­­­­Simulation for the Uithoflijn

**Problem description**

The operational performance of the soon-to-be-constructed Uithoflijn, connecting the center of Utrecht with the Utrecht University campus, is analysed in this report.

The line consists of nine stops: *Centraal Station (CS), Vaartsche Rijn, Galgenwaard, Kromme Rijn, Padualaan, Heidelberglaan, UMC, WKZ, P+R De Uithof*. Trams run in both directions (i.e. CS -> P+R and P+R -> CS respectively).

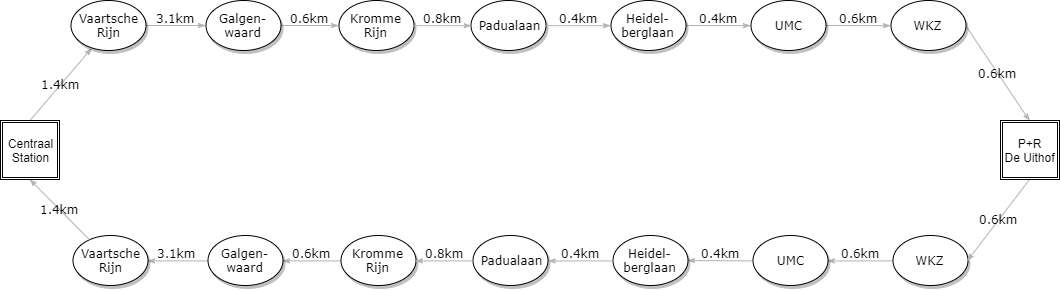


Figure 1: Transportation Map

*CS* and *P+R* are the line’s endstations, where there is a daily timetable spanning from 06:00 to 21:30. The schedule’s frequency is every 15 minutes for the first hour (i.e. 06:00-07:00) and after 19:00. During the peak hours (i.e. 07:00-19:00) there is a higher frequency of every *f* minutes. Assuming an end-to-end driving time of 17 minutes and a turnaround time of *q minutes*, these frequencies determine the scheduled departures at each endstation as follows:

* .
* .

For safety reasons, there is a 40-second time interval trams have to respect, so a tram cannot depart from a station before 40 seconds from the time of the last departure.

Moreover, trams can enter or withdraw from the line only at *P+R*. At each endstation, there are switches for changing directions, which block for *switch\_delay* seconds after traversing them. All trams have a fixed capacity of 420passengers.

In the case of a tram arriving at a station before the next scheduled departure, the tram has to wait and utilize this time to (dis)embark passengers.

**Research Questions**

1. Which are the feasible frequencies of the tram?
2. What is the maximum amount of passengers the line can handle?
3. What measures do you recommend to improve the operational performance?

As these questions are difficult to answer by analysis alone, we have developed a simulation of the Uithoflijn, based on techniques of *discrete-event* *simulation*.

The implementation is based on an abstract model of the actual system (see *Simulation Model*), which was designed with the help of given data on a corresponding bus route that operates in a similar area and prognostic data from the transportation company (see *Input Analysis*).

Another source of information was a group interview with domain expert Marcel van Kooten Niekerk from the Qbuzz company, which gave us insight on designing our model (see *Appendix: Interview Minutes*).

The resulting system enables us to simulate the tram line in a highly parametric way and gives us the ability for further analysis (see *Output analysis*). In order to validate our model, we run our simulation against simpler artificial data (see *Validation*).

**Assumptions**

In order to manage the complexity of the system and make it possible to provide a bug-free simulation system, we have made the following assumptions:

* There are no unexpected failures to any part of the tram line (line tracks, tram engines, etc…).
* The change of the tram drivers is executed immediately without breaks, thus we can totally ignore any delay relating to that.
* The simulation is not affected by the weather conditions.
* All passengers disembark on arrival at an endstation.
* Every passenger wants to embark on the first tram that he encounters.
* All trams have the same capacity.
* The 40-second safety measure at each station ensures with 100% guarantee that there will be no tram collision between two stops.
* The arrival of the passengers on the stops of the tram line can be adequately modeled as a Poisson process of a varying rate λ, which changes every 15 minutes.

