# Evaluating Passenger Seating Capacity at NS Using Simulation

Master Thesis
Econometrics & Management Science
Operations Research and Quantitative Logistics

*Author:* Nemanja Milovanović[1]
*Supervisor:* Prof. Dr. Dennis Huisman
*Co-reader:* Prof. Dr. Ir. Rommert Dekker

Erasmus School of Economics
Erasmus University Rotterdam

September 16, 2016

[1]Studentnumber: 378244

## Abstract

In this thesis we develop a discrete-event simulation framework capable of evaluating the KPI "transport capacity during rush hour". The problem is split into several subproblems, namely (i) modeling passenger arrivals, (ii) representing the planned timetable and rolling stock schedule, and (iii) determining passenger routes. Finally, we consider the impact of deviations from the planned timetable and rolling stock schedule in the form of train cancellations and rolling stock mismatches.

In order to evaluate our simulation framework, we use passenger check-in/check-out data, and the planned timetable and rolling stock schedule for January 12 in order to predict the KPI of February 9. We conclude that we are adequately able to predict passenger arrivals, but tend to underestimate the KPI with respect to the realized KPI by about 3%. Furthermore we provide some insight into the KPI. We see that overall both train cancellations and rolling stock mismatches negatively impact the KPI, although the effect of mismatches is much larger. We also notice that mismatches have different effect for different compositions: for high-capacity compositions the effect is negative, but low-capacity compositions actually benefit from rolling stock mismatches.

# Contents

# Chapter 1

# Introduction

## 1.1 Company Background

Netherlands Railways (NS) is the largest provider of passenger rail transportation in the Netherlands, serving over 1.1 million passengers everyday in about 5,200 trains. In total, NS employs about 30,000 people. Furthermore, NS holds the concession for the main rail network in the Netherlands for the period 2015-2025, meaning that in this time-window only NS may operate on this network. As a result, the only competition NS faces regarding rail transit is on the regional lines, by companies like Arriva and Veolia.

This research conducted in this thesis is performed at the department of Process quality & Innovation (PI). PI is a research department that is mainly concerned with (as the name suggests) maintaining and innovating various processes at NS. Typical products of PI are decision support tools in the form of computer-implemented mathematical models regarding timetabling, line planning, crew scheduling, and rolling stock rescheduling.

## 1.2 Thesis Motivation

One of the responsibilities of PI is the *handover letter* from the department of Network Design (which PI is a part of) to the Operations Control[1] department. Network Design is responsible for designing the timetable and the accompanying rolling stock schedule, among others. On the other end of the spectrum, Operation Control controls the execution of the plans that are in this handover letter. The letter also reports on the *expected quality* of these

[1]The department names have been translated from Dutch. The original names for *Network Design* and *Operations Control* are *Netwerk Ontwerp* and *Besturing Operatie*, respectively.

plans in the form of predicted *key performance indicators* (KPIs). A KPI is a method for a company to measure its performance, and can be both quantitative and qualitative. Examples of KPIs are customer acquisition, customer satisfaction, and punctuality.

Due to the rise of the smart card in Dutch public transit, more detailed passenger travel information can be gathered. Therefore NS has decided to redefine one of the KPIs in the handover letter, namely the KPI "*transport capacity during rush hour*". In the remainder of this thesis, the term "KPI" will refer to exactly (and only) this KPI.

Knowing where the bottlenecks are in terms of this KPI, or where service can be raised cheaply is vital for NS in order to improve customer service as efficiently as possible. The importance of this has become more apparent lately, as recently there has been a lot of pressure from both the public as from politics due to many complaints about a lack of seats (Bos, 2015; Baars, 2015), and overall crowdedness (Duursma, 2015; Van Houwelingen and Voermans, 2015) in passenger trains.

Therefore, PI has opted for a simulation model, which will be developed in this thesis. The main allure of using simulation is the freedom it gives in modeling. Additionally, it is very difficult to model phenomena like passenger route choice on such a scale, as the entire Dutch main rail network needs to be considered. Of course, there are also downsides to simulation. A good example is that it requires a lot of computational effort due to model complexity (even more so with stochastic models). Also, where analytic models are very transparent, and often give important insight into inner workings, simulation models can be seen as a black box and insight needs to be retrieved through statistics and sensitivity analysis.

The main goal of this thesis is to provide a simulation framework able to predict the new KPI. However, it is also desirable to be able to obtain insights from it. Therefore, we aim to answer the following two questions:

1. *How does one use simulation to predict the KPI* "transport capacity during rush hour*"?*

2. *What insights can be derived from the simulation model?*

## 1.3 Overview

The thesis is outlined as follows. Chapter 2 describes the problem at hand. Next, the relevant literature is discussed in Chapter 3. Then, the simulation model is split into several components. First, Chapter 4 describes how to model passenger arrivals into the system as non-homogeneous Poisson processes, and also gives two methods to approximate the intensity function $\lambda(t)$. Second, we look at the issue of timetable representation and passenger routing preference in Chapter 5. Third, Chapter 6 shows how to incorporate deviations from the planned timetable and rolling stock schedule into the simulation framework. Fourth, the discrete-event simulation model that is at the core of this thesis is described in Chapter 7. Next, the results are reported in Chapter 8, which contain parameter choice and sensitivity experiments, and also the simulation results. Finally, the thesis is concluded in Chapter 9.

# Chapter 2

# Problem Description

This chapter aims to define the problem at hand. To this end, first the terminology used in this thesis regarding timetables, rolling stock, and passengers is introduced in Section 2.1. Then, the definition of the KPI is given in Section 2.2. Finally, the challenges of the simulation framework proposed in this thesis are discussed in Section 2.3.

## 2.1 Terminology

The terminology sed in this section (including examples) mainly originates from Nielsen (2011).

### 2.1.1 Timetable

A *timetable* is a set of *train services*, where a train service is a train that goes from one terminal station to another with some intermediate stops along the way. A train service then is dividable into *trips*, during which a train transports passengers from one station to the next in the train service. Every trip has a departure time $t_d$, arrival time $t_a$, departure station $S_d \in \mathcal{S}$, and arrival station $S_a \in \mathcal{S}$. Thus, a trip executed by a train with number $Z \in \mathcal{Z}$ is characterized by the tuple $(Z, t_d, t_a, S_d, S_a)$. A *line* is a sequence of stations visited by the same train service. For the remainder of this thesis, we shall denote with $\mathcal{T}$ the set of trips in the union of all train services.

Figure 2.1 depicts the 3000 line, which has train services running from Den Helder (Hdr) to Nijmegen (Nm), with some intermediate stops.

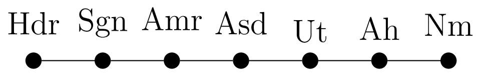


Figure 2.1: An example of a line: the 3000 line running from Den Helder (Hdr) to Nijmegen (Nm).

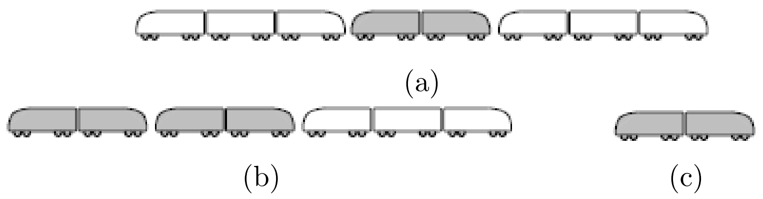


Figure 2.2: Three examples of train compositions: (a) three rolling stock units with 8 carriages total, (b) 3 units with 7 carriages total, and (c) 1 unit with 2 carriages total.

### 2.1.2 Rolling Stock

*Rolling stock units* at NS are fully independent units with a fixed number of carriages. The set of different rolling stock types will be denoted by $\mathcal{M}$. Often, rolling stock units are combined into what is called *compositions*. A *duty* is a time-ordered sequence of trips for a single day that is executed by the same rolling stock unit. Also, a train unit is used as a synonym for a rolling stock unit.

Figure 2.2 provides three examples of train compositions. In (a) we see a train composition consisting of 3 rolling stock units with a total of 8 carriages. The composition in (b), however, consists of 3 units with 7 carriages total. Finally, in (c) we see a train composition consisting of a single train unit with 2 carriages.

### 2.1.3 Passenger Traveling

A passenger of NS makes a *journey* from station $A$ to station $B$, where $A, B \in \mathcal{S}$ and $\mathcal{S}$ is the set of stations that are operated. Journeys at NS are registered by means of a check-in at the origin station, and a check-out at the destination station. Now, a *passenger route* $r$ states exactly how that journey is executed. In our scope, a passenger route (or *path*) is expressed as a sequence of trips. Note that a journey can generally be executed by more than one route. Also, we would like to stress that although we can always observe the journey a passenger makes, we have to infer the route which is traveled by, as in Dutch public transit passengers (more specifically, their smart cards) are not always checked intermediately. Furthermore, we make the distinction between a *planned* and a *realized* passenger route. A planned route is the route that we infer for the passengers, using the planned timetable and rolling stock schedule in the handover letter. A realized route is the route that the passenger actually traveled by. Planned and realized routes do not have to coincide due to changes in timetables and rolling stock schedules, or due to inaccurate modeling of the passenger's travel preferences.

## 2.2 KPI Definition

The term *transport capacity* in rail transit can have multiple meanings, depending on the context. In this thesis, however, *transport capacity* refers to the transport of passengers. Furthermore, although the KPI is named "transport capacity", the goal is not to determine how many passengers can be transported. Instead, "transport capacity" is measured by means of the *seating probability*, which includes both normal and foldable seats. The seating probability entails the probability of a random passenger boarding a random train being able to find a seat.

If we denote the number of passengers boarding the train executing trip $t$ as $b_t$, of which $f_t$ ($0 \leq f_t \leq b_t$) find a seat, then for a set of trips $X$ the KPI is calculated (weighing according to passenger boarding volumes) as

$$KPI = \frac{\sum_{t \in X} \left( b_t \cdot \frac{f_t}{b_t} \right)}{\sum_{t \in X} b_t} \tag{2.1}$$

$$= \frac{\sum_{t \in X} f_t}{\sum_{t \in X} b_t}. \tag{2.2}$$

In the official definition, the set $X$ is the set of all trips executed during both morning and evening rush hour, where morning rush hour is defined as the period 07:00–09:00, and evening rush hour 16:00–18:00.

Note that $f_t$ can be calculated using $c_t^{\text{seat}}$, $b_t$, and $n_t$. We want to determine the number of passengers able to find a seat. Before the train departs, we have $n_t$ passengers on board. Of these $n_t$ passengers, $n_t - b_t$ were already on board of the train executing trip $t$. The number of available seats for the people boarding the train is $a_t = [c_t^{\text{seat}} - (n_t - b_t)]^+$, where $x^+ \equiv \max\{x, 0\}$. Finally, the number of boarding passengers able to find a seat is given by $f_t = \min\{a_t, b_t\}$.

The following example illustrates the calculation of the KPI. Let $X = \{1, 2\}$ represent a time-ordered set of trips, where trips are executed in chronological order. For both trips, let the seating capacity be equal to

5. Assume that the rolling stock executing the trips in $X$ starts out empty. Then, let $b_1 = 3$ and $b_2 = 5$ and, for simplicity's sake, assume no passengers leave the train. Now, filling in Equation (2.2) gives

$$KPI = \frac{3+2}{3+5} = 0.6250.$$

## 2.3 Simulation Framework

We identify the following questions that the simulation framework must answer in order to determine the KPI:

- How do we represent the timetable and rolling stock schedule in a useful manner?
- How do we model the passenger arrival process?
- How do we determine passenger routes?

The term "system" refers to the scope of the simulation. By modeling arrivals of passengers (with a certain destination) into the system, we know the passenger volumes for each journey. Then by representing the timetable, we are able to determine the passenger routing. Together, the rolling stock schedule, and passenger volumes enable us to obtain for each trip (i) how many passengers have boarded, (ii) how many were already on the train, and finally, (iii) the (seating) capacity. With this information we are able to compute the KPI. In the remainder of this section, we discuss these problems briefly.

### 2.3.1 Timetable Representation & Passenger Routing

One of the first actions that the simulation performs is importing the planned timetable and the accompanying planned rolling stock schedule. However, the data representation must be such, that our passenger routing algorithms are able to use the timetable. As routing algorithms are usually graph-based, it makes sense to represent the timetable as a graph. However, a timetable has two dimensions, namely space and time. Although it is often intuitive how space is modeled by a graph, including time complicates the manner somewhat.

Next, as the only information we have on passenger traveling is their journey (i.e., their check-in and check-out time, and their origin and destination), we need to somehow infer the route they traveled through the rail network.

With the rise of the smartphone, passengers are up-to-date with the latest travel information. They also have access to the NS "reisplanner"[1] application, which provides updated travel information. With this application, all smartphone users are able to retrieve shortest paths at the press of a button. More familiar commuters, however, may prefer more comfortable trips and therefore might endure a slightly longer route if it saves them one or more transfers.

