	19	19	19	19	17	16	0.65	88.30
4	3	35	18	18	18	18	18	18	19	19	17	16	0.62	103.84

$i = 4$, $\Lambda = (8, 10, 14)$ and $\sigma = 1/2$

N_i	0	0	8	8	9	10	10	10	12	12	14	15		
Method	Schedule												Time	Value
1	8	8	9	10	10	10	10	10	10	10	8	5	231.16	38.14
2	8	8	9	10	10	8	13	9	10	10	4	9	49.64	39.53
3	8	8	9	10	10	10	10	10	10	11	7	5	0.97	38.19
4	0	16	9	10	10	10	10	10	10	10	6	7	0.80	42.12

[12]Note that $\xi = (9, 7.5, 6)$ is a short notation for $\xi = (9, \ldots, 9, 7.5, \ldots, 7.5, 6, \ldots, 6)$, the inflow of non app users per interval.

$i = 4$, $\Lambda = (8, 10, 14)$ and $\sigma = 1$

N_i	0	0	16	16	17	20	20	20	22	22	28	29		
Method	Schedule												Time	Value
1	16	16	17	19	18	18	18	18	18	18	19	15	255.61	53.03
2	16	16	17	18	18	18	18	18	18	18	19	16	2.84	53.08
3	16	16	17	18	18	18	18	18	18	18	19	16	1.14	53.08
4	7	25	17	18	18	18	18	18	18	18	17	18	0.79	61.11

$i = 5$, $\Lambda = (10, 10, 10)$ and $\sigma = 1/4$

N_i	0	0	6	5	4	5	4	4	3	5	5	7		
Method	Schedule												Time	Value
1	3	5	4	4	4	4	4	4	4	4	5	3	138.52	17.28
2	4	4	4	4	4	4	4	4	4	4	5	3	0.70	17.32
3	4	4	4	4	4	4	4	4	4	4	5	3	0.26	17.32
4	0	8	4	4	4	4	4	4	4	4	4	4	0.32	17.67

$i = 5$, $\Lambda = (10, 10, 10)$ and $\sigma = 1/2$

N_i	0	0	9	10	10	11	10	12	9	8	10	10		
Method	Schedule												Time	Value
1	7	9	8	8	9	8	8	9	8	8	9	8	436.77	29.31
2	8	8	8	9	8	9	8	9	8	8	9	7	1.08	29.42
3	8	8	8	9	8	9	8	9	8	8	9	7	0.44	29.42
4	2	15	8	8	8	8	8	8	8	8	9	9	0.40	31.28

$i = 20$, $\Lambda = (12, 12, 12)$ and $\sigma = 1/4$

N_i	0	0	6	5	5	4	6	7	6	6	7	8		
Method	Schedule												Time	Value
1	0	0	6	5	5	5	5	7	6	6	7	8	1323.44	85.33
2	0	0	11	5	0	13	5	0	13	5	0	8	1490.98	92.87
3	0	0	7	7	7	5	0	7	7	7	7	6	1.23	87.94
4	0	1	10	6	5	6	6	5	6	5	5	5	1.06	92.62

$i = 20$, $\Lambda = (12, 12, 12)$ and $\sigma = 1/2$

N_i	0	0	11	12	13	13	12	10	11	14	12	12		
Method	Schedule												Time	Value
1	0	9	11	10	10	10	11	10	11	14	12	12	633.81	120.42
2	1	11	8	11	6	20	11	6	13	9	12	12	1047.99	129.25
3	0	13	13	4	19	0	12	11	11	13	12	12	0.62	126.24
4	0	3	19	12	11	11	12	11	11	10	10	10	0.41	138.71

Appendix D: simulation and travel time prediction models

To predict the travel times of the app users, a reliable travel time prediction model has to be developed. Based on the predicted travel times the deadlines at the traffic lights and the departure times can be calculated as shown in this thesis.

In this appendix we look at two types of road networks: freeway networks and urban road networks. Freeway networks are the main roads for which a lot of sensor data is available, so on most parts of freeway road the traffic situation can be measured. The junctions at freeways are on/off ramps or merge/bifurcation point of two freeways. On freeway networks delays due to congestion are mostly incurred on links. On the contrary on urban roads the main cause of delay are incurred at intersections. Urban networks are more complex then freeway networks, due to the high intensity of junctions and the complex intersections which are hard to model. Therefore more models are developed for freeway networks than for urban networks. In this appendix the relevant simulation and travel-time prediction models found in the literature for both freeway and urban networks are discussed.

First, we will describe the approach of Trinité to model traffic and give some general background about travel time prediction and different ways to model traffic. In the third paragraph the two main models to predict and simulate traffic on freeways are described, Fastlane and METANET. The models to simulate traffic on urban road networks are described in the following paragraph. In this subsection we discuss the Kashani model, which is the most common used model, and a simple data-driven model. Finally, the described models are discussed and compared.

Approach of Trinité

In most papers a complex road network is divided into several parts. In this paragraph the approach of Trinité of dividing the network in subnetworks, buildingblocks and links is shortly described. For more information see [19]. A subnetwork is a subset of a road network and consists of several buildingblocks. A buildingblock is a stretch of road in one driving direction that can contain junctions. A link is a part of a road in one direction without junctions or other choice points. For now the focus will be on the link level.

Links are homogeneous motorway stretches, i.e., links have uniform characteristics, no on/off-ramps and no major changes in geometry. Therefore a link is bounded by points where the road capacity changes, like a merge or bifurcation point. There are two types of links: mainlinks and accessorlinks. At every major change in characteristics of the road or geometry an accessorlink is placed. Hence a mainlink is the link from merge point to the choice point and an accessorlink is the link from the choice point to the the merge point. The road network can be seen as a set of mainlinks which are connected by accessorlinks. This is graphically shown in Figure 41.

The flow of the mainlink has to be divided over the accessorlinks. If the relation is 1-to-1 than the outflow of the mainlink is directly the inflow of the accessorlink. If there are more accessorlinks connected to the mainlink, the outflow of the mainlink has to be distributed over the accessorlinks, this is done by turn fractions. The inflow of the accessorlink is equal to the outflow of the mainlink times the turn fraction.