**Quantitative Analysis**

Some portion of the questions posed can be answered from quantitative analysis alone, without the need to resort to a full-blown simulation of the tram line.

1. **Which are the feasible frequencies of the tram?**Assuming the fact that there are exactly 13 trams available, a specific turnaround time of minutes and a end-to-end driving time of 17 minutes, we can easily find out the absolute maximum (integral) frequency that is feasible.

Additionally, these 13 trams will be divided in half, to serve the initial departures from each endstation. To get the maximum frequence possible, we assume trams will serve *P+R* and trams will serve *Centraal Station*. Let’s name each tram .  
  
To do that, let’s consider a timetable starting at the first minute of an hour at *P+R,* operated by tram. Consequently, the first departure at *Centraal Station* will be minutes later, where

and will be operated by tram . The same tram will arrive at *P+R* minutes later, where

Therefore, we have to satisfy all scheduled departures at *P+R* in the first minutes, using only trams to . In other words, we can only satisfy 6 scheduled departure in the first minutes. Hence, the highest (integral) frequency is given by the formula below:

As an example, given minutes, we derive this highest frequency .  
  
Of course lower frequencies are always feasible, but as our simulation results will later suggest, this will definitely affect the operational performance of the Uithoflijn. For example, although having trams depart every 100 minutes from each endstation, is a feasible frequency, it will have dramatic effects on passengers’ waiting times, but will definitely reduce departure delays.

1. **What is the maximum amount of passengers the line can handle?**Assume 18 stops, at each of which passengers arrive as a Poisson process with rate for , tram capacity of passengers and a timetable frequency .

We can now reformulate the question of finding the maximum amount of passengers, to finding the maximum rates . If we also assume that there are no departure delays, we can derive the following formula for calculating the maximum rates:

In conclusion, the maximum rates are those, who have a certain sum given by this equation:

**Simulation Model**

**Events**

As we conduct *discrete-event simulation*, we need to define a meaningful set of events that will drive the simulation progress; since they will capture the most significant changes in the system’s state.

The events we propose are the following:

* ***END\_SIM***The simulation has come to an end.
* ***LAMBDA\_CHANGE***The rate λ, at which passengers arrive at the tram stops, changes .  
  (i.e. at this time, the flow of passengers is increased or decreased)
* ***PASSENGER\_ARRIVAL***  
  A passenger arrives at a tram stop.
* ***TRAM\_EXPECTED\_ARRIVAL***A tram has reached a station, but may need to wait for other trams to depart first, resulting in subsequent trams queueing up on the tram line.
* ***TRAM\_ARRIVAL***A tram has progressed through the waiting queue of a stop and has arrived at the station’s platform, where passengers can start (dis)embarkation. At this point in time, there will be a certain delay, named *dwell\_time*.
* ***TRAM\_EXPECTED\_DEPARTURE***A tram is ready to depart for the next stop, but may need to wait before doing so, for safety reasons (40-second rule) or due to the endstation switches being blocked.
* ***TRAM\_DEPARTURE***A tram finally departs for the next stop, which will take a certain time, named *driving\_time*. If there are any trams enqueued on the tram line, the next one can proceed to the station’s platform.
* ***TRAM\_DESTROY***A signal for trams to start withdrawing at *P+R*, when switching from peak hours to off-peak hours, as less trams are required to continute operating until the end of the day.