### 2.3.2 Passenger Arrival Process

Having determined *how* passengers travel through the rail network, to determine the KPI we also need to know *how many.* Modeling arrivals into a system is a well-studied topic in Operations Research, and appears in numerous fields like call center capacity design, revenue management, and supply chain management. The major problem faced in modeling arrivals is how to deal with inhomogeneity over time. Typically, arrivals vary over time, and usually one can derive peak hours. This is not different in our case, as we expect two major peaks; the morning and evening rush hour. Another modeling choice we need to consider is how we define an arrival. Do we simply model passengers arriving at a train station, onto platforms, or do we model arrivals in some other way?

### 2.3.3 Effects of External Factors

If all goes according to the plans in the handover letter, then we are done after we have solved above problems. In practice, unfortunately, this is not the case. There are many external factors influencing rail operations, like kids playing beside the tracks, bad weather conditions, suicides, and material malfunctions. It is impossible to directly model these factors due to them being unknown, we could, however, attempt to model the *effects* they cause. To simplify things a bit, we only consider effects that we think have considerable impact on transport capacity, namely:

- Train delays.
- Train cancellations.
- Changes in the rolling stock schedule.

First is the most common disruption, namely train delays. When a delay occurs, there is more time for passengers to arrive so it is expected that the

[1]`http://www.ns.nl/reisplanner/#/`

train will be busier, negatively influencing passenger travel capacity. Next, when a train is canceled altogether, we expect that most passengers take the next train. This adds to the passengers that already planned to take that other train, again being bad for the KPI. Finally, we consider changes in rolling stock with respect to the planned schedule, which might occur due to malfunctions or disruptions from previous days. When such a rolling stock mismatch occurs, it might happen that the substitute rolling stock has lower capacity, again hurting passenger travel capacity.

### 2.3.4 Simulation Outline

Finally we provide a simple outline of the simulation process, which is illustrated in Figure 2.3. In this figure, rounded rectangles represent processes, ellipses represent input data, and arrows show the general flow of the simulation. The dashed arrows signify a cause-effect relation. For example, disturbances affect both the realized timetable and the realized rolling stock schedule, and the realized timetable affects which passenger routes are chosen.

First, the simulation processes the planned timetable and planned rolling stock schedule. Then, arrivals are generated based on the CiCo data, and disturbances are generated. Next, given these disturbances, passenger routes are inferred and executed. Finally, we calculate the KPI and a report is generated.

Note that the disturbances affect both the timetable and the rolling stock schedule, which results in the *realized* timetable and rolling stock schedule. The realized timetable is used to determine passenger routes, and the realized rolling stock schedule to compute the KPI.

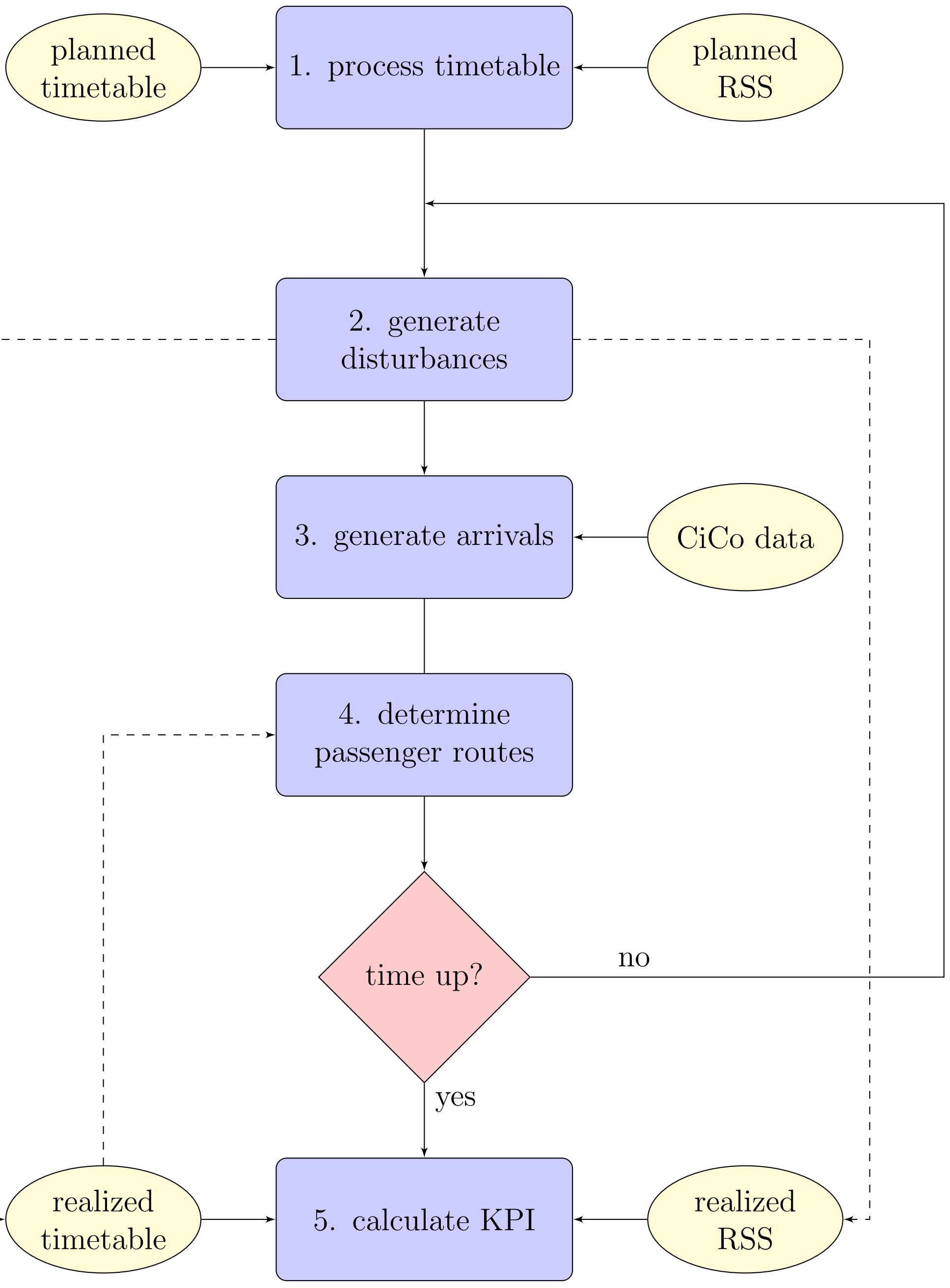


Figure 2.3: Flowchart giving a simplified view of the simulation framework. Rounded rectangles denote processes, ellipses denote input data. Also, “RSS” stands for “rolling stock schedule”.

# Chapter 3

# Literature Review

In this chapter we briefly discuss some methods we have found in the literature which we deem relevant for the problems identified in Section 2.3. We do not presume this discussion to be exhaustive, as there are many different angles one could explore. Section 3.1 discusses some papers with methods to fit non-homogeneous Poisson processes to arrivals, whereas Section 3.2 discusses some angles on modeling passenger route choice. Finally, this chapter is concluded with Section 3.3, which provides a quick glance at two methods to represent timetables that have been proven useful in railway applications.

## 3.1 Passenger Arrivals

Arrivals into any system have often been modeled as a stochastic process, most notably the Poisson process. In its most basic form, the *Poisson process* assumes that the arrival rate is constant over time, which is not a plausible assumption in many fields where it is applied. The problem is that often several peaks can be distinguished in the data, e.g. often call centers experience more calls during daytime than during nighttime. Thus, the assumption of a constant arrival function is relaxed, resulting in what is called a *non-homogeneous Poisson process* with an arrival function $\lambda(t)$, dependent on time.

Now it becomes interesting, as $\lambda(t)$ is allowed to be any positive piecewise function. If historical data is present, several methods exist in order to find a candidate $\lambda(t)$. A popular method is to divide the time-horizon into several subintervals (not necessarily of the same length), where for each interval the rate of arrival is determined. Then, each subinterval $i$ constitutes a Poisson process with arrival rate $\lambda_i$, as proposed in Brown et al. (2005) for call centers. This methodology is also proposed by Wang et al. (2015), the only paper found which considers arrival processes in public rail transit. Although the method is easy to implement, the constant rate ignores the (often) smooth transition of arrival rates, causing the rate to “jump” from one level to the other. Of course, the “jumping” behavior can be mitigated by decreasing the length of the intervals, but this also means reducing the number of data points in each interval.

In order to fix the “jumps” in the piecewise-constant arrival rate, smoother transitions are necessary. Massey et al. (1996) propose piecewise-linear rates instead of the piecewise-constant rates, and describe three ways to estimate the line parameters; ordinary least squares (OLS), iterative weighted least squares (IWLS), and maximum likelihood estimation (ML). The authors, however, only describe estimation for a single subinterval, though the same methodology can be employed for all subintervals independently. To avoid “jumps” form the end of one subinterval to the begin of the next, one needs to fix the endpoints of the lines, however.

Still, Alizadeh et al. (2008) note that piecewise-linear arrival rates suffer from abrupt changes. Therefore they suggest to approximate the arrival rate by *non-negative cubic splines*. A cubic spline is a function consisting of piecewise-defined cubic polynomials, often used in the field of interpolation. The result of the approximation by cubic splines is, contrary to piecewise constant and linear approximations, a smooth function. Due to the smoothness of the cubic spline, the arrival rate varies gradually over time.

However, the obtained smoothness comes at a price in the form of complexity. Whereas the case of piecewise-constant arrival rates involves almost trivial calculations, and piecewise-linear rates involve relatively simple OLS and ML estimation, fitting non-negative cubic splines involves solving second-order cone constraints with a non-linear objective function.

Due to this added complexity, we only consider piecewise-constant and piecewise-linear approximations.

## 3.2 Route Choice

Traditional passenger route choice modeling is based on utility maximization in the form of choice theory (Ben-Akiva, 1974; Oppenheim, 1993). Often, the multinomial logit model is used to model route choice (Ben-Akiva, 1974; Raveau et al., 2011). However, due to shortcomings of this model, alternatives have been proposed (Cascetta et al., 1996). With these utility models a database of realized passenger routes is often required in order to estimate the effects of route characteristics. At NS, however, regarding route choice only the check-in and check-out times and locations are registered, making this approach difficult to implement.

Recently, however, Van Der Hurk et al. (2015) compared several route generation and route selection methods as applied in railway transportation and validated them based on conductor counts. The results of this study are promising, with their best route deduction method selecting the correct route (the route a passenger actually traveled) in 95% of the cases. The reader should note, however, that these methods are only applicable when dealing with realized routes and timetables.

It is of note to report that passenger flow modeling is a subproblem in many railway optimization problems, ranging from line planning (Schöbel and Scholl, 2006) to rolling stock rescheduling (Kroon et al., 2014). However, to keep the model complexity in check, more often than not simplifying assumptions are made. For example, Cadarso and Marín (2011), and Haahr et al. (2014) assume that passenger flows are static (using point forecasts in their models), whereas Caprara et al. (2007) and Dollevoet et al. (2012) and others assume that passengers always choose shortest paths.

As no individual route preference data was available at the time of writing, we are forced to simplify our approach. We assume that passengers consider only two route characteristics, namely travel time and number of transfers. To aid us, we follow the approach proposed in Schulz (2005), which consists of solving a bi-criteria shortest-path problem with travel time and number of transfers for criteria.

## 3.3 Timetable Graphs

Finally we would like to mention the choices available in representing the train timetable in graph form. As the number of nodes and edges of a network directly affect the complexity of path algorithms, how the timetable is represented is of some importance.

Seemingly the most popular way of representing train timetables is using the so-called *event-activity network*, introduced by Nachtigall (1998). In this time-space network, nodes represent events, which in our context would be train departures, and edges represent activities, like a train traveling from one station to the other. An advantage of this approach is that it is very intuitive. If the network is drawn, one is easily able to see the paths that trains follow throughout the network.

Another, less popular, approach is to model the timetable as a dynamic time-dependent graph. In this graph, only space is modeled. Every station has his own node, and the edges between nodes (representing direct station-to-station connections) are time-dependent. For these type of networks, standard shortest-path algorithms are no longer an option due to the time-dependence of edge weights. Shortest-path algorithms for this type of network are given by Orda and Rom (1990) and Orda and Rom (1991).

Schulz (2005) compares both networks in the context of timetable queries. The results are that for basic queries time-dependent graphs are over a factor 10 faster. However, when including transfer times, the added nodes and edges amount to time-dependent graph queries being "only" 1.5 times faster, on average.

Also, including transfer time for time-dependent graphs is much more difficult. Therefore, we opt for the event-activity network approach in this thesis, seeing as the performance advantage for time-dependent graphs with transfer time is limited.