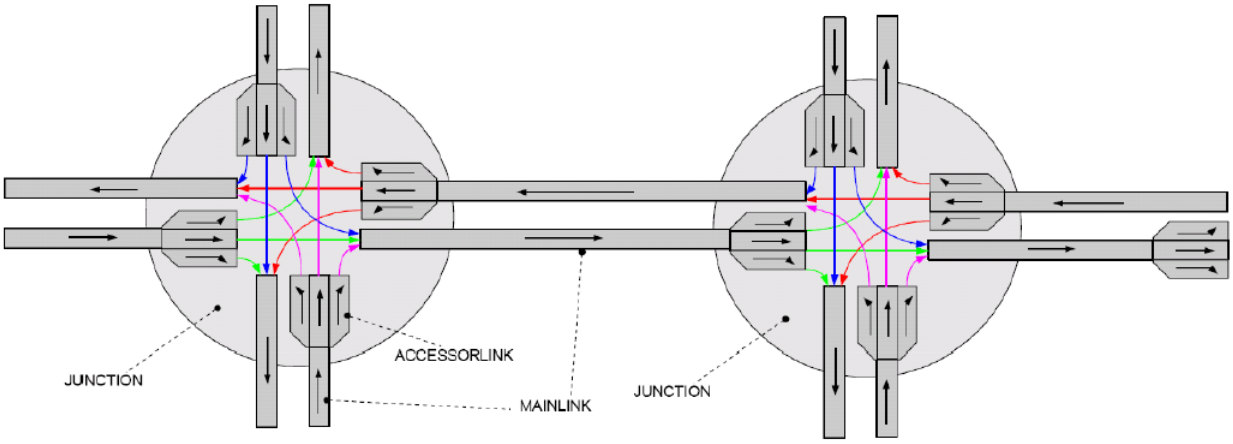


Figure 41: Link structure, source [19].

A mainlink is build up out of segments of 100 meter. In a segment the macroscopic variables, like density, flow, average speed etc, are uniform. Data from the road sensors give information about these macroscopic variables. Not all segments include sensors, so by extrapolation of the measurements the value of the flow and speed of every segment is determined. The reliability of the measurements are also taken into account, less reliable measurements are less heavily counted in the extrapolation.

The flow (velocity) of the link is the average value of the flow (velocity) values of all the segments of the links. The inflow of the link is the flow of the first segment and the outflow is the flow of the last segment of the link.

Background

Prediction horizon

For prediction of travel times it is important to know the prediction horizon. The prediction horizon is the time distance between the current time and the the period for which we calculate the travel time. There are three types of prediction horizons:

- Direct (online): if predictions are made for the current time period, so the prediction horizon is zero.
- Short-term: the travel time is calculated for vehicles that depart in the near future, within 60 minutes.
- Long-term: the travel time is predicted for a vehicle that departures in the future, after 60 minutes or more, for example the following day.

Per type of prediction horizon other data and methods are needed to predict travel times. In general for a longer prediction horizon more assumptions have to be made. For a shorter time horizon real time data is more valuable, since the current travel information has a large impact on the near future traffic conditions.

Overview travel time prediction approaches

There are three types of approaches to predict the travel time: the naïve approach, the data driven approach and simulation. Below these approaches are described and the pros and cons are discussed, for more information see [11].

Naïve approach:
Naïve methods to predict travel time generally do not rely on theoretical relations, but directly use the available data. The naïve predictions that are most commonly used are historical profiles/averages or the instantaneous travel time. Due to the fact that travel time distributions are very wide, i.e., the difference between travel times in a given period is large, predictions based on only historical profiles are not accurate. The current road situation can significantly change in a small period of time, i.e., traffic conditions are not stationary in time, so using the instantaneous travel time is also inaccurate. Despite the low accuracy, naïve approaches are widely used because they are simple and fast.

Data driven approach:
Data driven models use general parameterized mathematical models to calculate the expected travel time over a route from current and historical data. This approach combines the two naive methods, so these models are still fast but also quite accurate. Data-driven models are very suitable for stand-alone traffic applications, its not very suitable for network-wide prediction tasks because this will lead to many parameters.

Simulation method:
For a specific time interval the traffic conditions are simulated by a traffic flow model. Because the traffic conditions in de future are 'known' the travel time can be calculated. In this method the route choice of drivers also have to be taken into account, there are two ways to process the route choice:
Turn fraction: The first and easiest way is to divide the traffic through turn fractions at junctions. The turn fractions are computed by data of the traffic flows at the junction. In this case the route choice of drivers is not modeled at all.
Dynamic traffic assignment: This model includes a route choice model and a dynamic network loading model. The route choice model distributes the routes over the network, these route flows are transferred to the dynamic network model which calculates the travel times. In every situation the traffic will be divided over the roads until a user equilibrium is reached.
The simulation method is suitable for predictions of traffic times and decision support. Because traffic is simulated, traffic conditions that never have occurred before can be modeled. The draw back of this method is that it is time consuming to configure and maintain the model.

The naïve approach is no option for a reliable travel time prediction. The data-driven method is not really suitable for a whole network, because this will result in too much degrees of freedom. For a network the simulation method is the best approach, because in this case we need more assumptions, which are hard to implement in a data-driven method. If the time horizon is small the data-driven method will also be a good option. Both methods need data to calibrate and validate the model. In our case we also want to simulate traffic situations that have never occurred before, so this is in favor of the simulation approach.

In this appendix the focus will lie on simulation methods, but data-driven methods will also be discussed.

Free-flow

The free-flow speed(FFS) is the average speed of the traffic stream when traffic is sparse, low flow and density, and traffic control is not presented. When traffic is sparse drivers are not influenced by the presence of other vehicles, so they can choose the speed at which they feel comfortable traveling under the physical, environmental conditions. By the lack of traffic control the free-flow speed is typically observed in the middle of road segments. The free-flow speed can be measured if there are less than 1300 veh/h/lane on a road [16].

If field measurements of the free-flow speed are unavailable, the FFS can be estimated by adjusting the base FFS, for example the maximum speed limit. Adjustments can be made by the influence of factors that effect the free-flow speed like the number of lanes, lane width, lateral clearance or the number of access points, for more information see [16].

Most of the time a driver does not drive at the FFS, because interactions among vehicles and effect of travel control have effect on the speed. The 85th percentile of the measurements or the maximum speed is often used in travel time calculations, but this is not the FFS.

Fundamental diagrams

The fundamental diagrams describe the theoretical relations between the macroscopic traffic variables, flow q, density ρ and velocity v. These relations are based on two microscopic variables, the time headway and the distance headway. The time headway is the difference between passing times at a point on the road of two successive vehicles. The distance headway is the distance between two successive vehicles at time instant t.

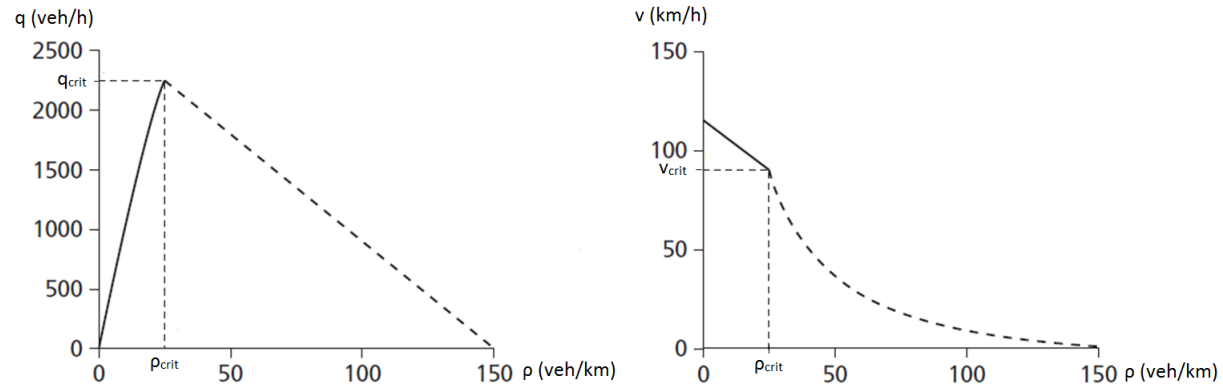


Figure 42: Fundamental diagrams: left the relation between density and flow and right between density and velocity.