**Initialization**

Initially, only a subset of the available trams will start operating, all other trams will be spawned at 07:00 (start of the peak hours). Then, at 19:00 (end of the peak hours) the majority of the trams will withdraw. To spawn a tram, we enqueue it at *P+R*. At the beginning of the simulation, we schedule all the changes of the λ rate (*LAMBDA\_RATE),* the initial trams will be divided into those that will begin at *P+R* and those that will go straight to Centraal Station (*TRAM\_EXPECTED\_ARRIVAL*), the rest of the trams will spawn one hour later (*TRAM\_EXPECTED\_ARRIVAL*), the first passengers at each stop (*PASSENGER\_ARRIVAL*), the withdrawal of the trams at the end of the peak hours (*TRAM\_DESTROY*) and the end of the simulation (*END\_SIM*).

**Event Graph**

Below is the *event graph* of our model, in which nodes represent types of events and edges represent scheduling capability (i.e. an arrow from A to B means events of type A can schedule events of type B). Dashed lines indicate that an event may *immediately* schedule another event, without any time delay.

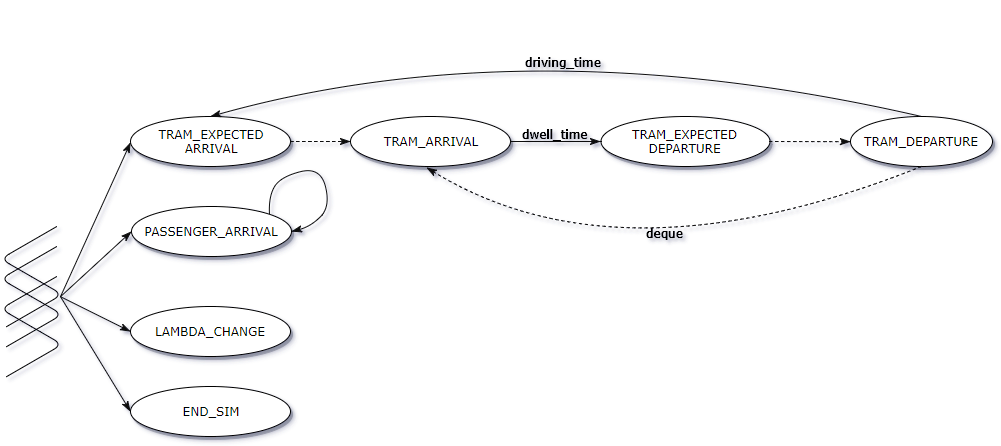


Figure 2: Event graph

**State**

Events will eventually change the overall state of the system. Therefore, the state of the simulation will have to capture any properties of the system that vary over time, in contrast to parameters of a particular execution, which are invariant as time progresses. The state consists of the following components:

* t : current time
* λ : current arrival rate of passengers (one for each stop)
* trams : a list of all the trams, each consisting of the following:
  + capacity : number of passengers onboard
  + [nonstop] : station to go straight to
* stops : a list of all the stops, each consisting of the following:
  + capacity : number of passengers waiting at the stop
  + last\_departure : time of the last train’s departure
  + arrivals : queue of the time each passenger has arrived
  + to\_destroy : number of trams to withdraw
  + [parked\_tram] : the tram currently (dis)embarking passengers

**Performance measures**

In order to evaluate and compare different configurations of the system, we have defined the following important system metrics:

1. *Punctuality*Overall panctuality of the tram operation, measured from the deviation of the actual departure times from the prearranged schedule (only at the endstations). Formally,

We care about the *average* and *maximum* departure delay, as well as the percentage of trams that experienced a departure delay of more than 1 minute. Formally,

1. *Passengers’ waiting time*Amount of time passengers have waited to board the tram, since their arrival at the stop. Formally,

We care about the *average* and *maximum* waiting times, as well as the percentage of passengers that waited more than minutes, where is the current timetable frequency. Formally,

1. *Stop congestion*Overall congestion of each stop, measured from the *average* number of passengers waiting at the same time. Formally,

**Event handlers**

The handling of each event is now given as pseudocode, in order to demonstrate the most significant changes events make to the current state.