# Chapter 4

# Passenger Arrivals

## 4.1 Modeling System Arrivals

In the scope of this thesis, we assume a passenger to arrive at their origin station $s_O \in \mathcal{S}$ at time $t_O$, from which he or she travels to their destination station $s_D \in \mathcal{S}$. We do not assume to know an arrival time at their destination, as at this point in time the passenger has not yet decided his or her route. Furthermore, we assume passenger arrivals into the system to be random over time.

Borrowing from queuing theory, an obvious choice to model arrivals is by means of a non-homogeneous Poisson process. Therefore we suggest to fit such process for each OD-pair $(s_O, s_D)$. This way, we capture the difference in OD popularity. For example, a journey to a central station of a big city is often more popular than a journey to a station in some small village. This approach also has its drawbacks. For starters, we estimate an amount of processes in the order of $|\mathcal{S}|^2$. Due to this large number (about 26,000), statistical testing is restricted to hypothesis tests, which may be too strict. Moreover, a lot of journeys $(s_O, s_D)$ are quite unpopular (e.g., traveling between two remote train stations), meaning that in these cases data is usually very scarce.

However, we find our approach better than the alternative, which is to somehow aggregate. The problem here is that by aggregating you lose precious information. For instance, when aggregating by destination, you model passenger arrivals into the system as arrivals to their origin stations. In doing so, however, every arrival at station $s_O$ is treated the same, which is unfortunate. Take for example as origin station Rotterdam Alexander. The empirical arrival rate for February 2, 2016 is depicted in Figure 4.1. We clearly discern two peaks, coinciding with the morning and evening rush hour, which are about the same. Now we take a look at a subset of these arrivals, namely from Rotterdam Alexander to Utrecht Centraal, of which the empirical arrival rate of the same day is depicted in Figure 4.2. Again, two clear peaks are easily discerned, but the peak corresponding to the morning rush hour is way higher than the one corresponding to the evening rush hour. This is a typical example of commuter behavior, which is very difficult to capture when aggregating arrivals.

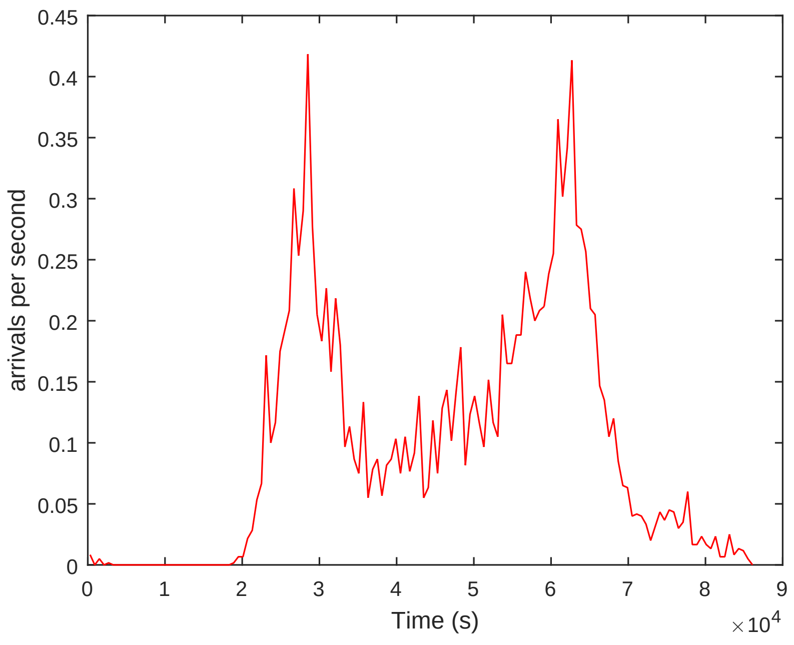


Figure 4.1: Empirical arrival rate of passengers with origin Rotterdam Alexander on February 9, 2016.

Fitting a non-homogeneous Poisson process is more involved than its homogeneous counterpart, as in this case the arrival rate $\lambda(t)$ is some function. Although technically one could fit any nonnegative function of $t$, we restrict ourselves to the family of piecewise-continuous nonnegative functions, where we consider the simple (and popular) piecewise-constant arrival rate, and the more involved piecewise-linear arrival rate.

## 4.2 Piecewise-Constant Arrival Rate

Fitting a piecewise-constant arrival rate is perhaps the simplest and most-popular estimation method for $\lambda(t)$. The process is as follows. Given a

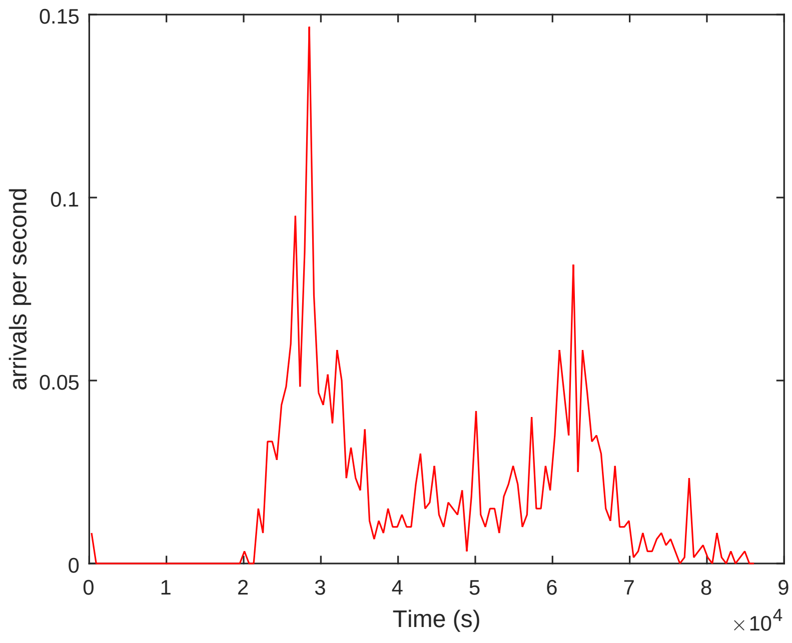


Figure 4.2: Empirical arrival rate of passengers with origin Rotterdam Alexander and destination Utrecht Centraal, on February 9, 2016.

time-interval $(0, T)$ over which passengers arrive in the system. Define knots $w_0, w_1, \ldots, w_m$, where $w_0 = 0 < w_1 < \cdots < w_m = T$ and $m$ is the number of intervals. Although the distance between knots does not need to be equal, for simplicity's sake we assume it to be. Now, the piecewise-constant arrival rate is specified as

$$\lambda(t) = \begin{cases} \lambda_1 & w_0 < t \leq w_1, \\ \lambda_2 & w_1 < t \leq w_2, \\ \vdots \\ \lambda_m & w_{m-1} < t \leq w_m. \end{cases} \tag{4.1}$$

In each interval $[w_i, w_{i+1})$ the process behaves as a homogeneous Poisson process. The log-likelihood function we use is the same as in Alizadeh et al. (2008):

$$\ell(\lambda) = \sum_{j=1}^{n} \ln \lambda(t_j) - \int_{w_0}^{w_m} \lambda(t)dt. \tag{4.2}$$

Expanding and evaluating the integral we get

$$\begin{aligned}\int_{w_0}^{w_m} \lambda(t)dt &= \int_{w_0}^{w_1} \lambda(t)dt + \ldots + \int_{w_{m-1}}^{w_m} \lambda(t)dt \\ &= \sum_{i=1}^{m} \int_{w_{i-1}}^{w_i} \lambda(t)dt \\ &= \sum_{i=1}^{m} \int_{w_{i-1}}^{w_i} \lambda_i dt \\ &= \sum_{i=1}^{m} \big[\lambda_i t\big]_{w_{i-1}}^{w_i} \\ &= \sum_{i=1}^{m} \lambda_i (w_{i-1} - w_i).\end{aligned}$$

Rewriting the summation in (4.2) (where $n_i$ is the number of arrivals in segment $i$) and plugging in the above result for the integration we get

$$\begin{aligned}\ell(\lambda) &= \sum_{i=1}^{m}\sum_{j=1}^{n_i} \ln \lambda_i - \sum_{i=1}^{m} \lambda_i(w_{i-1} - w_i) && (4.3)\\ &= \sum_{i=1}^{m} n_i \ln \lambda_i - \sum_{i=1}^{m} \lambda_i(w_{i-1} - w_i) && (4.4)\\ &= \sum_{i=1}^{m} \big[n_i \ln \lambda_i - \lambda_i(w_{i-1} - w_i)\big] && (4.5)\end{aligned}$$

Thus, the first-order conditions are

$$\frac{\partial \ell(\lambda)}{\partial \lambda_i} = \frac{n_i}{\lambda_i} - (w_i - w_{i-1}) = 0, \quad i = 1, \ldots, m,$$

which is easily seen to be solved by

$$\widehat{\lambda}_i = \frac{n_i}{w_i - w_{i-1}}, \quad i = 1, \ldots, m, \tag{4.6}$$

the commonly used estimator for homogeneous Poisson processes.

Although estimation of a piecewise-constant arrival rates is quite simple, the downside to this approach is that the rates are not continuous along its entire domain $(0, T)$, which causes abrupt changes in the transition from one interval to the next.

## 4.3 Piecewise-Linear Arrival Rate

### 4.3.1 Formulation

We attempt to resolve this issue by modeling $\lambda(t)$ as a piecewise-linear function continuous over $(0, T)$. Again, define knots $w_0, w_1, \ldots, w_m$, where $w_0 = 0 < w_1 < \cdots < w_m = T$ and $m$ is the number of intervals. The arrival rate $\lambda(t)$ now has the form

$$\lambda(t) = \begin{cases} a_1 + b_1 t & w_0 < t \leq w_1, \\ a_2 + b_2 t & w_1 < t \leq w_2, \\ \vdots & \\ a_m + b_m t & w_{m-1} < t \leq w_m. \end{cases} \tag{4.7}$$

Massey et al. (1996) proposes to estimate a piecewise-linear arrival rate, but this rate is only piecewise-continuous. Furthermore, the maximum-likelihood estimation is based on grouping arrivals during an interval into smaller partitions. The number of arrivals for each of these partitions is then Poisson distributed. However, the reader is left to determine the number of partitions for each interval.

On the other hand, Alizadeh et al. (2008) suggests a method to estimate the arrival rate using nonnegative cubic splines. Although this is an improvement to estimating a piecewise-linear arrival rate, because of the added complexity in estimation and the fact that we need to estimate a number of arrival processes in the order of $|\mathcal{S}|^2$, we do not consider this option.

We start deriving the log-likelihood function in the same way as before. Expanding and evaluating the integral in (4.2) we get

$$\begin{aligned} \int_{w_0}^{w_m} \lambda(t)dt &= \int_{w_0}^{w_1} \lambda(t)dt + \int_{w_1}^{w_2} \lambda(t)dt + \ldots + \int_{w_{m-1}}^{w_m} \lambda(t)dt \\ &= \sum_{i=1}^{m} \int_{w_{i-1}}^{w_i} \lambda(t)dt \\ &= \sum_{i=1}^{m} \int_{w_{i-1}}^{w_i} (a_i + b_i t)dt \\ &= \sum_{i=1}^{m} \big[a_i t + \frac{1}{2} b_i t^2\big]_{w_{i-1}}^{w_i} \\ &= \sum_{i=1}^{m} \big[a_i(w_i - w_{i-1}) + \frac{1}{2} b_i (w_i^2 - w_{i-1}^2)\big]. \end{aligned}$$

Plugging this result in (4.2) we get the log-likelihood function

$$\ell(\lambda) = \sum_{i=1}^{m}\sum_{j=1}^{n_i} \ln(a_i + b_i t_{ij}) - \sum_{i=1}^{m} \left[a_i(w_i - w_{i-1}) + \frac{1}{2} b_i (w_i^2 - w_{i-1}^2)\right], \quad (4.8)$$

where we rewrote the summation in the first term for convenience. (4.8) can be shown to be concave by inspecting its Hessian.

The piecewise-constant arrival rate estimation was rather straightforward, as only piecewise-continuity was considered, and the rate was inherently nonnegative. The piecewise-linear arrival rate estimation is a little more involved when considering continuity over $(0, T)$ and nonnegativity, as this must be taken into account explicitly.