In the left graph of Figure 42 the fundamental relation between density and flow is given. In this

figure is shown that at low density the flow increases if the density increases. Due to the minimum distance headway of drivers the flow can not increase infinitely if density increases. The flow increases with the density until the critical point, the capacity flow, is reached. Related to the capacity flow also a critical speed and critical density are defined. After the capacity flow (and corresponding critical density) is reached, the flow decreases until the drivers return to a save headway again.
Something similar occurs in the relation between the density and speed in the right picture of Figure 42. If the density is low, drivers can choose the speed they prefer. When the density increases, the driving behavior of other vehicles and the interaction of vehicles will influence the speed of car drivers. So when density increases the velocity decreases. This is due to the fact that by driving at a lower speed a driver need less distance headway to drive safely and comfortable. When the critical density is reached, the speed of vehicles will take a drop until the headway is save again.

There are many different formulas to describe the fundamental relations shown in Figure 42. In the simulation model Fastlane a formula is used to describe the fundamental relation between speed and density. In the other freeway model, METANET, a different formula is used, but the shape of both formulas are the same.

Freeway models

In this paragraph we describe the two most relevant macroscopic freeway models, Fastlane and METANET.

Fastlane

In Fastlane the road network consists of links and nodes as described in [18]. Links are divided into cells which are similar to the segments in the approach of Trinité. Within a link all cells have the same length, Δx_m(=length of link m/number of cells in link m). In Fastlane the links are also homogeneous, just as described in the approach of Trinité. In Figure 43 the cells in a link are graphically described. In every iteration step of the simulation the density ρ, average speed v and flow q are calculated for every cell.

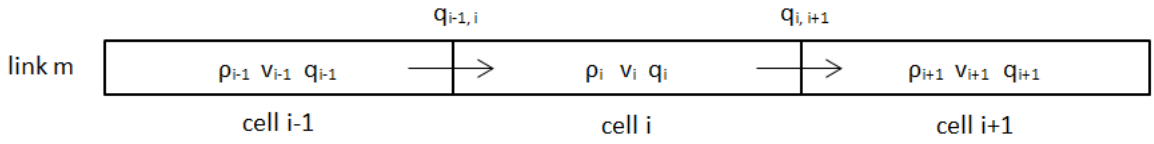


Figure 43: Cell structure of Fastlane.

Derivation of the macroscopic traffic variables

Fastlane is based on the conservation of vehicles equation:

$$\frac{d\rho}{dt} + \frac{dq}{dx} = 0 \tag{16}$$

The conservation of vehicles says that if the density in a link increases then more vehicles flow in

then out, and the other way around:

$$\rho_i(k) > \rho_i(k+1) \iff q_{i-1,i}(k) < q_{i,i+1}(k)$$
$$\rho_i(k) < \rho_i(k+1) \iff q_{i-1,i}(k) > q_{i,i+1}(k)$$

By discretization of equation (16) and using the Euler forward formula we get

$$\begin{aligned}
\frac{\rho_i(k+1) - \rho_i(k)}{\Delta t} + \frac{q_{i,i+1}(k) - q_{i-1,i}(k)}{\Delta x_i} &= 0 \\
\frac{\rho_i(k+1) - \rho_i(k)}{\Delta t} &= \frac{q_{i-1,i}(k) - q_{i,i+1}(k)}{\Delta x_i} \\
\frac{\rho_i(k+1) - \rho_i(k)}{\Delta t} &= \frac{q_{i-1,i}(k) - q_{i,i+1}(k)}{\Delta x_i} \\
\rho_i(k+1) &= \rho_i(k) + \frac{\Delta t}{\Delta x_i}(q_{i-1,i}(k) - q_{i,i+1}(k))
\end{aligned}$$

Here the flow from one cell to another is given by:

$$q_{i,i+1} = \min(d_i, s_{i+1}) \tag{17}$$

d_i: demand of cell i, the number of vehicles that want to leave cell i and enter cell $i+1$.
s_{i+1}: supply of cell $i+1$, space available in cell $i+1$ for incoming vehicles from cell i.

So the flow from vehicles from cell i to cell $i+1$ is bounded by the supply of cell $i+1$ and the demand of cell i, with:

$$d_i = \begin{cases} q_i & \rho_i < \rho_{crit,i} \\ q_{max,i} & \rho_i \geq \rho_{crit,i} \end{cases} \quad s_i = \begin{cases} q_{max,i} & \rho_i < \rho_{crit,i} \\ q_i & \rho_i \geq \rho_{crit,i} \quad (congestion) \end{cases}$$

A cell can be in two stages: congested and non-congested. ρ_{crit} is the density at which capacity flow occurs. In congestion ($\rho_i \geq \rho_{crit,i}$) the demand of the cell is equal to the maximum flow q_{max} of the cell, the supply is then equal to the flow of the cell. If there is no congestion the demand is equal to the flow of the cell and the supply is equal to the capacity of the cell.

The velocity at every cell is calculated by the fundamental relation (see Background) that is based on the density:

$$v_i = v_i(\rho_i) = \begin{cases} v_{max,i} + \frac{v_{crit,i} - v_{max,i}}{\rho_{crit,i}} \rho_i & \rho_i < \rho_{crit,i} \\ v_{crit,i} \frac{\rho_{crit,i}}{\rho_{max,i} - \rho_{crit,i}} \left(\frac{\rho_{max,i}}{\rho_i} - 1 \right) & \rho_i \geq \rho_{crit,i} \quad (congestion) \end{cases} \tag{18}$$

Here v_{crit} is the speed at which the capacity flow occurs. v_{max} and ρ_{max} are respectively the maximum speed and the maximum density of a road. If the density and velocity of a cell are known the flow can be calculated by

$$q_i = \rho_i v_i \tag{19}$$

This equation follows directly from the definition of the three macroscopic traffic variables.

Nodes
Until now the calculations for the macroscopic variables are given for one cell to another. The equations are the same for 1-to-1 links, but at bifurcation and merge nodes the dynamics are a bit different. In both cases the density and speed are calculated in the original way. Some adjustments have to be made to the original equation for the flow in and out of a link (17).