*# END\_SIM*

**def** handle(state):

state.end\_simulation := True

*# LAMBDA\_CHANGE*

**def** handle(state):

state.λ := next λ

*# PASSENGER\_ARRIVAL*

**def** handle(state, passenger, stop):

**if** (tram = stop.parked\_tram):

tram.embark(self)

UPDATE

**else**:

stop.enter(passenger)

UPDATE

inter\_arrival := generate passenger\_arrival(stop)

SCHEDULE new PASSENGER\_ARRIVAL(now + inter\_arrival, stop)

*# TRAM\_DESTROY*

**def** handle(state, trams\_to\_withdraw):

state.stops["P+R"].to\_destroy := trams\_to\_withdraw

*# TRAM\_EXPRECTED\_ARRIVAL*

**def** handle(state, tram, stop):

**if** tram.nonstop:

**return**

**if** stop.parked\_tram == None:

SCHEDULE new TRAM\_ARRIVAL(now, tram, stop)

**else**:

**if** stop.to\_destroy > 0:

stop.to\_destroy--

delete tram

**else**:

stop.enqueue(tram)

*# TRAM\_ARRIVAL*

**def** handle(state, tram, stop):

stop.parked\_tram = tram

**if** tram.nonstop == stop:

tram.nonstop := None

**if** tram.nonstop:

SCHEDULE new TRAM\_EXPECTED\_DEPARTURE(now, tram, stop)

**else**:

p\_out\_percentage := GENERATE passenger\_exit\_percentage(self.stop)

p\_out := tram.capacity \* p\_out\_percentage

tram.disembark(p\_out)

p\_in := min(420 - tram.capacity, stop.capacity)

tram.embark(p\_in) -> update passenger\_waiting\_time

UPDATE

dwell\_time := GENERATE dwell\_time(p\_in, p\_out)

**if** stop == endstation:

next\_schedule := state.timetable[stop].next\_schedule()

seconds\_late := now - next\_schedule

UPDATE

**if** next\_schedule > now:

wait\_for\_schedule := next\_schedule - now

**else**:

wait\_for\_schedule := 0

delay := max(safety\_time, dwell\_time, wait\_for\_schedule)

SCHEDULE new TRAM\_EXPECTED\_DEPARTURE(now + delay, tram, stop)

*# TRAM\_EXPECTED\_DEPARTURE*

**def** handle(state, tram, source, target):

source.parked\_tram := None

dwell\_inter := GENERATE dwell\_time(intermediate\_passengers)

dwell\_switch := source.use\_switches()

safety\_time := now - (stop.last\_departure + 40sec)  
 delay := max(dwell\_inter + dwell\_switch, safety\_time)

SCHEDULE new TRAM\_DEPARTURE(now + delay, tram, source, target)

*# TRAM\_DEPARTURE*

**def** handle(state, tram, source, target):

source.last\_departure := now

dt := GENERATE driving\_time(source, target)

SCHEDULE new TRAM\_EXPECTED\_ARRIVAL(dt, tram, target)

**if** (next\_tram = source.queue.pop()):

SCHEDULE new TRAM\_ARRIVAL(now, next\_tram, source)

**Input Analysis**

**Stop correspondence**

The Uithoflijn consists of 9 stops with a total distance of 7.9 km. In order to inform our choice of random number generators for our simulation, we used datasets from the bus 12 route that follows more or less the same route as the new tram line. The bus 12 also consists of 9 stops with a total distance of 6.7 km**.**

In Table \_ we see the correspondence between the stops of the new tram line and the bus stops. Unfortunately, there is no one-to-one correspondence between the station of the future tram line and the bus stops. To remedy this, we have assumed that the bus stop AZU corresponds to P+R De Uithof station and we also fitted the two missing bus stops from the correspondence table, Rubenslaan and Sterrenwijk, so that every station of the tram has a counterpart bus stop.