First, we consider continuity over it domain $(0, T)$. For a piecewise-linear function we only need to reinforce continuity over the interval transitions. Thus, when transitioning from interval $i$ to $i + 1$ we require

$$a_i + b_i w_i = a_{i+1} + b_{i+1} w_i, \quad i = 1, \ldots, m - 1 \quad (4.9)$$

Next, we require $\lambda(t)$ to be nonnegative over $(0, T)$. Using the fact that a linear function over $[a, b]$ is negative if and only if it is negative at either $a$, $b$, or both, we enforce nonnegativity by requiring $\lambda(t)$ to be nonnegative at each knot $w_i$ using

$$a_1 + b_1 w_0 \geq 0, \quad (4.10)$$

$$a_i + b_i w_i \geq 0, \quad i = 1, \ldots, m. \quad (4.11)$$

Combining the log-likelihood function (4.8) with constraints (4.9)–(4.11) we obtain the non-linear program

$$\begin{array}{lllll} \max & \ell(\lambda) = \displaystyle\sum_{i=1}^{m}\sum_{j=1}^{n_i} \ln(a_i + b_i t_{ij}) - \sum_{i=1}^{m} \left[a_i(w_i - w_{i-1}) + \frac{1}{2} b_i (w_i^2 - w_{i-1}^2)\right] & & & \\ \text{s.t.} & a_i + b_i w_i & = & a_{i+1} + b_{i+1} w_i, & i = 1, \ldots, m-1, \\ & a_1 + b_1 w_0 & \geq & 0, & \\ & a_i + b_i w_i & \geq & 0, & i = 1, \ldots, m, \\ & a_i, b_i & \in & \mathbb{R}, & i = 1, \ldots, m. \end{array}$$

As $-\ell(\lambda)$ is convex (due to concavity of $\ell(\lambda)$) and the constraints are linear, this problem belongs to the family of convex optimization problems, which can be solved by interior point methods (see Boyd and Vandenberghe, 2004). Popular solvers for these type of problems are KNITRO (Byrd et al., 2006), IPOPT (Wächter and Biegler, 2006), and LOQO (Benson et al., 2002). As all of these are either commercial, exclusively for C/C++, or difficult to set up for Java (we experienced a bug for IPOPT during compilation), we solve the non-linear program with the freely available Java library jOptimizer[1].

### 4.3.2 Additional Constraints

If data is sparse for some segments, it might be that for those segments you have to resort to other estimation methods. In order to still be able to fit a piecewise-linear arrival rate to the remaining segments, we suggest the following. Suppose we already have estimated arrival rates for some groups of segments. Then we group the segments for which we want to estimate a piecewise-linear rate consecutively, as follows. Say we have 10 segments, numbered 1 to 10. If for segments 3 and 6 we already have arrival rate estimates, then we group the remaining segments as 1–2, 4–5, and finally 7–10. Now, for each of those groups we can estimate piecewise-linear arrival rates, but if we also want continuity, we need to add some constraints. Suppose one such group of segments is $i_L$–$i_R$, where segment $i_L$ start at time $t_L$, and segment $i_R$ ends at time $t_R$. Then, in order to guarantee continuity, if $t_L \neq 0$ we need to add the constraint

$$a_{i_L} + b_{i_L} t_L = \widehat{\lambda}_L(t_L) \tag{4.12}$$

to the original non-linear program, where $\widehat{\lambda}_L$ is the estimated arrival rate function of the segment preceding $i_L$. Next, if $t_R \neq T$ we also need to add

$$a_{i_R} + b_{i_R} t_R = \widehat{\lambda}_R(t_R), \tag{4.13}$$

where similarly $\widehat{\lambda}_R$ is the arrival rate function of the segment succeeding $i_R$.

### 4.3.3 Scaled Formulation

After testing the formulation, we found that jOptimizer was not able to solve quite a few problems. Alizadeh et al. (2008) suggest the following scaling to help with numerical instabilities. Instead of specifying each linear function over $[w_{i-1}, w_i]$, we rescale to $[0, 1]$. With $t \in [w_{i-1}, w_i]$, the arrival rate function with scaling is

$$\mu(t) = u_i^0 + u_i^1 \frac{t - w_{i-1}}{w_i - w_{i-1}} = u_i^0 + u_i^1 \frac{t - w_{i-1}}{d_i}, \tag{4.14}$$

[1]`http://www.joptimizer.com`

where $u_i^0$ is the intercept for segment $i$, and $u_i^1$ the slope. Note that due to the scaling, the parameters in (4.14) are generally different from the ones in (4.7), which we signify by using a different notation.

After similar algebraic manipulation as before, we obtain the the formulation for the scaled version in (4.15).

$$\begin{array}{ll} \max & \ell(\lambda) = \sum_{i=1}^{m}\sum_{j=1}^{n_i} \ln(u_i^0 + u_i^1 \frac{t_{ij} - w_{i-1}}{d_i}) - \sum_{i=1}^{m} d_i(u_i^0 + \frac{1}{2}u_i^1) \\ \text{s.t.} & \begin{array}{lcll} u_i^0 + u_i^1 & = & u_{i+1}^0, & i = 1, \dots, m-1, \\ u_i^0 + u_i^1 & \geq & 0, & i = 1, \dots, m-1, \\ u_1^0 & \geq & 0, & \\ u_i^0, u_i^1 & \in & \mathbb{R} & i = 1, \dots, m. \end{array} \end{array} \tag{4.15}$$

The constraints are analogous to the unscaled formulation. Note that as the linear functions are now defined on $[0, 1]$, $w_i$ is gone from the constraints.

The additional constraints (4.12) and (4.13) for the scaled version of the formulation become

$$u_{i_L}^0 = \widehat{\lambda}_L(t_L), \tag{4.16}$$

and

$$u_{i_R}^0 + u_{i_R}^1 = \widehat{\lambda}_R(t_R), \tag{4.17}$$

## 4.4 Sampling From a Poisson Process

Drawing from homogeneous Poisson process is rather straightforward due to inter-arrival times being drawn from an exponential distribution with the same constant rate. Unfortunately, when considering a non-homogeneous Poisson process, this procedure is no longer valid due to the arrival rate being time-dependent. Luckily, realizations can still be drawn relatively easily, by using a technique proposed by Lewis and Shedler (1979) called *thinning*.

The procedure can be seen as the process analogue of the acceptance-rejection method for drawing from distributions, and is described in Algorithm 1.

**Algorithm 1** Drawing from a non-homogeneous Poisson process by thinning.

```
1: procedure DrawByThinning($T$, $\lambda(t)$)
2:     $\lambda^* \leftarrow \sup_{0 \leq t \leq T} \lambda(t)$
3:     $t^* \leftarrow 0$
4:     $X \leftarrow \emptyset$                                  ▷ List of drawn numbers
5:     while $t < T$ do
6:         Draw $E$ from $Exp(\lambda^*)$                     ▷ Exponential distribution
7:         Draw $U$ from $U(0, 1)$                           ▷ Uniform distribution
8:         $t^* \leftarrow t^* + E$
9:         if $t^* < T$ and $U \leq \lambda(t^*)/\lambda^*$ then
10:            $X \leftarrow X \cup \{t^*\}$
11:        end if
12:    end while
13:    return $X$
14: end procedure
```

# Chapter 5

# Passenger Routing

## 5.1 Basic Timetable Representation

By far the most popular method of representing timetables for routing purposes is the *event-activity network* due to its intuitiveness and simplicity. In this representation a network is created with nodes posing as events (like arrivals and departures), and edges serving as activities (like executing a trip).

Given a set $\mathcal{T}$ of all trips we consider, the basic event-activity network representation of a timetable is given as follows. For each trip in $\mathcal{T}$, which can be characterized as seen in 2.1 by the tuple $(Z, t_d, t_a, S_d, S_a)$, there is a node corresponding to the departure and arrival events, where the departure node belongs to station $S_d$ and the arrival node belongs to station $S_a$. Linking these two nodes is the trip edge, with weight equal to $t_a - t_d$. Let $v_1, v_2, \ldots, v_k$ be the time-ordered nodes belonging to station $S$. Wait edges connect nodes $v_i$ and $v_{i+1}$ where $1 \leq i \leq k-1$. These wait edges represent transfer opportunities and train dwell times. As with the trip edges, their weight is equal to the difference in time between $v_i$ and $v_{i+1}$.

For example, consider Figure 5.1. Here, a timetable snippet is represented with 6 trips, executed by 3 trains. The first two trains start in station $A$, dwell for some time at station $B$, and carries on to station $C$. The third train executes its trips in reverse so that it eventually ends up at station $A$. Going into more detail, the odd numbered nodes are all departure nodes, whereas the even numbered nodes are arrival nodes. Furthermore, nodes 1, 5, and 12 belong to station $A$, nodes 2, 3, 6, 7, 10, and 11 belong to $B$, and finally nodes 4, 8, and 9 belong to $C$. The basic network contains 6 arrival nodes, 6 departure nodes, 6 trip edges, and 9 wait edges, totaling 12 nodes and 15 edges.

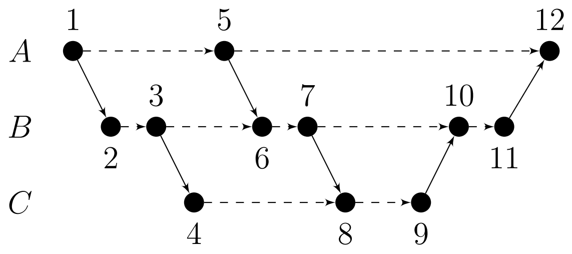


Figure 5.1: Small example of the basic event-activity network, with three trains traveling between three stations. Solid arrows represent trip edges, whereas dashed arrows represent wait edges.

## 5.2 Modeling Transfer Time

Although the basic event-activity network represented in the previous section is quite simple, it does result in unrealistic passenger routes as no transfer time is taken into account. In its ultimate form, it is even possible to incorporate platforms in the network, resulting in very detailed representations. We, however, restrict ourselves to the case of constant transfer times, as incorporating platforms increases the complexity of the simulation too much.

We call the event-activity network explicitly modeling constant-time transfers the *extended event-activity network*. For every trip $(Z, t_d, t_a, S_d, S_a)$, the network contains an arrival and departure node, and two transfer nodes. The transfer node corresponding to the departure node belongs to station $S_d$, and the transfer node corresponding to the arrival node belongs to station $S_a$. The edge connecting the transfer node at $S_d$ and corresponding departure node is a wait edge with zero weight. However, the arrival node at $S_a$ is connected by a transfer edge (if available) with the first transfer node at station $S_a$ with corresponding time after $t_a + c$, where $c$ is the constant transfer time. The edge weight (as usual) is equal to the difference in time between the nodes. Finally, for each station, if $v_1, v_2, \ldots, v_k$ is the time-ordered sequence of transfer nodes, $(v_i, v_{i+1})$ are the wait edges, for $i = 1, \ldots, k-1$. Again, the edge weight is equal to the absolute time difference.

To illustrate the extended network representation, we continue with the previous example of Figure 5.1. We additionally assume that the constant transfer time is such, that we are always able to catch the next departing train for the station we arrive on. In this figure, we arrival and departure nodes are black, whereas transfer nodes are gray. Moreover, trip edges are solid black, wait edges are dashed, and finally transfer edges are dash dotted. The first train to depart (in chronological order, assuming time flows from left to right) starts in station $A$. When it arrives at $B$, we have the choice to either *transfer* to another train (train 2 or 3, in this case), or we stay and continue to station $C$. The extended network contains 6 arrival nodes, 6 departure nodes, 9 transfer nodes, 6 trip edges, 12 wait edges, and 4 transfer edges, totaling 21 nodes and 22 edges, resulting in an overall much larger network.

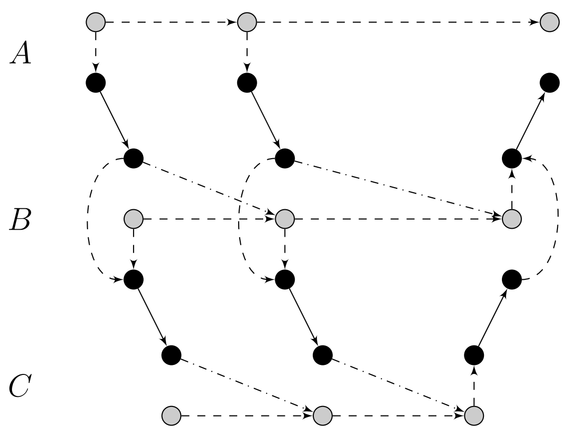


Figure 5.2: Extended network representation of the example in Figure 5.1. Black circles are arrival and departure nodes, whereas gray nodes are transfer nodes. Next, solid arrows correspond to trip edges, dashed arrows to wait edges, and finally dash dotted arrows to transfer edges.

## 5.3 Passenger Route Choice

In the literature, route choice is almost always modeled by means of random utility models. The modeling tool of choice in this field seems to be (a derivative of) the family of logit and probit models (Ben-Akiva, 1974; Oppenheim, 1993; Cascetta et al., 1996; Raveau et al., 2011). Following this way of modeling, one would need for each OD-pair a realized route and a way to generate plausible routes. Usually, these realized routes are derived from survey data.