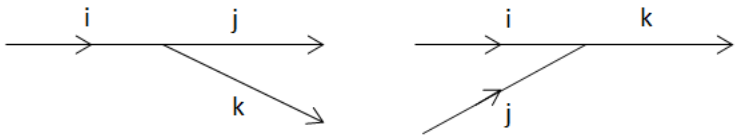


Figure 44: Bifurcation(left) and merge node(right).

Bifurcation node
At a bifurcation node, the demand of the incoming link should be divided over the outgoing links. In Figure 44 (left picture) a bifurcation point is shown, here the demand of link i is equal to the demand of link i to link j plus the demand of link i to link k, $d_i = d_{i,j} + d_{i,k}$. The demand to every link leaving the bifurcation node is determined by turn fractions γ.

$$q_{i,j} = \min(d_{i,j}, s_j) \text{ with } d_{i,j} = \gamma d_i$$
$$q_{i,k} = \min(d_{i,k}, s_k) \text{ with } d_{i,k} = (1-\gamma) d_i$$

If the supply of one of the outgoing links in not large enough, the remainder of the vehicles stay at the incoming link.

Merge node
Two or more links merge into one link at a merge point, for two links this is graphically shown in Figure 44 (right picture). The supply of the outgoing link should be divided over de incoming links. The supply of link k is distributed over the incoming links proportionally to the number of lanes L of the links.
$s'_{i,k} = \frac{L_i}{L_i+L_j} s_k$ and $s'_{j,k} = \frac{L_j}{L_i+L_j} s_k$
If the demand of link j is smaller than the supply $s'_{j,k}$, then there is extra room available for inflow from link i to link k and vice versa. Hence the supply of link k is divided over the links as follow:

$$s_{i,k} = s'_{i,k} + \max(0, s'_{j,k} - d_j)$$
$$s_{j,k} = s'_{j,k} + \max(0, s'_{i,k} - d_i)$$

Hence for merge nodes the flow is given by:

$$q_{i,k} = \min(d_i, s_{i,k})$$
$$q_{j,k} = \min(d_j, s_{j,k})$$

The inflow into the network at the boundary is based on historical data. Schuppen, Wang and Vranken made a model to predict the inflow at the boundary [19]. This prediction algorithm is based on the predictions of the week profile and the relative errors. This algorithm is already in use in the system of Trinité.

Multi-class model
Fastlane reproduces the differences of vehicle classes and the interactions between different vehicle classes. Vehicle classes are defined by their different characteristics, such as maximum speed, vehicle length, reaction times, minimum distance headways etc. The two most obvious vehicles classes are person cars and trucks. At low density and high speed, headways have a large impact on traffic flow. While at high density and low velocity vehicle length have large impact on traffic flow. These two observation are illustrated in Figure 45. This lead to the passenger car equivalent (pce) value, where the headway h and length of a vehicle l are taken into account.

$$\text{pce value: } \pi_u = \frac{l_u + h_u v_u}{l_{car} + h_{car} v_{car}} \tag{20}$$

where u refers to the vehicle class.

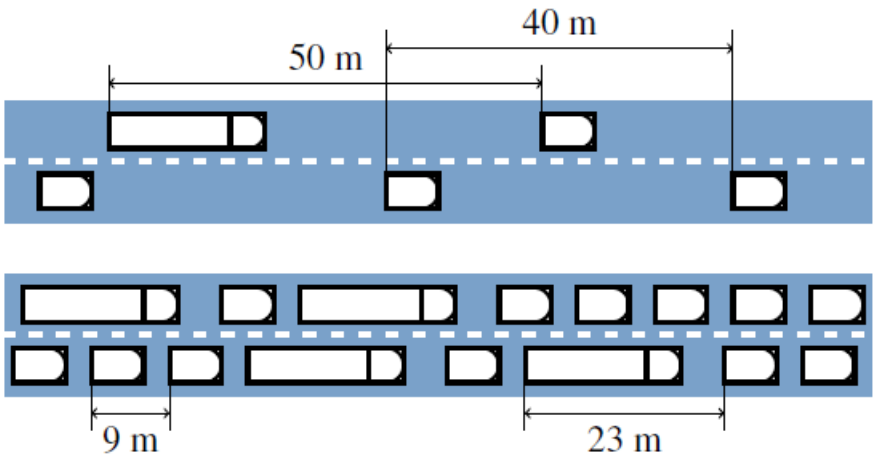


Figure 45: Top: no congestion, the headway of the different vehicle classes do not differ that much. Bottom: congestion, the length of a vehicle has significant impact on the headway of a vehicle. Source [13] .

The conservation of vehicles equation (16) and the equations we have derived for the velocity (18) and flow (19) also hold per vehicle class. With these equations of the macroscopic variables the traffic conditions can be simulated.

METANET

In METANET the motorway network is represented as a directed graph as described in [12] and [3]. Links represent homogeneous motorways and nodes the changes in road geometry. In this model a link m is also divided into N_m segments of equal length L_m. The length of the discrete time step of every iteration is denoted by T.

Different types of links
METANET uses five different types of links:

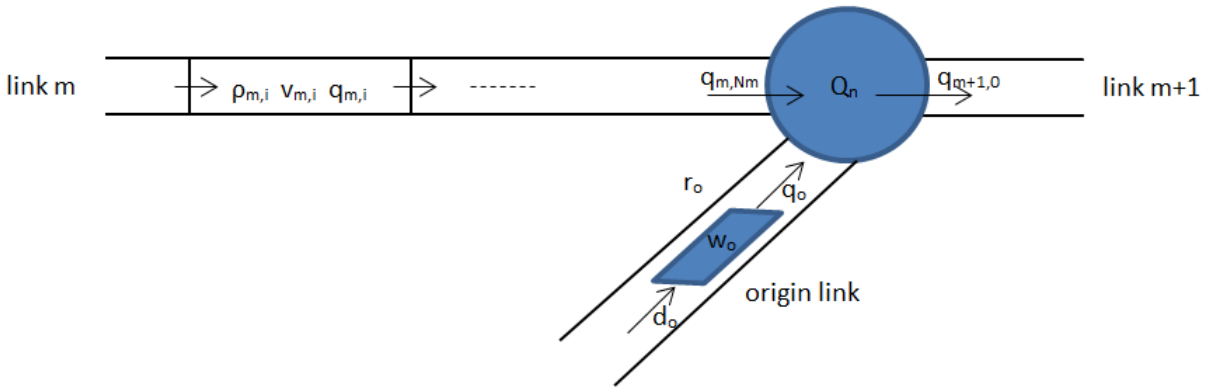


Figure 46: Link and node structure of METANET.

Motorway link: these are the general links used for homogeneous motorway stretches. The traffic conditions in these links are described by the basic macroscopic variables, with on segment i of link m density $\rho_{m,i}(k)$, mean speed $v_{m,i}(k)$ and flow $q_{m,i}(k)$. Every iteration k these basis macroscopic variables are defined for every segment i.