|  |  |
| --- | --- |
| Stop | Corr. stop route 12 |
| Centraal Station | CS Centrumzijde |
| Vaartsche Rijn | Bleekstraat |
| Galgenwaard | Galgenwaard |
| Kromme Rijn | De Kromme Rijn |
| Padualaan | Padualaan |
| Heidelberglaan | Heidelberglaan |
| UMC | Rubenslaan |
| WKZ | Sterrenwijk |

**Table 1: Tram correspondence with the route of Bus 12route**

Beyond that, we used two datasets which contained measures of entering and leaving passengers within a month for each direction of the bus 12. Moreover, we have assumed that the passengers arrive at the stops according to a Poisson process under a rate which varies every 15 minutes. Another assumption was that the arrival changing rates are the same among all days. For instance, the rate of the day 2/9/15 from 6:00 to 6:15 is the same for all the days in the dataset and the same goes from 6:15 to 6:30 and so forth.

**Passenger arrivals**

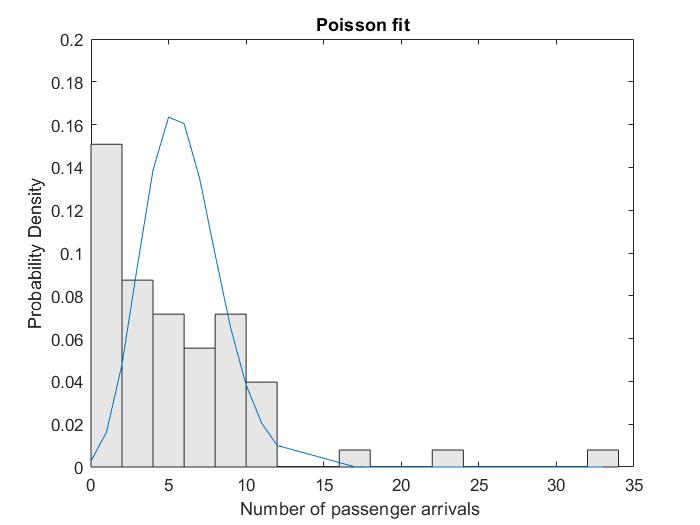
Under the aforementioned assumptions, we collected the entries from every station of each direction every 15 minutes and we used the Poisson distribution to fit our data and determine the rate λ of the passenger arrivals. In general, the Poisson distribution describes a random variable which represents the number of events (in our simulation the passenger arrivals) that occur in a time interval. So, this gave us the opportunity to exploit the convenient relationship between Poisson and Exponential distribution and use the latter to model inter-arrival times.

The following table shows the average λ-rates across all stops for some periods of the day, which are used by our random number generators to model passenger inter-arrival times as an exponential distribution of the same rate:

|  |  |
| --- | --- |
| Time Periods | λ rate |
| 06:00-06:15 | 1.107 |
| 06:15-06:30 | 2.911 |
| 06:30-06:45 | 2.816 |
| 06:45-19:00 | 3.66 |
| 19:00-19:15 | 3.086 |
| 19:15-19:30 | 4.550 |
| 19:30-19:45 | 5.615 |
| 19:45-20:00 | 6.353 |
| . | **.** |
| . | **.** |
| 19:45-20:00 | 3.709 |
| 20:00-20:15 | 3.580 |
| 20:15-20:45 | 3.441 |
| 20:30-20:45 | 2.957 |
| 20:45-21:00 | 2.304 |
| 21:00- 21:15 | 2.148 |
| 21:15-21:30 | 2.202 |

Table 2: Average Poisson rates λ

Below is a histogram of a random quarter of a peak hour, as fitted to a Pisson distribution by our Matlab scripts:



**Passenger exit percentage**

Regarding the exit percentage of the passengers at each stop, we transformed our dataset into percentages, as show in the table below:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CS Centrum | Bleekstraat | Sterrenwijk | Rubenslaan | Stadion Galgenwaard | De Kromme Rijn | Padualaan | Heidelberglaan | AZU |
| 0 | 1.5385 | 1.5625 | 9.375 | 18.966 | 6.383 | 15.909 | 40.541 | 100 |
| 0 | 4.7619 | 1.6393 | 1.6667 | 14.754 | 1.9231 | 9.8039 | 60.87 | 100 |
| 0 | 1.6129 | 0 | 6.3492 | 25 | 4.4444 | 11.628 | 78.947 | 100 |
| 0 | 4.3478 | 0 | 4.3478 | 15.909 | 5.4054 | 14.286 | 76.667 | 100 |
| 0 | 0 | 3.1746 | 9.2308 | 18.333 | 4.0816 | 23.404 | 75 | 100 |
| 0 | 4.8387 | 0 | 1.6949 | 8.6207 | 0 | 45.283 | 48.276 | 100 |
| 0 | 1.5873 | 0 | 11.29 | 14.286 | 0 | 41.667 | 57.143 | 100 |
| 0 | 1.4286 | 0 | 7.0423 | 8.8235 | 3.2258 | 40 | 61.111 | 100 |
| 0 | 0.84746 | 0.84746 | 1.6667 | 8.1967 | 2.6316 | 47.748 | 68.966 | 100 |

Table 3: Exit percentages of each stop

We then chose the beta distribution function to fit our data - one for each every individual station - in order to acquire the probability that someone will leave throughout this station within the day. Our motivation for using this specific distribution was that it is suitable model for the random behavior of percentages and proportions and, moreover, has scored better than other regular distributions in well-known fitness tests provided by the Matlab standard toolboxes. This resulted in the and parameters of each station’s beta distribution, as shown in the table below:

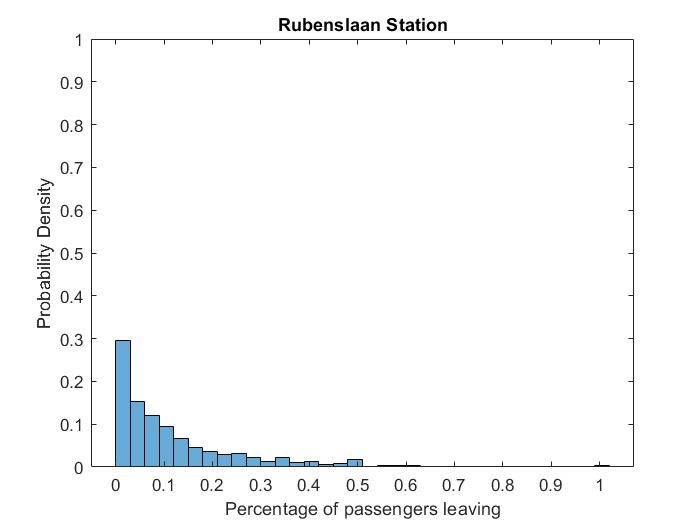
|  |  |
| --- | --- |
| Stops | (, ) |
| P+R | (0, 0) |
| WKZ | (0, .04) |
| UMC | (0, .1) |
| Heidelberglaan | (.01, .09) |
| Padualaan | (.003, .18) |
| Kromme Rijn | (.002, .17) |
| Galgenwaard | (.001, .27) |
| Vaartsche Rijn | (.007, .19) |
| Centraal Station | (0, 0) |

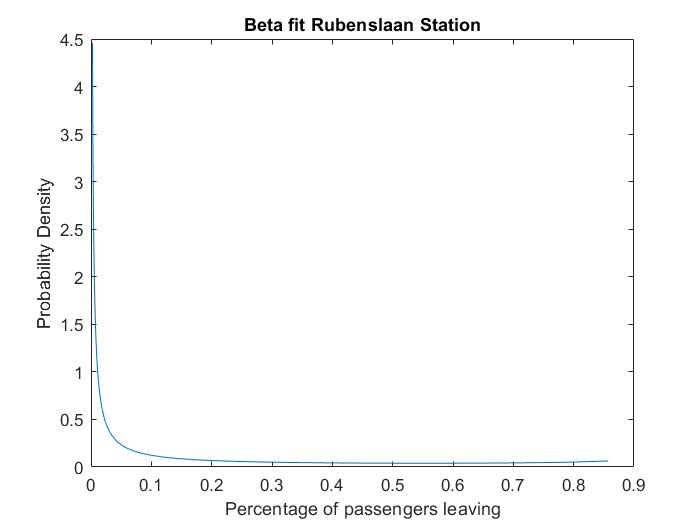
Table 4: Beta parameters of exit percentage (P+R -> CS)