Unfortunately, we were unable to acquire such data, meaning we are not able to pursue the random utility theory approach. Instead, we need to make simplifying assumptions. As such we decide to only consider two factors, namely total travel time and number of transfers. Although these factors were largely motivated by intuition, they are also mentioned by Schulz (2005) as important criteria in providing timetable information. Therefore, we have made the following assumption:

*Passengers always choose the earliest arriving path with minimum number of transfers.*

This results in a shortest-path problem where we minimize two criteria. Often with problems involving the optimization of two criteria, the concept of Pareto-optimality is utilized. We say a solution with objective value $(x_1, x_2)$ is Pareto-optimal if there exists no other solution with objective value $(y_1, y_2)$ such that $y_1 < x_1$ and $y_2 < x_2$. However, it is possible that numerous Pareto-optima exist. In our case, when considering travel time and number of transfers as criteria, one could create a rule which selects the most preferred Pareto-optimum, but this would require knowledge about passenger preferences, of which we have no data.

We therefore suggest selecting the *lexicographically first Pareto-optimum.* The lexicographical ordering is defined as

$$(a, b) \prec (a', b') \iff (a < a') \vee (a = a' \wedge b < b'),$$

meaning that the lexicographically first Pareto-optimum when considering criteria $(C_1, C_2)$ is the Pareto-optimum first minimizing $C_1$, and then $C_2$. In our case, we feel that minimizing travel time is the most important objective, so we let $C_1$ correspond to travel time, and $C_2$ to the number of transfers.

Before we provide an algorithm for the lexicographically first Pareto-optimum problem involving travel time and number of transfers, we shortly consider Dijkstra's algorithm for the shortest-path problem.

Dijkstra's algorithm is initialized by marking every node as *unvisited*, and every distance label of each node is set to $+\infty$. Let the current node being processed be $v$, and set it to be equal to the origin. Now comes the main loop of the algorithm. The distance labels of $v$'s neighbors are updated, and they are all marked as *visited.* Then, $v$ is set to be equal to the node marked *visited* with the lowest value for distance label. Now the loop repeats itself, until the very moment $v$ is set to be the target.

Fast implementations of Dijkstra's algorithm maintain a *priority queue*, which allows for fast retrieval of the *visited* node with smallest distance label. For a comparison of different priority queue implementations for Dijkstra's algorithm the reader is referred to Chen et al. (2007).

Our lexicographically first Pareto-optimum in bicriteria shortest-paths is solved in a modified Dijkstra's algorithm. Instead of keeping a distance label, we keep a label with both travel time and number of transfers, say $(x, y)$, where $x$ denotes total travel time and $y$ the total number of transfers. Now, in the label updating step, a neighbor of $v$ only gets his label updated if the new label precedes the old one in lexicographical order. When the algorithm terminates, it means that we have found a path from origin to destination that precedes all others in lexicographical order. The entire algorithm is depicted in Algorithm 2. Here, $\mathcal{N} = (V, E)$ is the network in which we find paths, $s$ is the origin node, and $t$ is the destination node. Note that in the algorithm $w(u, v) = (w_1, w_2)$ for $(u, v) \in E$ is the weight, where $w_1$ is the time between $u$ and $v$, and $w_2$ is 1 if $(u, v)$ is a transfer edge and 0 otherwise. Finally note that the variant of Dijkstra's algorithm we use comes from Chen et al. (2007).

**Algorithm 2** Dijkstra's algorithm modified to find the lexicographically first Pareto-optimum in bicriteria shortest-path problems.

```
1: procedure LexicographicallyFirstOptimum($\mathcal{N}$, $s$, $t$)
2:     $\mathcal{N} = (V, E)$                              ▷ Network with node and edge set
3:     $Q \leftarrow \emptyset$                                          ▷ Priority queue
4:     $d : V \to \mathbb{R}_+ \times \mathbb{R}_+$                            ▷ Label mapping
5:     for each $v \in V$ do
6:         $d[v] \leftarrow (+\infty, +\infty)$
7:     end for
8:     $Q$.insert($s$, $(0, 0)$)                         ▷ Insert origin into queue
9:     while $Q \neq \emptyset$ do
10:        $(u, \ell) \leftarrow Q$.removeMin()
11:        if u = t then
12:            return                   ▷ We found the destination, so terminate
13:        end if
14:        if $\ell \prec d[u]$ then
15:            $d[u] \leftarrow \ell$                                      ▷ Update label
16:            for each $(u, v) \in E$ do
17:                if $d[u] + w(u, v) \prec d[v]$ then
18:                    $Q$.insert($v$, $d[u] + w(u, v)$)
19:                    $d[v] \leftarrow d[u] + w[u, v]$                      ▷ Update label
20:                end if
21:            end for
22:        end if
23:    end while
24: end procedure
```

# Chapter 6

# Deviating From the Plans

Almost every day the execution of a timetable and/or rolling stock schedule did not go as planned. Although small deviations (like a minute delay) are manageable, and bearable for the public, larger deviations often do cause problems. In this chapter, we consider two types of deviation. Section 6.1 describes what we understand by rolling stock mismatch and how to model them, while Section 6.2 is concerned with train cancellations.

## 6.1 Rolling Stock Mismatches

Sometimes in the execution of a timetable, it is impossible for a train to run in the composition it was planned to. This often occurs when, due to disruptions from previous days, rolling stock units did not end their day where they were supposed to. If a train were to be planned to run as a VIRM4-VIRM6, it could be that at that moment at that location no VIRM6 is available. Instead, it could be that the best available alternative is to operate it with a VIRM4-VIRM4. We call this mismatch between the planned and realized timetable a *rolling stock mismatch.*

In order to model these mismatches, we look at the moments in time that compositions are formed or changed. We assume that this is at the beginning of a train service, and at intermediate train stations as designated in the *planned* timetable. Denote $\mathcal{C}$ as the set of possible compositions.We denote $\pi_{c_{\text{plan}},c_{\text{real}}}$, where $c_{\text{plan}}, c_{\text{real}} \in \mathcal{C}$, as the probability that given a planned composition $c_{\text{plan}}$ for some composition formation or change, the actual composition is $c_{\text{real}}$.

Suppose we have historic data of the planned and realized timetables.

Then, $\pi_{c_{\text{plan}},c_{\text{real}}}$ can be estimated by

$$\widehat{\pi}_{c_{\text{plan}},c_{\text{real}}} = \frac{n_{c_{\text{plan}},c_{\text{real}}}}{n_{c_{\text{plan}}}}, \quad \forall c_{\text{plan}} \in \mathcal{C}, \tag{6.1}$$

where $n_{c_{\text{plan}},c_{\text{real}}}$ denotes the number of composition formations or changes where composition $c_{\text{plan}}$ was planned, but $c_{\text{real}}$ is realized, and $n_{c_{\text{plan}}}$ is the number of composition formations or changes in the historic data where $c_{\text{plan}}$ was planned.

The method we use unfortunately has some disadvantages. For example, we ignore the actual availability of rolling stock units present at any location at any time, and we also ignore composition restrictions imposed by railway infrastructure. As an example, a VIRM6-VIRM6 is not allowed to dwell at relatively small stations due to its length.

## 6.2 Train Cancellations

Although it does not occur that often, sometimes due to various reasons (bad weather, technical problems) NS decides to cancel a train service. A direct result is that every passenger that planned on taking that canceled train service, needs to find an alternative, which is usually the next train. This means that this next train receives more passengers than expected, often resulting in a busy train.

There are two ways one can model train cancellations. The way that is more true to reality is modleing them *dynamically*, by which we mean that cancellations occur during simulation, and happen “unexpectedly”. An easier approach, the one taken in this thesis, is to model cancellations *statically*, or, before execution of the simulation. The main difference between the two methods is that when modeling cancellations statically, we assume passengers are aware of future cancellations. In contrast, when modeling them dynamically, passengers only become aware of cancellations when they occur.

If we assume that the probability of a train service being canceled is independent of all other events, and if we assume that the probability of canceling a train service is equal for all services, then services are canceled as follows. For each train service in the planned timetable, with probability $\psi$ we remove all trips associated to it.

# Chapter 7

# Simulation Framework

## 7.1 Discrete-Event Simulation

In order to determine the KPI according to the old and new definition, we use *discrete-event simulation* (DES). DES is a way of simulating systems (both deterministic and stochastic of nature) by generating and processing events. This allows for an efficient simulation of systems that only change their state at discrete times.

A DES usually consists of the following components:

1. A system state;
2. An event queue;
3. An event handler;
4. Statistical counters.

The *system state* can be seen as a snapshot of the simulated system at a certain point in time. The *event handler* is in charge of retrieving *events* from the *event queue* and passing them on to the *event handler*. The event queue (often called *event list*) stores the events in a particular order. Often in implementations a binary heap is used, which is able to store and retrieve objects in $O(\log n)$ time. Next, the event handler processes the event according to its *type*. In our implementation, the event handler passes the event to the appropriate *callback method*, which in turn processes the event. Each event type has its own callback method. Lastly, the statistical counters are perhaps the most important in a DES, as they are used to calculate the value of interest.

The goal of a DES is to process events and update the statistical counters accordingly, so that the value of interest can be calculated. To achieve this goal, a DES can be split into several phases:

1. Initialization;
2. Event processing;
3. Calculation of performance measures.

The initialization phase loads and processes the data, and queues events. Next, the events are processed by the event handler. After all events are processed, the statistical counters are used to calculate the value of interest. In the case of a stochastic simulation, multiple runs are performed and (usually) the mean and standard deviation of the value of interest are reported.

## 7.2 Model Specification

Now we present our DES. The choice for the counters is obvious, as they follow directly from the definition in Section 2.2. Hence, for every trip $t$ we calculate

- $b_t$: the number of people boarding the train right before executing trip $t$;
- $n_t$: the number of people on board during execution of trip $t$.

Following the choice for the counters, the system state consists of the norm ($c_t^{\text{norm}}$) and seating capacity ($c_t^{\text{seat}}$) for each trip $t$, as well as the event-activity network corresponding to the planned timetable, all estimated arrival processes, and all derived passenger routes.

In a DES, the system state can only be changed by processing events. To determine $n_t$ and $b_t$, we need events for every passenger at boarding and alighting. Therefore we introduce the *boarding event* $e_t^{\text{board}}$ denoting that a passenger has boarded the train executing trip $t$ at the corresponding departure station. On the other hand, the *alighting event* $e_t^{\text{alight}}$ denotes that a passenger has alighted the train executing trip $t$ at the arrival station.

The order in which we store the events is according to the execution time of the trip they belong to, where alighting events have priority over boarding events in case they belong to the same trip. Here we assume the common courtesy of allowing other passengers to alight before boarding a train.

### 7.2.1 Initialization

The DES is initialized as follows. First, the check-in/check-out data is imported. Then for each OD-pair piecewise-linear arrival rates are estimated using formulation (4.15) in Section 4.3. Note that the log-likelihood function contains the term $\ln(u_i^0 + u_i^1 \frac{t_{ij}-w_{i-1}}{d_i})$, meaning that it is impossible to estimate a zero rate ($u_i^0 = u_i^1 = 0$) as $\ln 0$ is not defined. So, before solving (4.15) we set $u_i^0 = u_i^1 = 0$ for all segments $i$ without arrivals. Then we group all other segments together and estimate piecewise-linear arrival rates for each group. Unfortunately, the non-linear solver we used (jOptimizer) was not able to solve all problem instances. For those that remain unsolved we estimate piecewise-constant rates using Equation (4.6).

Having estimated the arrival rates, we now draw from the processes using Algorithm 1. Next, for each passenger arrival drawn, a path is generated from origin to destination using Algorithm 2. Boarding and alighting events are created for boarding at the origin and alighting at the destination, respectively. If the path contains one or more transfers, then for each transfer an alighting event is created for the trip containing the transfer station as arrival station, and a boarding event is created for the trip with the transfer station as departure station. All events are of course then added to the event queue. Next, if we allow rolling stock mismatches, we go through all planned train services and determine for each time the composition of a service is assembled or altered, what the realized composition will be according to the estimated probabilities $\widehat{\pi}_{c_{\text{plan}},c_{\text{real}}}$. Upon realization, we alter the corresponding state variables $c_t^{\text{norm}}$ and $c_t^{\text{seat}}$. Finally, we set $n_t$ and $b_t$ equal to zero for all trips $t \in \mathcal{T}$.

### 7.2.2 Event Callback Methods

After initialization, the event handler processes all events in the queue by means of the corresponding callback method. As we have two event types, we discern two different callback methods:

**Boarding event callback** As a boarding event $e_t^{\text{board}}$ signifies one passenger boarding the train executing trip $t$, we increment both $b_t$ and $n_t$.

**Alighting event callback** An alighting event $e_t^{\text{alight}}$ signifies one passenger alighting the train executing trip $t$, so we decrement $n_t$ by one upon processing.