The density is based on the conservation of vehicles equation, so the equation for the density is approximately the same as we have seen in Fastlane:

$$\rho_{m,i}(k+1) = \rho_{m,i}(k) + \frac{T}{L_m l_m}(q_{m,i-1}(k) - q_{m,i}(k)) \tag{21}$$

(In this case the number of lanes l_m is taken into account, this has to do with the definition of the flow variable veh/h/lane or veh/h). The calculation of the flow is also the same as in Fastlane

$$q_{m,i}(k) = \rho_{m,i}(k) v_{m,i}(k) l_m \tag{22}$$

The mean speed in a segment is calculated by the empirical speed equation [18]:

$$\begin{aligned} v_{m,i}(k+1) = v_{m,i}(k) + \frac{T}{\tau}(V(\rho_{m,i}(k)) - v_{m,i}(k)) + \frac{T}{L_m} v_{m,i}(k)(v_{m,i-1}(k) - v_{m,i}(k)) \\ - \frac{\nu T}{\tau L_m} \frac{\rho_{m,i+1}(k) - \rho_{m,i}(k)}{\rho_{m,i}(k) + \kappa} \end{aligned} \tag{23}$$

$$\text{with } V(\rho_{m,i}(k)) = v_{free,m} \exp\left(-\frac{1}{a_m}\left(\frac{\rho_{m,i}(k)}{\rho_{crit,m}}\right)^{a_m}\right)$$

This speed update equation contains the average speed $v_{m,i}(k)$ in the current iteration k, the second term $\frac{T}{\tau}(V(\rho_{m,i}(k)) - v_{m,i}(k))$ expresses the behavior that the driver wants to achieve a desired speed $V(\rho)$. So if the desired speed is higher than the current average speed the speed will increase. The third term $\frac{T}{L_m} v_{m,i}(k)(v_{m,i-1}(k) - v_{m,i}(k))$ gives the difference between de speed of incoming and outgoing vehicles, so the increase (or decrease) of speed caused by the inflow of vehicles. The last term $\frac{\nu T}{\tau L_m} \frac{\rho_{m,i+1}(k) - \rho_{m,i}(k)}{\rho_{m,i}(k) + \kappa}$ expresses the speed increase (decrease) that drivers experience due to an lower (higher) density in the following segment than in the current segment. So if the density in the following segment $\rho_{m,i+1}(k)$ is higher than in the current segment $\rho_{m,i}(k)$ the speed will decrease.

The constant model parameters τ, ν and κ are the same for all links in the model. $V(\rho)$ represents the fundamental relation between the speed and density (note that this formula is different than the one used in Fastlane (18)).
In the speed equation (23) additional terms can be added to model merging near on-ramps or lane drops.

Next to these normal variables METANET also introduces destination-orientated variables:
Partial density, $\rho_{m,i,j}(k)$: the density of vehicles in segment i of link m with destination link $j \in J_m$.
Composition rate, $\gamma_{m,i,j}(k) \in [0,1]$: the portion of traffic volume of $q_{m,i}(k)$ with destination $j \in J_m$.
Where J_m is the set of destinations reachable via link m. The partial density is calculated as follows:

$$\rho_{m,i,j}(k+1) = \rho_{m,i,j}(k) + \frac{T}{L_m \lambda_m}(\gamma_{m,i-1,j}(k) q_{m,i-1}(k) - \gamma_{m,i,j}(k) q_{m,i}(k))$$
$$\text{with } \gamma_{m,i,j}(k) = \frac{\rho_{m,i,j}(k)}{\rho_{m,i}(k)}$$

Origin link: origin links receive traffic demand from outside the network and forward that into the network. Origin links are characterized by their outflow and queue length. The length of the queue, in number of vehicles, depends on the demand d_o and the outflow q_o at an origin. As illustrated in Figure 46 the queue length at origin o is given by:

$$w_o(k+1) = w_o(k) + T(d_o(k) - q_o(k))$$

An on-ramp is sometimes provided with ramp-metering. $r_o(k) \in [r_{min}, 1]$ is the metering rate for origin link o. If $r_o(k) = 1$ there is no ramp-metering, if $r_o(k) < 1$ then the ramp-metering is active. The outflow is given by:

$$q_o(k) = r_o(k) \min\{d_o(k) + \frac{w_o(k)}{T}, q_{max,o}(k) p(k)\}$$
$$\text{with } p(k) = \min\{1, \frac{\rho_{max} - \rho_{\mu,1}(k)}{\rho_{max} - \rho_{crit,\mu}}\}$$

Here $q_{max,o}$ is the free-flow on-ramp capacity and $p(k)$ is the portion of the flow capacity that can enter link μ. If $\rho_{\mu,1} < \rho_{crit,\mu}$ then $p(x) = 1$ otherwise $p(x) < 1$. The outflow is thus the minimum between the capacity of link μ and the flow of the queue plus the demand flow arriving at the queue at the time interval $[kT, (k+1)T)$.

The partial queue length is the number of vehicles in the queue of origin o with destination j:

$$w_{o,j}(k+1) = w_o(k) + T(\vartheta_{o,j}(k) d_o(k) - \gamma_{o,j}(k) q_o(k))$$

here $\vartheta_{o,j}(k)$ is the portion of arrivals at origin o at period k having link j as destination and $\gamma_{o,j}(k) = w_{o,j}(k)/w_o(k)$ the portion of vehicles in the queue at period k with destination link j.

The demand at an origin link is based on historical data and current data.

Destination link: destination links receive traffic flow from inside the network and forward it outside the network. The traffic flow in the destination link is influenced by the condition at

the boundary of the network. If there is no information available of the traffic condition at the boundary, it is assumed to be uncongested. The outflow of the destination link μ is limited by the maximal outflow $Q_{max,\mu}(k)$(boundary condition). Hence

$$q_{\mu,N_\mu}(k) = \min(q^b_{\mu,N_\mu}(k), Q_{max,\mu}(k))$$

where q^b is the outflow obtained by the basic equation (22) and N_μ is the last segment of link μ. When the outflow is limited by congestion outside the network the speed has to be recalculated:

$$v_{\mu,N_\mu}(k) = \begin{cases} v^b_{\mu,N_\mu} & \text{if } q^b_{\mu,N_\mu}(k) < Q_{max,\mu}(k) \\ v^b_{\mu,N_\mu} \frac{Q_{max,\mu}(k)}{q^b_{\mu,N_\mu}(k)} & \text{if } q^b_{\mu,N_\mu}(k) \geq Q_{max,\mu}(k) \end{cases}$$

Store-and-Forward link: to enable the model to approximately consider urban zones the store-and-forward links are used. These links are characterized by their flow capacity, queue length and constant travel time. The determination of the outflow and the queue length of a store-and-forward link are similar to the equation given above for the origin links.

Dummy link: dummy links have zero length and do not effect traffic but are used to decompose complex networks, they can be seen as help links.