|  |  |
| --- | --- |
| Stops | (, ) |
| Centraal Station | (0, 0) |
| Vaartsche Rijn | (0, 0) |
| Galgenwaard | (.006, .14) |
| Kromme Rijn | (.003, .17) |
| Padualaan | (.012, .2) |
| Heidelberglaan | (.006, .16) |
| UMC | (.004, .2) |
| WKZ | (.01, .11) |
| P+R | (.03, .04) |

Table 5: Beta parameters of exit percentage (CS -> P+R

Obviously, we use the parameters on our random number generators to produce numbers from these beta distributions. A sample of the input data and the fitting distribution is represented in the figures below:





However, we have to note that the dataset in some cases appears to be inconsistent. In particular, there are stations that more passengers appear to leave from the tram than those who are already inside, which we eschewed from by allowing boarding passengers to immediately get out.

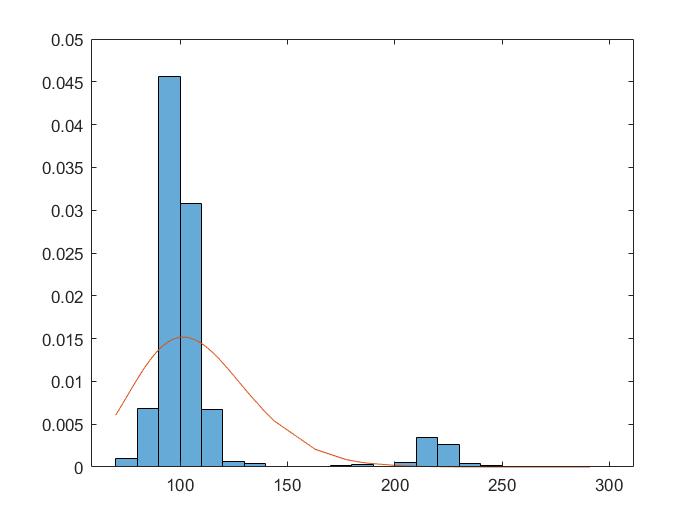
**Runtimes**

To model the tram runtimes we used a large set of measurements from the Nieuwegein-tramline which consists of 14 stops, shown in this table:

|  |
| --- |
| Bus 12 route |
| Graadt van Roggenweg |
| 24 Oktoberplein |
| 5 Meiplein |
| Vasco da Gamalaan |
| Kanaleneiland-Zuid |
| P+R Westraven |
| Zuilenstein |
| Batau Noord |
| Wijkersloot |
| Stadscentrum |
| Merwestein |
| Fokkesteeg |
| Wiersdijk |
| Nieuwegein Zuid |

Table 6: Stops of the Nieuwegein tram line

In this case we used the gamma-distribution to fit our dataset, as it is generally suitable for service times. Another reason for choosing this type of distribution is that we have measurements of different scale (i.e. different distances between each stop) that we assume follow a more or less similar shape, so we can normalize their scale and fit a certain shape parameter (for every station). We, then, utilize the given distance averages between each stop to get the gamma-distribution’s scale and shape parameters, which will be used to generate random driving times in our simulation. On the right, we see a random fit of the driving times to a gamma-distribution and, on the left, we see the resulting global shape parameter :



|  |
| --- |
| Shape |
| 87.76 |

Table 7: Shape parameter of driving times

**Output Analysis**

**Questions answered by the experiments**

1. **Which are the feasible frequencies of the tram?**Having quantitatively analysed the question, we have a limited range of frequencies to consider. Assuming a turnaround time of minutes, we start simulations using the highest possible frequency . We then repeatedly increase this value to obtain lower frequencies. It
2. **What measures do you recommend to improve the operational performance?**  
   The simulation runs immediately suggest keeping the frequency below a certain threshold, in order to keep the passengers’ waiting time low. For instance, if we require the average waiting time to be below 1 minute, a possible measure to achieve this would be to set a frequency minutes.  
     