### 7.2.3 Implementation Note

We note that every simulation replication is independent, meaning they can be executed in parallel. Do note, however, that every replication is reliant on its own timetable and therefore event-activity network. When running $m$ instances in parallel, $m$ copies of timetables and networks are needed to be loaded into memory at the same time, resulting in quite some memory overhead. For instance, using typical input, and taking $m = 4$, the simulation needed just over 4 GB of RAM.

# Chapter 8

# Results

## 8.1 Data

In this thesis, we use the planned rolling stock schedule and timetable for February and March 2016. This dataset is cyclic in the sense that it repeats itself every week. Moreover, it contains 4,263 different train services, executing a total of 253,599 trips. Furthermore, we disregard all night and international trains.

In addition, NS has also provided check-in/check-out data for January 12 and February 9. Tuesdays were chosen, as it seems to be the most representative weekday regarding rail transit behavior. In these datasets, all smart card passenger journeys are registered, meaning we know the time and station where they checked in and out. Due to privacy concerns, we only mention that the datasets contain about one million records.

In addition to passengers checking in and out, there is also a significant number who travel without doing so. The research "meten in de trein" (MidT) provided us with correction factors for each type of product issued by NS. Unfortunately we have to aggregate these factors into one, as otherwise we need to estimate a different arrival process for each OD-pair for each product type. We aggregate the correction factors by taking a weighted average, with the weights corresponding to the passenger volumes from the check-in/check-out data. The resulting factor is $k_{\text{cor}} = 1.2158$. We then take into account this correction by applying it to the statistical counters, thus obtaining the corrected counters $\tilde{b}_t = k_{\text{cor}} \cdot b_t$ and $\tilde{n}_t = k_{\text{cor}} \cdot n_t$.

Lastly, we have rolling stock schedule and timetable realization data for January 25 until February 8 (15 days), which contain both planned and realized arrival and departure times, and train compositions, for all trips executed by NS. This dataset contains 786,338 records.

## 8.2 Arrival Rate Approximation

Chapter 4 discussed two methods for approximating the arrival rate for a non-homogeneous Poisson process, namely by a piecewise-constant and by a piecewise-linear function. Both methods are parameterized by segment width $d$. However, no methods were given to determine which of the two perform better, or how to choose $d$. We now provide such a method, by comparing the approximations with the empirical rate.

The arrival rate only impacts the number of passengers boarding a certain train. Thus, let $b_t^{\text{emp}}$ and $b_t^{\text{appr}}$ be the number of passengers boarding the train executing trip $t$, under the empirical and approximated arrival rate, respectively, where passengers were routed using Algorithm 2. We then define the loss $L$ as

$$L = \sum_{t \in \mathcal{T}} \left(b_t^{\text{emp}} - E[b_t^{\text{appr}}]\right)^2,$$

where the closer the approximation is to the empirical rate, the lower the value of $L$.

We consider the piecewise-constant and piecewise-linear approximations for $b_t^{\text{appr}}$, with $d \in \{1, 3, 5, 10, 15, 20, 30\}$ minutes, we obtain the results as depicted in Figure 8.1 using the January 12 check-in/check-out dataset.

The results depicted in Figure 8.1 were calculated using 16 simulation runs per value for $d$. The figure clearly shows that the piecewise-linear approximation is superior. As expected, the larger $d$, the worse the approximation tends to be (in both cases). We also see that for both cases the difference between choosing $d$ to be 1, 3, or 5 is very small, although for the piecewise-linear approximation $d = 1$ minute scores the best.

Figure 8.2 depicts the predicted and realized arrival rates for the morning rush of passengers traveling from Rotterdam Alexander to Utrecht Centraal, for the piecewise-constant ($d = 3$) and piecewise-linear ($d = 1$) approximations.

## 8.3 Transfer Time

Next, we take a look at the impact of transfer time to the KPI. We consider the transfer windows 0, 1, 2, 3, 5, 7, and 8 minutes. For an average person, 8 minutes gives a big enough transfer window for even the largest train stations. Furthermore, we set $d = 1$ minute, and use the piecewise-linear arrival rate approximation. The results are shown in Figure 8.3.

Again, the results were computed using 16 simulation runs for each configuration. Taking into account the scale of the figure, we can safely say that

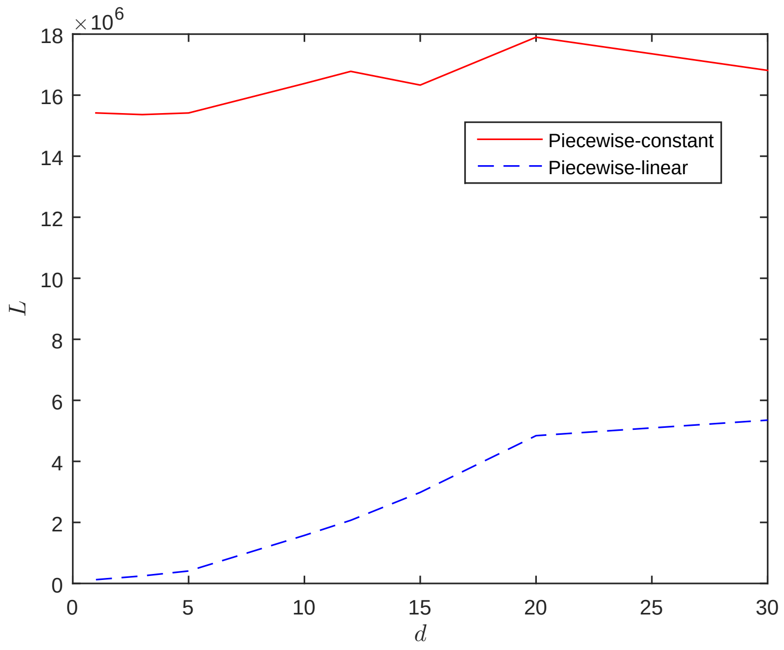


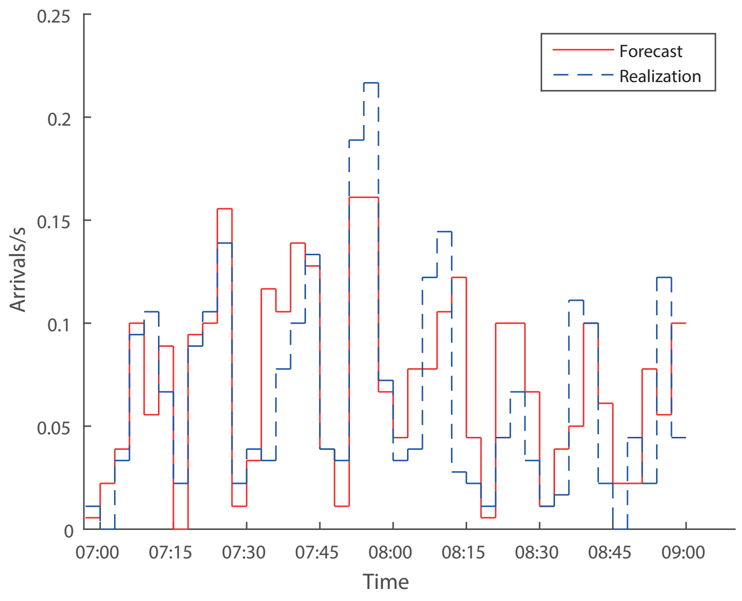
Figure 8.1: Comparison between piecewise-constant and piecewise-linear arrival rate approximations for several values of segment width $d$.

the choice for constant transfer time barely affects the KPI. However, we do see an unusual low KPI for a transfer time of zero, which unfortunately we cannot explain. For the non-zero transfer times, considering the scale of the figure, we see a rather constant curve. Therefore we come to the conclusion that, in general, the transfer time parameter does not have a significant impact on seating probability.

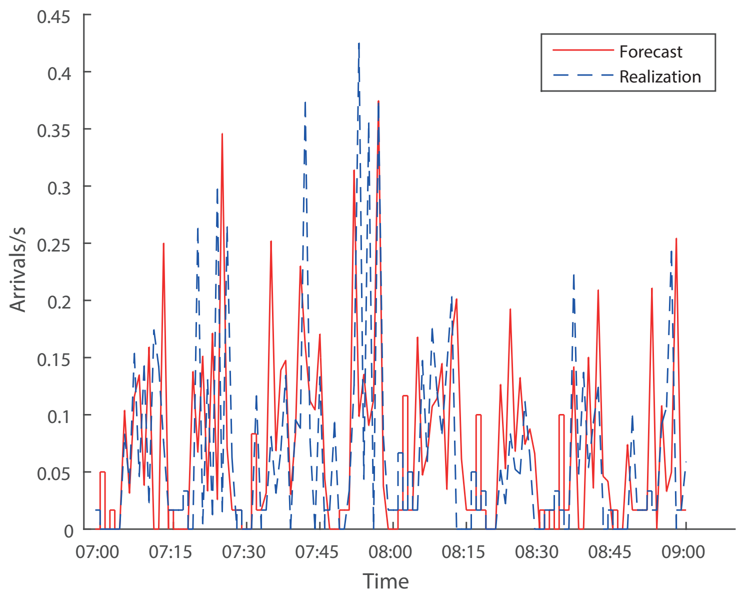
For the remainder of this section, we select a constant transfer time of 1 minute. The reason for this, is that it eliminates the impossible transfers with a zero minute transfer window, while also not being as restrictive as selecting a relatively high transfer time. Preliminary tests have shown that with a transfer time of 1 minute, we still get realistic transfers in most of the cases.

## 8.4 Simulation Output

Using the January 12 dataset, we predict the KPI for the entire main rail network for February 9, for (i) the entire day, (ii) the morning rush hour, (iii) evening rush hour, and (iv) both rush hour periods combined.



(a) $d = 3$ minutes



(b) $d = 1$ minute

Figure 8.2: Visual comparison between the predicted and realized rate, for the piecewise-constant ($d = 3$) and piecewise-linear ($d = 1$) approximations. Here, the arrival rate is approximated for passengers traveling from Rotterdam Alexander to Utrecht Centraal, from 07:00 to 09:00.

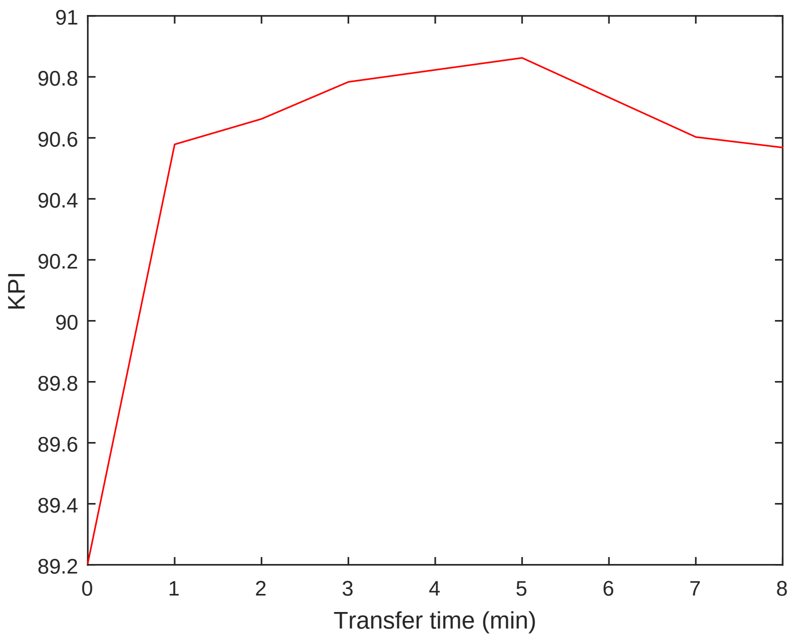


Figure 8.3: Plot of the KPI with varying constant transfer time.

Table 8.1: Simulation results for the February 9 dataset, using a predicted and realized arrival rate. The KPI is displayed in percentages, where the numbers between parentheses are the standard deviation.

| | KPI | |
|---|---|---|
| | Generated arrivals | Real arrivals |
| Entire day | **95.02** (0.046) | **94.28** (–) |
| Rush hour - Combined | **90.57** (0.094) | **89.11** (–) |
| Rush hour - Morning | **88.38** (0.130) | **86.17** (–) |
| Rush hour - Evening | **93.04** (0.150) | **92.44** (–) |

The results are reported in Table 8.1. The first column shows on what levels we aggregate the KPI. The second column shows the prediction for the KPI approximating the arrival rate with a piecewise-linear function with $d = 1$ minute. Finally, the third column shows the KPI obtained from using the real February 9 passenger arrivals. This allows us to assess the absolute quality of the piecewise-linear approximation, rather than the relative quality we investigated in Figure 8.1. Using the framework presented in this thesis, we predict a KPI of 90.57% for February 9, with the current planned timetable and rolling stock. However, using the real arrivals of that day, we obtain a KPI of 89.11%, meaning that our piecewise-linear approximation for the arrival rate causes an error of 1.46%. Breaking the rush hour into morning and evening, we see that main cause of that error is due to the morning rush (error for morning rush is 2.21%, whereas the error for evening rush is 0.60%).