Nodes

The nodes contain static information about the in and outflow of a link. O_n is the set of links entering node n and D_n is the set of links leaving node n. The total traffic flow of all entering links $i \in O_n$ of node n at period k is given by $Q_n(k)$, which by every iteration is evolved by $Q_n(k) = \sum_{\mu \in O_n} q_{\mu,N_\mu}(k)$.

For the outflow of the node and the corresponding inflow in an outgoing link we need turn fractions $\beta^m_n(k)$, the portion of traffic volume $Q_n(k)$ that leaves node n through link m. So the traffic flow that leaves node n through link m is given by $q_{m,0}(k) = \beta^m_n(k) Q_n(k)$ for all $m \in D_n$

We have to take into account the influence of the traffic situation of the incoming links on the outgoing links and vice versa. At a bifurcation node, the densities of the outgoing links have to be taken into account in the last segment of the incoming link:

$$\rho_{m,N_{m+1}}(k) = \frac{\sum_{\mu \in D_n} \rho^2_{\mu,1}(k)}{\sum_{\mu \in D_n} \rho_{\mu,1}(k)}$$

$\rho_{m,N_{m+1}}(k)$ can be used in equation (23) for $i = N_m$. The quadratic term is used, because one congested outgoing link may block the entering link even if the other leaving links are uncongested. At a merge point, the speed of the incoming links has to be taken into account according to (8). The mean speed value of incoming links are given by:

$$v_{m,0} = \frac{\sum_{\mu \in O_n} v_{\mu,N_\mu}(k) q_{\mu,N_\mu}(k)}{\sum_{\mu \in O_n} q_{\mu,N_\mu}(k)}$$

Urban network models

In this paragraph two models to predict travel times on urban road networks are presented. First we discuss a detailed urban network model, the Kashani model. Secondly we present a simple model based on data-driven methods to predict the travel time.

Kashani model

In 1983 Kashani developed an urban traffic model which is further developed by van den Berg [9] and Lin and Xi. In this paragraph the model of van den Berg is presented.

The focus will be on the traffic flow at a controlled intersection, because this is the most complex situation in an urban traffic network. The equations for uncontrolled intersection or any other flow restrictions can be derived in a similar way.

When describing an urban traffic network, the queues at an intersection are important. In this model every destination at an intersection has his own lane, so cars arriving at an intersection can move into the correct lane without blocking the traffic to other destinations. Hence every destination lane will contain its own queue. In practice this is not always the case. If there is only one origin lane for multiple destinations at an intersection, cars go to their destination as long as their is space in all destination links. As soon as one destination link is full the entire flow stops at the origin lane.

To model the traffic flows and queue length we need the following variables:

T: the simulation time step.
O_i: set of origins (incoming links) at intersection i.
D_i: set of destination (outgoing links) at intersection i.
$x_{o,i,d}(k)$: the queue length at intersection i in the lane for traffic going from origin o to destination d at time step k.
$\gamma_{o,i,d}(k)$: the relative fraction of the traffic arriving from origin o at intersection i that are going to destination d.
$n^a_{o,i}$: the number of cars arriving at the queue in the link connecting o and i during the time interval $[kT, (k+1)T)$.
$n^a_{o,i,d}$: the number of cars arriving at the queue in the lane to destination d in link (o, i).
$n^d_{o,i,d}$: the number of cars departing from the lane to destination d in link (o, i).
$s_{o,d}$: the available storage space of link (o, d), expressed in the number of vehicles.

These variables are graphically shown in Figure 47. The traffic leaving link (α, β) to destination d_i in time-period $[kT, (k+1)T)$ is determined by:

- The number of cars waiting and arriving at the intersection.
- The available space in the destination link.
- The capacity of the intersection, i.e., the number of cars that can pass an intersection in saturated condition of link (α, β) per unit of time, $Q^{sat}_{\alpha,\beta,d_i}$.

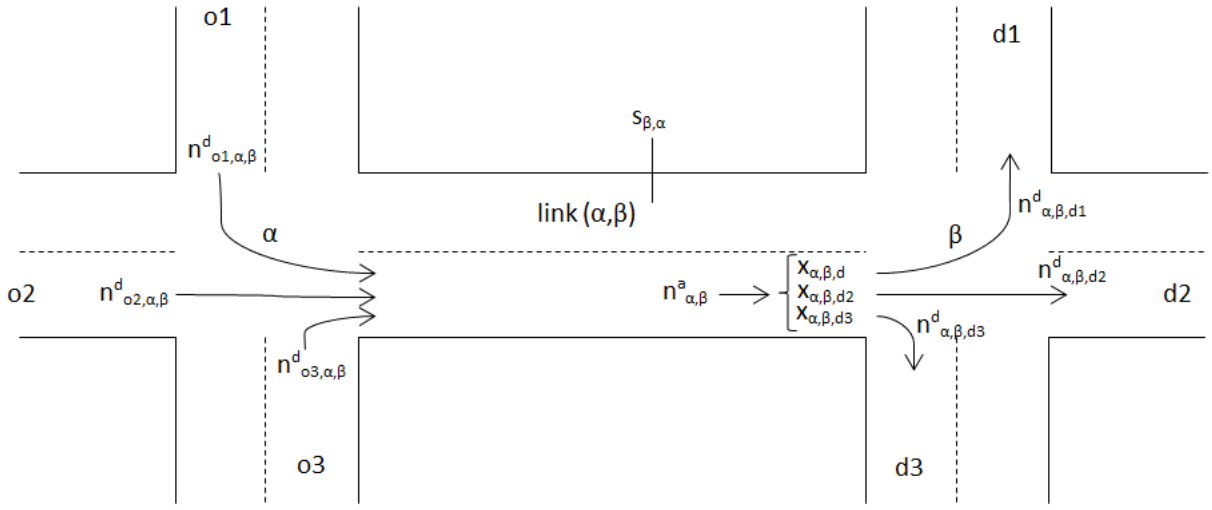


Figure 47: Graphical representation of the variables of the Kashani model on two successive intersections.