   Moreover, in order to keep the departure delay low, we can limit the frequency to be above a certain threshold. For example, if we require the average departure delay to be below

**Investigated scenarios**

We will examine two scenarios:

1. *Off-peak hours*: 06:00-07:00 and 19:00-21:30
2. *Peak hours*: 07:00-19:00

**#runs**

**Tables**

**Graphs**

**Observations from tables/graphs**

**Statistical analysis**

**Compare interesting scenarios (>= 10), using confidence intervals**

**EXTRA: Comparison with a standard**

**EXTRA: All pairwise combinations**

**EXTRA: Ranking and selection**

**Validation**

In order to validate our simulation system, we re-calculate the λ-rate of the exponential distributions from which our stochastic passenger inter-arrivals are generated. In order to do that, we use the given artifical input dataset and determine the new rate for each of the following periods:

* 06:00-07:00
* 07:00-09:00
* 09:00-16:00
* 16:00-18:00
* 18:00-21:30

Having done that, we acquire the following re-computed parameters (for brevity’s sake, we show the average rate across all stops):

|  |  |
| --- | --- |
| Time Periods | λ rate |
| 06:00-07:00 | - |
| 07:00-09:00 | - |
| 09:00-16:00 | - |
| 16:00-18:00 | - |
| 18:00-21:30 | - |

Table \_: Average rates λ

When we run our simulation with the re-computed parameters for our random number generators, we get the following results:

**Conclusions**

**Appendix: Interview minutes**

The meeting with a domain expert from the Qbuzz company validated some of our assumptions and certainly aided us on important decisions of our modelling process, as well as provided us with hard constraints on the actual operation of the tram line.

Regarding some sane assumptions we can make in our simulation:

* No failures anywhere in the tram line.
* No driver breaks or changes, everything is ideally done in zero time.

Regarding the operational aspects of the tram line we concluded the following facts:

* Simulation begins at 6 a.m. and runs for the whole day until 10 p. m.
* There is a frequency of 15 minutes during the off-peak hours (06:00-07:00 and 19:00-22:00) and parametric frequenct of minutes during the peak hours (07:00-19:00).
* The first schedules at both endstations must be absolutely satisfied, i.e. the first tram has to leave no later than 06:00.
* The last schedules at both endstations should not be delays, i.e. no tram departure after 21:30.
* Trams can only pop-up at the *P+R* station and only during the start of the simulation and during the switch from off-peak to peak hours (07:00).
* Tram can only withdraw from the *P+R* station during the end of the simulation and during the switch from peak to off-peak hours (19:00).
* There should be a 40-sec safety interval between trams, which is only asserted at the stations and not in between.
* The turnaround time is at least 3 minutes, but includes the .

Regarding the endstations’ layout, we were given a simple explanation of the switch mechanism that takes place, as shown in the following picture:

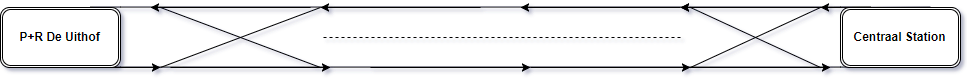


Figure Endstation layout

Finally, there are 2 possible models of dwell times at the stations (values in seconds):

1. *Non-stochastic option*

where is the number of passengers that board the tram and is the number of passengers that disembark.

1. *Stochastic option*

which is drawn from a γ-distribution with shape parameter and minimum value .