The realized KPI value for February 9 was 93.11%, meaning that our simulation underestimates the KPI by almost 3%. This seems rather peculiar, as the realized KPI was calculated using the historical KPI with actual realization data, which includes train cancellation and delays, two negative factors which we do not include. However, the way that the realized KPI is calculated is different than how we do it. First and foremost, for the realized KPI the correction factors are not aggregated into one like we did with $k_{\text{cor}}$. Second, only the OD-pairs are included in the realized KPI that have at least 100 passengers in the last 100 days, where the passengers must travel on 20 different days (to exclude rare events). We on the other hand include every OD-pair with at least one passenger. Third, when a passenger has $x > 1$ travel options, the passenger is "split" over these $x$ options equally. Thus, the passenger counts for $1/x$ passenger for each of the $x$ alternatives. We, on contrary, pick exactly 1 of the alternatives, namely the lexicographically-first Pareto-optimum. By splitting the passenger, the load is smeared out over all alternatives, which relieves the burden of a "whole" passenger to already busy trips. On top of that,

We also see that the morning rush hour scores significantly lower than the evening rush hour. There are two factors contributing to the seating capacity: passenger volume, and transport capacity. Table 8.2 displays the results of our comparison using real passenger arrivals for February 9. We have chosen the average number of seats (including foldable seats) per trip as measure for seat capacity. The results show that the relatively bad performance during the morning rush is most likely caused by not allotting enough capacity. We see from Table 8.2 that for both rush hours, approximately an equal amount of seat capacity is allotted, the morning rush hour sees 28,283 (11.39%) more passenger arrivals than the evening rush.

Table 8.2: Comparison of morning and evening rush passenger volume and seat capacity. As a measure for seat capacity we have chosen the average number of seats per trip.

| | Passenger arrivals | Ave. seats/trip |
|---|---|---|
| Morning Rush | 276,533 | 475.52 |
| Evening Rush | 248,250 | 467.53 |

Table 8.3: List of ten worst train services, measured by KPI. Produced using a transfer time of 1 minute, using the piecewise-linear rate approximation with $d = 1$ minute, after 32 simulation runs.

| | Station | | | | |
|---|---|---|---|---|---|
| Train | Start | End | KPI | $c_{\text{plan}}$ | RH |
| 14869 | Amsterdam Centraal | Hoorn | **36.35** (2.58) | SGM3 | E* |
| 2825 | Rotterdam Centraal | Utrecht Centraal | **44.66** (2.44) | VIRM4 | M |
| 14867 | Amsterdam Centraal | Hoorn | **46.30** (3.12) | SGM5 | E |
| 14871 | Amsterdam Centraal | Hoorn | **48.35** (3.38) | SGM3 | E* |
| 2827 | Rotterdam Centraal | Utrecht Centraal | **49.09** (3.40) | VIRM6 | M |
| 4919 | Almere Oostvaarders | Utrecht Centraal | **49.58** (3.22) | SLT6 | M |
| 5663 | Utrecht Centraal | Zwolle | **50.61** (4.71) | DDZ4 | E |
| 5618 | Zwolle | Utrecht Centraal | **51.60** (3.85) | DDZ4 | M |
| 5020 | Breda | Den Haag Centraal | **52.28** (3.37) | SLT10 | M |
| 2218 | Dordrecht | Amsterdam Centraal | **52.74** (1.11) | VIRM8 | M |

* : The 14869 starts at 18:04 and the 14871 at 18:34, which is technically after the evening rush hour.

Apart from providing the user with just the predicted overall KPI, the simulation is also able to help provide insight. In order to help increase the KPI as efficiently as possible, the framework is able to display a list of worst performing train services. An example of such list is shown in Table 8.3 for 10 services, which was produced using 32 simulation runs, with a transfer time of 1 minute and segment width of $d = 1$ minute using the piecewise-linear rate approximation. The table shows the position in the worst ten list, the train number, origin and destination, predicted KPI, planned composition, and during which rush hour (RH) the service is executed. Out of the 10 services, 6 are executed during the morning rush hour, 2 during the evening rush hour, and 2 are executed just barely after the evening rush hour. Also, out the 10 services 7 are of the sprinter type (SGM, SLT, and DDZ[1]). Most notable is the poor performance of the three consecutive train services from the 14800 line.

Of course, it is also possible to aggregate the KPI on station level. To do this, we determine for each passenger arriving into the system at station $O$ which trip he boards. So, for each origin we obtain a set of boarding trips for which we can easily calculate the KPI using (2.2). Afterwards, we take the ten worst scoring origins and make a list, which is displayed in Table 8.4.

Table 8.4: Top ten worst performing origin stations (during rush hour), based on KPI.

| Pos. | Origin | KPI |
|---|---|---|
| 1 | Purmerend | **78.66** (0.97) |
| 2 | Sassenheim | **81.20** (0.64) |
| 3 | Diemen Zuid | **82.16** (0.51) |
| 4 | Santpoort Noord | **82.76** (0.75) |
| 5 | Soest | **83.01** (1.10) |
| 6 | Santpoort Zuid | **83.07** (0.86) |
| 7 | Waddinxveen Noord | **83.43** (0.85) |
| 8 | Den Haag Moerwijk | **83.94** (0.50) |
| 9 | Zaandam Kogerveld | **83.97** (1.04) |
| 10 | Soestdijk | **84.27** (0.64) |

First of all we note that all the stations in Table 8.4 are relatively small. In fact, all of them are only frequented by sprinter train services. This is not that surprising, as compositions used for sprinter train services (except DDZ

[1]The DDZ can be employed both as sprinter or intercity. However, in this case, they were employed as sprinters.

compositions) have a much lower ratio of seated to non-seated capacity than compositions used for intercity services.

## 8.5 Deviating From the Plans

Next, we investigate the impact of rolling stock mismatches. As the number of rolling stock combinations is too large to display on paper, we only provide a brief summary. We identified 54,686 possible composition changes in the planned timetable. Out of these 54,686 possible changes, 35,242 (64.44%) trains were executed as planned (until a possible next composition change). 9,595 (17.55%) were executed in a composition with lower capacity, and 9,849 (18.01%) with higher capacity. Note that although 64.44% trains running in the planned compositions seems low, do note that after the handover letter, the rolling stock schedule is still subject to some possible changes.

Now we investigate the sensitivity of the estimated probabilities. We introduce a parameter $\varphi$, indicating how often rolling stock mismatches occur. These altered mismatch probabilities are defined as

$$\begin{aligned} \tilde{\pi}_{c_{\text{plan}},c_{\text{plan}}} &= \pi_{c_{\text{plan}},c_{\text{plan}}}(1-\varphi)+\varphi, && c_{\text{plan}} \in \mathcal{C}, \\ \tilde{\pi}_{c_{\text{plan}},c_{\text{real}}} &= \pi_{c_{\text{plan}},c_{\text{real}}}(1-\varphi), && c_{\text{plan}}, c_{\text{real}} \in \mathcal{C}, \end{aligned}$$

with $\varphi \in \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$. In the case $\varphi = 0$, the probabilities are left unaltered and mismatches occur according to the originally estimated probabilities. On contrary, if $\varphi = 1$, all trains run in their planned compositions.

In addition to presenting the sensitivity of rolling stock mismatches to the KPI on national level (Table 8.5a), we also investigate the effect on a less aggregated level, namely for the ten worst performing train services of Table 8.3 (Table 8.5b).

Table 8.5a indicates that, in general, rolling stock mismatches tend to affect the KPI negatively, and significantly so. In our case, we estimate that mismatches cause the KPI to drop from 90.57% to 85.54%.

Looking at the results in Table 8.5b, we see that it is not always the case that mismatches worsen the KPI. In fact, we see the following happening. The more likely a mismatch, the more we see services with low-capacity compositions (e.g. 14869, 14867, and 14871) improve, and services with high-capacity compositions (e.g., 2218 and 5020) worsen. Take for example an SGM3. As this is the composition with the smallest seat capacity, if any other composition takes its place, the KPI must increase (keeping the passenger volume fixed). Similarly, the opposite occurs for a composition with a large seat capacity (like the VIRM8).

Table 8.5: Sensitivity of rolling stock mismatch probabilities on the KPI, aggregation based on (a) nation-wide, (b) worst ten train services.

(a)

| | Percentage as planned | | | | | |
|---|---|---|---|---|---|---|
| | 64.44 ($\varphi = 0.0$) | 71.55 ($\varphi = 0.2$) | 78.66 ($\varphi = 0.4$) | 85.78 ($\varphi = 0.6$) | 92.89 ($\varphi = 0.8$) | 100.00 ($\varphi = 1.0$) |
| KPI | **85.54** (0.546) | **86.58** (0.550) | **87.48** (0.501) | **88.67** (0.353) | **89.57** (0.286) | **90.57** (0.094) |

(b)

| | Percentage as planned | | | | | | |
|---|---|---|---|---|---|---|---|
| Train | 64.44 ($\varphi = 0.0$) | 71.55 ($\varphi = 0.2$) | 78.66 ($\varphi = 0.4$) | 85.78 ($\varphi = 0.6$) | 92.89 ($\varphi = 0.8$) | 100.00 ($\varphi = 1.0$) | $c_{\text{plan}}$ |
| 14869 | **41.90** (10.91) | **45.97** (13.97) | **41.17** (10.89) | **37.30** (5.10) | **37.76** (5.81) | **36.35** (2.58) | SGM3 |
| 2825 | **56.91** (22.19) | **53.47** (18.61) | **56.45** (22.08) | **50.58** (15.18) | **45.44** (7.19) | **44.66** (2.44) | VIRM4 |
| 14867 | **56.71** (14.29) | **50.63** (10.03) | **49.93** (13.20) | **47.50** (8.93) | **49.11** (7.43) | **46.30** (3.12) | SGM5 |
| 14871 | **58.58** (15.49) | **55.60** (12.87) | **54.23** (15.87) | **53.79** (12.88) | **49.80** (8.89) | **48.35** (3.38) | SGM3 |
| 2827 | **47.29** (13.67) | **42.87** (7.09) | **51.83** (15.66) | **49.50** (10.41) | **47.95** (3.80) | **49.09** (3.40) | VIRM6 |
| 4919 | **53.89** (15.43) | **54.97** (16.27) | **51.04** (11.88) | **51.02** (10.57) | **50.60** (5.18) | **49.58** (3.22) | SLT6 |
| 5663 | **53.81** (15.83) | **53.11** (16.23) | **50.15** (10.59) | **52.61** (10.24) | **52.55** (12.58) | **50.61** (4.71) | DDZ4 |
| 5618 | **55.11** (15.23) | **56.91** (17.74) | **58.41** (16.42) | **53.37** (9.60) | **50.76** (5.33) | **51.60** (3.85) | DDZ4 |
| 5020 | **42.24** (13.49) | **44.27** (14.74) | **45.50** (13.12) | **46.27** (13.01) | **49.11** (9.02) | **52.28** (3.37) | SLT10 |
| 2218 | **45.18** (14.19) | **46.01** (11.67) | **44.93** (12.65) | **48.68** (12.12) | **51.39** (4.14) | **52.74** (1.11) | VIRM8 |

Table 8.6: First 10 order statistics of the KPI aggregated on train services, for different values of $\varphi$

| | Percentage as planned | | | | | |
|---|---|---|---|---|---|---|
| | 64.44 | 71.55 | 78.66 | 85.78 | 92.89 | 100.00 |
| Pos. | ($\varphi = 0.0$) | ($\varphi = 0.2$) | ($\varphi = 0.4$) | ($\varphi = 0.6$) | ($\varphi = 0.8$) | ($\varphi = 1.0$) |
| 1 | **19.36** | **19.43** | **22.02** | **24.50** | **25.50** | **36.06** |
| | (3.74) | (4.42) | (5.44) | (4.90) | (6.13) | (1.82) |
| 2 | **22.34** | **23.06** | **26.11** | **29.13** | **31.35** | **43.08** |
| | (3.61) | (4.10) | (4.44) | (4.53) | (5.36) | (1.24) |
| 3 | **24.88** | **26.09** | **28.54** | **32.13** | **35.20** | **44.99** |
| | (3.31) | (3.91) | (3.81) | (4.11) | (4.59) | (1.29) |
| 4 | **27.00** | **28.16** | **30.82** | **34.36** | **37.37** | **46.71** |
| | (3.21) | (3.76) | (3.95) | (4.44) | (3.76) | (1.19) |
| 5 | **28.82** | **29.72** | **32.97** | **35.96** | **39.58** | **47.81** |
| | (2.72) | (3.81) | (2.75) | (3.57) | (4.19) | (1.23) |
| 6 | **30.33** | **31.88** | **34.41** | **37.68** | **41.31** | **49.07** |
| | (2.68) | (2.88) | (2.74) | (3.27) | (4.31) | (1.46) |
| 7 | **31.42** | **33.40** | **35.89** | **39.32** | **42.81** | **50.08** |
| | (2.58) | (2.80) | (2.74) | (3.84) | (4.05) | (1.45) |
| 8 | **32.63** | **35.08** | **37.09** | **40.85** | **44.10** | **50.94** |
| | (2.47) | (2.63) | (2.68) | (3.61) | (3.52) | (1.30) |
| 9 | **33.74** | **35.77** | **38.03** | **41.88** | **45.30** | **51.87** |
| | (2.51) | (2.64) | (2.69) | (3.49) | (3.05) | (1.17) |
| 10 | **34.57** | **37.04** | **38.80** | **42.88** | **46.29** | **52.67** |
| | (2.24) | (2.41) | (2.66) | (3.35) | (2.65) | (1.04) |

For Table 8.5b we have varied the likelihood of rolling stock mismatches, for a pre-selected set of train services. However, to provide another perspective to the impact of mismatches, we determine the KPI for all train services for some values of $\varphi$. Then, this list of KPIs is sorted, and the worst 10 are returned. This is different than Table 8.5b as we do not fix the train services. Except for some minor disturbances, we generally see that the more train services are executed with their planned compositions, the better the KPI. The only real outlier being the results for position 1, which after investigating turns out to be train 14869 for all values of $\varphi$. This discrepancy is caused by (as already discussed) the fact that if its composition varies, the KPI automatically improves.