Hence for link (α, β), the number of cars leaving link (α, β) for destination d_i in time-period $[kT, (k+1)T)$ is given by:

$$n^d_{\alpha,\beta,d_i}(k) = \begin{cases} 0 & \text{if } g_{\alpha,\beta,d_i} = 0 \text{ (red light)} \\ \max(0, \min\{x_{\alpha,\beta,d_i}(k) + n^a_{\alpha,\beta,d_i}(k), s_{\beta,d_i}(k), TQ^{sat}_{\alpha,\beta,d_i}\}) & \text{if } g_{\alpha,\beta,d_i} = 1 \text{ (green light)} \end{cases} \tag{24}$$

The time for vehicles to run from the beginning of a link to the tail of the queue is given by $(\delta_{\alpha,\beta}(k) + \varphi_{\alpha,\beta}(k))T$, with

$$\delta_{\alpha,\beta}(k) = fix\left(\frac{s_{\alpha,\beta}(k)L_{veh}}{v_{\alpha,\beta}l_{\alpha,\beta}T}\right)$$
$$\varphi_{\alpha,\beta}(k) = rem\left(\frac{s_{\alpha,\beta}(k)L_{veh}}{v_{\alpha,\beta}l_{\alpha,\beta}T}\right)$$

Where L_{veh} is the average vehicle length, $l_{\alpha,\beta}$ the number of lanes in link (α, β) and $v_{\alpha,\beta}$ the free-flow speed on link (α, β). $Fix(x)$ gives the largest integer smaller than x and $rem(x)$ is the remainder. The vehicles that enter the link in iteration k will take $\delta_{\alpha,\beta}(k) + \varphi_{\alpha,\beta}(k)$ iterations to arrive at the tail of the queue. If a vehicle arrives at the tail of a queue in iteration k it should have entered link (α, β) $\delta_{\alpha,\beta}(k)$ or $\delta_{\alpha,\beta}(k) + 1$ steps ago. The fraction $\frac{T-\varphi_{\alpha,\beta}(k)}{T} = 1 - \varphi_{\alpha,\beta}(k)$ of the arriving vehicles will enter link (α, β) in time step $k - \delta_{\alpha,\beta}(k)$, hence
$n^a_{\alpha,\beta} = (1 - \varphi_{\alpha,\beta}(k)) \sum_{o_i \in O_\alpha} n^d_{\alpha,\beta}(k - \delta_{\alpha,\beta}(k)) + \varphi_{\alpha,\beta}(k) \sum_{o_i \in O_\alpha} n^d_{\alpha,\beta}(k - \delta_{\alpha,\beta}(k) - 1)$

The number of vehicles arriving at the queue to destination d_i in link (α, β) is calculated from $n^a_{\alpha,\beta}$ and the corresponding turn-fraction:

$$n^a_{\alpha,\beta,d_i}(k) = \gamma_{\alpha,\beta,d_i} n^a_{\alpha,\beta}(k)$$

The new queue lengths are derived by the old queue lengths plus the arriving vehicles minus the departing vehicles in iteration k.

$$x_{\alpha,\beta,d_i}(k+1) = x_{\alpha,\beta,d_i}(k) + n^a_{\alpha,\beta,d_i}(k) - n^d_{\alpha,\beta,d_i}(k)$$

The new storage depends in a similar way on the number of cars leaving and entering a link:

$$s_{\alpha,\beta}(k+1) = s_{\alpha,\beta}(k) - \sum_{o_i \in O_\alpha} n^d_{o_i,\alpha,\beta}(k) + \sum_{d_i \in D_\alpha} n^d_{\alpha,\beta,d_i}(k)$$

Instead of working with the number of vehicles n leaving or entering links, we could also use (average) flows q leaving or entering links. So the number of cars leaving link (α, β) for destination d_i, $n^d_{\alpha,\beta,d_i}(k)$, will be given by the flow leaving link (α, β) for destination d_i, $q^d_{\alpha,\beta,d_i}(k)$. The equation for q^d_{α,β,d_i} is derived by dividing the terms in equation (9) by the simulation time step, i.e., the number of vehicles are divided by the time step, so we derive at the flow:

$$q^d_{\alpha,\beta,d_i} = \begin{cases} 0 & \text{if } g_{\alpha,\beta,d_i} = 0 \text{ (red light)} \\ \max(0, \min\{x_{\alpha,\beta,d_i}(k)/T + q^a_{\alpha,\beta,d_i}(k), s_{\beta,d_i}(k)/T, Q^{sat}_{\alpha,\beta,d_i}\}) & \text{if } g_{\alpha,\beta,d_i} = 1 \text{ (green light)} \end{cases}$$

The definition of the queue length stays the same, the only difference is that the departing and arriving flow are used to calculate the new queue length:

$$x_{\alpha,\beta,d_i}(k+1) = x_{\alpha,\beta,d_i}(k) + (q^a_{\alpha,\beta,d_i}(k) - q^d_{\alpha,\beta,d_i}(k))T$$

The other equations are derived in a similar way.

Simple model

Urban networks are very complex to model, due to the queues and traffic lights at intersections. This can be seen in the number of variables of the Kashani model. Therefore we have developed a simple data-driven model.

The route on a network consist of a combination of two parts:

- Travel time on a link
- Waiting time at an intersection

The travel time at a link can be calculated straight forward by the length of link L divided by the average speed on that link, $L/v(t)$. If there is sensor data available, the time-dependent average speed based on historical data is calculated for every link. In the calculation of travel time on a link we take both the historical average speed and the current average speed into account. The average speed $v(t)$ will be a combination of the current and historical data: $(1 - K) \cdot$ (historical average) + $K \cdot$ (real time average), with K the value that gives the best estimates. If there are no sensor measurements available on a link, the free-flow speed is taken, see Background for more information.

The waiting time $w(t)$ at an intersection is modeled as the average waiting time at the queue at the intersection. If there is data available of the average waiting time, the waiting time at an intersection will be calculated as a combination of the current and historical data as described above. If there is no data available over the average waiting time, a fixed value $c(t)$ for the waiting time is chosen. These values $c(t)$ are time depended and the correct values have to be set in further research.

Discussion

Freeway models

Both Fastlane and METANET divide the network into links, which are defined in the same way as in the approach of Trinité. Furthermore, in both models the links are divided into segments and all segments in a link have the same length. METANET uses a graph structure with links and nodes, while Fastlane uses only links and transition flows. In both models the route choice of vehicles is based on turn fractions.

For the flow inside the segment both models use the same straight forward formula: $q = \rho v$. In both models the density is based on the conservation of vehicles equation. In contrary to METANET, Fastlane uses the transition flows between segments in this equation, where the demand and supply of segments are taken into account. Therefore Fastlane's method is preferable for calculating the density.

For the speed in the segments the models use very different equations. Fastlane uses a formula to describe the fundamental relation between the density and speed. METANET calculates the speed by the empirical speed equation, which contains many terms to model the different influence factors. The many model parameters which have to be set is a downside of METANET's formula. However the equation of METANET seems to be more accurate, since many features of the speed in a segment are taken into account. The fundamental relation is only one of the inputs of the equation of METANET.

The difference in characteristics of different vehicle classes can be easily taken into account by Fastlane. Fastlane is also a simpler model than METANET and is already used by Trinité. In METANET the origin-destination variables are already defined. Furthermore METANET has different types of links, so it can better describe the traffic situation of on- and of-ramps. The main difference is that METANET is more complex than Fastlane, therefore it is also more suitable for complex networks.

Based on the pros and cons for both models I think that Trinité should develop a highway model that is a combination of both models. The model has to contain the best of the two models. As a start Fastlane can be used, since it is already implemented and developing a new model will take time. Some features of METANET like the origin and destination links and the speed equation can be good additions to the model.

Urban networks

For urban networks there are not so many reliable models available. Kashani is by far the best model we have found in our research. A drawback of this model are the many variables and that it is difficult and time consuming to implement at once. Therefore we would like to start with the Simple model, which is simple and fast to implement. This model is not a simulation model but a data-driven method, so the results will not be very accurate. However it is a good model to start with and to further build on, with as goal to develop a model similar to the Kashani's model.

Another possibility, instead of the Simple model, is to use the freeway model also for the urban road networks. Despite the difference in characteristics of urban and freeway networks, this is also a good way to start.

In generally the focus lies on modeling the main roads, like the A-, N- and S-roads. For travel time calculations also the travel times on the small roads are needed. The traffic situations on these small road will not be simulated, because they are less important and there is few sensor data available. Therefore the travel time on a small urban road link will be calculated by dividing the length of the link by the free-flow speed.

Symbols

Symbol	Definition
S	Number of traffic lights.
λ	Inflow at the queue in vehicles per minute.
μ	Outflow of the queue in vehicles per minute.
ρ	The occupation rate.
N	Maximum number of vehicles in the queue, which is never reached.
ΔT	Length of the time interval in which the fixed arrival rate changes.
Δt	Length of the time interval in which the distribution is calculated.
π_t	Exact distribution at time t.
$\pi_t(n)$	Probability that the process is in state n at time t.
$\widehat{\pi}_t$	Expected distribution at time t.
$\widehat{\pi}_t(n)$	Expected probability that the process is in state n at time t.
φ_i	Exact distribution at transition i.
$\varphi_i(n)$	Probability that the process is in state n at transition i.
$\widehat{\varphi}_i$	Expected distribution at transition i.
$\widehat{\varphi}_i(n)$	Expected probability that the process is in state n at transition i.
δ	Rate that the process stays in the current state.
C	Total rate out of a state, i.e., $\lambda + \mu + \delta$.
$n_\alpha(t)$	$1-\alpha$ upper confidence limit of the number of vehicles in the queue at time t.
D	Deadline to be out of the queue.
T_D^a	Latest arrival time such that the app user is with confidence level $1-\alpha$ on time out of the queue, before deadline D.
Δx	Length of the time interval in which an app user can be scheduled.
ξ_i	Fixed arrival flow in interval i in vehicles per minute, the rate of inflow of the vehicles that do not use the app.
ξ	Arrival flow of non app users per interval, i.e., $\xi = (\xi_1, \ldots, \xi_n)$.
η_i	Number of app users that are scheduled in interval i.
η	Schedule of app users per interval, i.e., $\eta = (\eta_1, \ldots, \eta_n)$.
λ_i	Total arrival flow in interval i in vehicles per minute, i.e., $\lambda_i = \xi_i + \eta_i/\Delta x + \gamma_i/\Delta x$.
λ	Total arrival flow per interval in vehicles per minute, i.e., $\lambda = (\lambda_1, \ldots, \lambda_n)$.
ΔL	Length of the time interval in which an app users can be scheduled.
l	An app user can be scheduled in l intervals, i.e., $\Delta L = l\Delta x$.
M_D	Number of app users with deadline D.
A_D	Latest arrival interval corresponding to the latest arrival time T_D^a, i.e., all users with deadline D that are scheduled before or at this interval will be with a pre-set confidence level on time out of the queue.
$L_i(x, \lambda)$	Mean queue length at time $i\Delta x$ when starting in initial state x with arrival rates λ.
N_i	Number of app users with latest arrival interval i.
σ	Fraction of citizens that use the app.
Δk	Time between two calculation times of the algorithm.
γ_i	Number of fixed scheduled users in interval i, users which already got a departure sign.
γ	Schedule of fixed scheduled app users per interval, i.e., $\gamma = (\gamma_1, \ldots, \gamma_n)$.

References

[1] Adan, I. Resing. J. *Queueing Theory*, February 28, 2002

[2] Been, de B. *Afrijcapaciteiten, aanzet tot herziening van bestaande richtlijnen*, Tilburg, DTV Consultants 1994

[3] Berg, van den M. Hegyi, A. de Schutter, B. Hellendoorn, J. *A macroscopic traffic flow model for integrated control of freeway and urban traffic networks*, Proceedings of the 42nd IEEE Conference on Decision and Control, Hawaii, pp. 2774-2779, 2003

[4] Bhulai, S. Koole, G. *Stochastic Optimization*, 2012

[5] Conolly, B.W. Langaris, C. *On a New Formula fo the Transient State Probabilities for M/M/1 Queues and Computational Implications*, Journal of Applied Probability, Vol. 30, No 1, pp. 237-246, March 1993

[6] Data IJburg webstats Amsterdam

[7] Englewood Cliffs, N.J., *Transportation and Traffic Engineering Handbook*, 2nd ed., Prentice-Hall, 1982

[8] Leeuwaarden, van J.S.H. *Delay analysis for the fixed-cycle traffic light queue*, 2000

[9] Lin, S. Schutter, de B. Xi, Y. Hellendoorn, J. *A simplified macropscopic urban traffic network model for model-based predictive control*, Proceedings of the 12th IFAC Symposium on Transportation Systems, Redondo Beach, california, pp. 286-291, 2009

[10] Lint, H. *Hoe meer realtime-adviezen hoe slechter de kwaliteit*, Verkeerskunde 2008

[11] Lint, van H. *Reader Innovation in dynamic traffic management, Integrated and Coordinated Networkmanagement, 2012*

[12] Kotsialos, A. Papageorgiou, M. Diakaki, C. Pavlis, Y. Middelman, F. *Traffic flow modeling of large-scale motorway networks using the macroscopic modeling tool METANET, IEEE transactions on intelligent transportation systems*. pp. 282-292, 2002

[13] Schreiter, T. Wageningen-Kessels, van F.L.M. Yuan, Y. Lint, van J.W.C. Hoogendoorn, S.P. and TRAIL Research School, *Fastlane, Traffic flow modeling and multi-class dynamic traffic management*, 2012

[14] Sharma, P.K. Gupta, D.V. Mohit, *Computational Solutions of Transient M/M/1/∞ Queue*, VSRD Technical & Non-Technical JOURNAL, Vol 2 2011, pp. 160-168

[15] Sharma, O.P. *Markovian Queues*, Ellis Horwood, Chichester, 1990

[16] Wachs, M. Samuals, J.M. Skinner, R.E. *Highway Capital Manual* Transportation Research Board, National Research Council, National Academy of Science, Chapter 8 Traffic Characteristics, 2000

[17] Wachs, M. Samuals, J.M. Skinner, R.E. *Highway Capital Manual* Transportation Research Board, National Research Council, National Academy of Science, Chapter 16 Signalized Intersection, 2000

[18] Wageningen-Kessels,van F. *Techical documentation Fastlane*, 2008

[19] Wang, Y. Vrancken, J. Ottenhof, F. Valé, M. *Next generation traffic control in the Netherlands.*

[20] Wilson, A. *Handboek verkeerslichtenregelingen*, january 2006