Furthermore we have included the first ten order statistics for the train

Table 8.7: Impact of train cancellation on the KPI, for various values of $\psi$.

| | $\psi$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 0.0 | 0.002 | 0.005 | 0.01 | 0.015 | 0.02 | 0.025 | 0.03 |
| KPI | **90.57** (0.094) | **90.37** (0.186) | **90.19** (0.294) | **89.91** (0.350) | **89.67** (0.369) | **89.12** (0.445) | **88.89** (0.432) | **88.59** (0.690) |

services KPI, under varying $\varphi$. The results are found in Table 8.6. The statistics were calculated as follows. For each of the 32 iterations, sort the train service KPIs in increasing order. Then, for statistic $x_{(i)}$, compute the mean and standard deviation of the $i$-th element in each of the 32 lists. It can be seen from the table that the impact of rolling stock mismatches can be quite large. Looking at the standard deviation, we see that the more we go to the end (or tail) of the list, the larger the deviations in KPI.

Finally we consider the impact on the KPI of canceling train services. As typically about 1.5% of all services are canceled, we compare between the values $\psi = 0.0$, 0.002, 0.005, 0.01, 0.015, 0.02, 0.025, and 0.03. The results are displayed in Table 8.7. If no services are canceled, the KPI is around 90.57%, as we have already seen. However, increasing $\psi$ gradually decreases the KPI. At $\psi = 0.015$ we obtain a KPI of 89.67%, which does not seem to be that much of a decrease. However, Increasing $\psi$ also tends to increase the uncertainty in the KPI estimation, which is to be expected.

Under typical circumstances ($\varphi = 0.0$ and $\psi = 0.015$), we see that rolling stock mismatches lower the KPI by 5%, whereas train cancellations decrease the KPI by not even 1%. As a matter of fact, even in the case that $\psi = 0.03$ do we see that train cancellations have a smaller effect on the KPI than rolling stock mismatches do.

# Chapter 9

# Conclusion

In this thesis we have looked at developing a simulation framework capable of evaluating the new definition for the KPI "transport capacity during rush hour", and we have also derived some insights from it. The problem of evaluating the KPI is split into that of (i) modeling passenger arrivals, (ii) representing the timetable and rolling stock schedule, and (iii) determining passenger routes. Next, insight was derived by looking at how the KPI reacted to deviations from the planned timetable and rolling stock schedule. Here, we looked at the impact of rolling stock mismatches and train cancellations.

We modeled passenger arrivals by assuming a passenger group for each OD-pair. For each passenger group (i.e., OD-pair), we estimated a non-homogeneous Poisson process, where we assume the arrival rate function $\lambda(t)$ to be either piecewise-constant, or piecewise-linear. We compared the two estimation methods using a criterion we developed based on the difference in actual arrivals per trip, and those generated by approximating $\lambda(t)$. Comparing both piecewise-constant and piecewise-linear approximations, we see that the piecewise-linear approximation with segment width $d = 1$ minute clearly performs best. Apart from relative fit, we also check the absolute goodness of fit and prediction power by comparing the KPI obtained from using actual arrivals, and that obtained from assuming a piecewise-linear approximation for $\lambda(t)$. Our model overestimates the KPI by 1.46%.

By far the most popular way to represent a timetable in railway literature is by means of an event-activity network. In this thesis we looked at the variant allowing for constant transfer times, although this approach can also be extended to allow for non-constant transfer times. We have seen that the KPI is pretty robust to choices of transfer time in the sense that it seems to barely effect it at all. However, we do see a small difference in zero and non-zero transfer times.

As we were not able to obtain passenger route preference data, we unfortunately had to make an assumption regarding passenger routing. We made the assumption that passengers always travel using the earliest arriving path with a minimum number of transfers. Following this assumption is a Dijkstra-like algorithm able to find the lexicographically-first Pareto-optimum using travel time and number of transfers as criteria (in that order).

As already mentioned, we also looked at the impact to the KPI of rolling stock mismatches and train cancellations. We model rolling stock mismatches by looking at the planned and realized train compositions. For every moment in time that a train composition is either formed or changed, we assume that a mismatch may occur, meaning that the realized composition differs from planned. We estimate probabilities for each pair of planned and realized composition. According to our framework, the KPI with planned timetable and rolling stock schedule for February 9 is 90.57%. We estimate that with rolling stock mismatches the KPI becomes 85.54%, meaning these mismatches have a significant impact of 5.03%. Next, we assume that a train is canceled with probability $\psi$, independently of all other events. Typically, about 1–1.5% of all trains is canceled, resulting in a KPI of 89.67–89.91%.

In general, it seems that rolling stock mismatches cause a greater loss in KPI than train cancellations. Looking with more detail into the KPI, we found the following. The ten worst performing origin stations (under planned timetable and rolling stock schedule) are exclusively so-called "sprinter stations", meaning only sprinter trains frequent them. Also, 7 out of the 10 worst performing train services are sprinter services. Checking the seating capacities, we indeed see that train types designated for sprinter services (SGM, SLT) have a relatively smaller ratio of seated to non-seated capacity. Regarding rolling stock mismatches, we found that, although in general the KPI deteriorates the more mismatches occur, it all depends on the planned train composition. For example, we have seen that if a low-capacity composition is planned (like an SGM3), increasing the probability of a mismatch means the KPI for that train service will most like increase, as there are more compositions with larger capacity than with less. If a high-capacity composition is planned, the reverse effect occurs. Therefore, we advise NS to focus reducing rolling stock mismatches for high-capacity compositions, as for low-capacity compositions mismatches are actually beneficial to the KPI.

Unfortunately, due to time constraints we were unable to model train delays for the simulation framework. This is unfortunate, as delays occur way more frequent than train cancellations, and thus may impact the KPI significantly.

# Bibliography

F. Alizadeh, J. Eckstein, N. Noyan, and G. Rudolf. Arrival rate approximation by nonnegative cubic splines. *Operations Research*, 56(1):140–156, 2008.

R. Baars. Massaclaim NS in de maak om tekort zitplaatsen. *AD*, 2015. URL `http://www.ad.nl/binnenland/massaclaim-tegen-ns-in-de-maak-om-tekort-zitplaatsen~a927d4aa/`. Accessed: 2016-09-06.

M. E. Ben-Akiva. *Structure of passenger travel demand models*. PhD thesis, Massachusetts Institute of Technology, 1974.

H. Y. Benson, D. F. Shanno, and R. J. Vanderbei. Interior-point methods for nonconvex nonlinear programming: Filter methods and merit functions. *Comput. Optim Appl.*, 23(2):257–272, 2002.

K. Bos. 'ik betaal voor een zitplaats, die wil ik'. *NRC.nl*, 2015. URL `http://www.nrc.nl/next/2015/12/01/ik-betaal-voor-een-zitplaats-die-wil-ik-1564917`. Accessed: 2016-09-06.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambride University Press, 2004.

L. Brown, N. Gans, A. Mandelbaum, A. Sakov, H. Shen, S. Zeltyn, and L. Zhao. Statistical analysis of a telephone call center: A queueing-science perspective. *Journal of the American Statistical Association*, 100(469): 36–50, 2005.

R. H. Byrd, J. Nocedal, and R. A. Waltz. Knitro: An integrated package for nonlinear optimization. In G. Pillo and M. Roma, editors, *Large-Scale Nonlinear Optimization*. Springer-Verlag, 2006.

L. Cadarso and Á. Marín. Robust rolling stock in rapid transit networks. *Computers & Operations Research*, 38(8):1131–1142, 2011.

A. Caprara, L. Kroon, M. Monaci, M. Peeters, and P. Toth. Passenger railway optimization. In C. Barnhart and G. Laporte, editors, *Handbooks in OR & MS*, volume 14, pages 129–187. Elsevier, 2007.

E. Cascetta, A. Nuzzolo, F. Russo, and A. Vitetta. A modified logit route choice model overcoming path overlapping problems: specification and some calibration results for interurban networks. In *Proceedings of the 13th International Symposium on Transportation and Traffic Theory*, pages 697–711, 1996.

M. Chen, R. Chowdhury, V. Ramachandran, D. Roche, and L. Tong. Priority queues and dijkstra's algorithm. Technical Report TR-07-54, UTCS, 2007.

T. Dollevoet, D. Huisman, M. Schmidt, and A. Schöbel. Delay management with rerouting of passengers. *Transportation Science*, 46(1):74–89, 2012.

M. Duursma. In 2016 wordt het weer drukker in de spitstreinen. *NRC.nl*, 2015. URL `http://www.nrc.nl/handelsblad/2015/12/01/in-2016-wordt-het-weer-drukker-in-de-spitstreinen-1565267`. Accessed: 2016-09-06.

J. T. Haahr, R. M. Lusby, J. Larsen, and D. Pisinger. A branch-and-price framework for railway rolling stock rescheduling during disruptions. Technical report, DTU Management Engineering, 2014.

H. van Houwelingen and T. Voermans. Recordaantal klachten over te volle treinen NS. *AD*, 2015. URL `http://www.ad.nl/binnenland/recordaantal-klachten-over-te-volle-treinen-ns~a6e07253/`. Accessed: 2016-09-06.

E. van der Hurk, L. Kroon, G. Maróti, and P. Vervest. Deduction of passengers' route choices from smart card data. *Intelligent Transportation Systems, IEEE Transactions on*, 16(1):430–440, 2015.

L. Kroon, G. Maróti, and L. K. Nielsen. Rescheduling of railway rolling stock with dynamic passenger flows. *Transportation Science*, 49(2):165–184, 2014.

P. A. W. Lewis and G. S. Shedler. Simulation of nonhomogeneous poisson processes by thinning. *Naval Research Logistics Quarterly*, 26(3):403–413, 1979.

W. A. Massey, G. A. Parker, and W. Whitt. Estimating the parameters of a nonhomogeneous poisson process with linear rate. *Telecommunication Systems*, 5:361–388, 1996.

K. Nachtigall. Periodic network optimization and fixed interval timetables. *Deutsches Zentrum für Luft–und Raumfahrt, Institut für Flugführung, Braunschweig*, 1998.

L. K. Nielsen. *Rolling Stock Rescheduling in Passenger Railways*. PhD thesis, Erasmus University Rotterdam, 2011.

N. Oppenheim. A combined, equilibrium model of urban personal travel and goods movements. *Transportation Science*, 27(2):161–173, 1993.

A. Orda and R. Rom. Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length. *Journal of the ACM*, 37(3):607–625, 1990.

A. Orda and R. Rom. Minimum weight paths in time-dependent networks. *Networks*, 21(3):295–319, 1991.

S. Raveau, J. C. Muñoz, and L. de Grange. A topological route choice model for metro. *Transportation Research Part A: Policy and Practice*, 45(2): 138–147, 2011.

A. Schöbel and S. Scholl. Line planning with minimal traveling time. In *OASIcs-OpenAccess Series in Informatics*, volume 2. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2006.

F. Schulz. *Timetable Information and Shortest Paths*. PhD thesis, Universität Fridericiana zu Karlsruhe, 2005.

A. Wächter and L. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program.*, 106(25), 2006.

Y. Wang, T. Tang, B. Ning, T. J. van den Boom, and B. de Schutter. Passenger-demands-oriented train scheduling for an urban rail transit network. *Transportation Research Part C*, 60:1–23, 2015.