UNIVERSITY OF TWENTE
Applied Mathematics: Stochastic Operations Research

Computations of the Demand for Ground Resources at KLM

Jord Rood
s1750372

February 2018

Abstract

The demand for ground resources at KLM Royal Dutch Airlines is determined by extensive resource planning tools that do not take any form of flight delay into account. This thesis proposes machine learning models that provide a quick estimation of the demand for ground resources during the flight network scheduling phase. These models show that this demand, as planned by current extensive calculations, can be estimated with a sufficient error in a significant smaller amount of time.

Furthermore, methods to include flight delay in ground resource planning are explored. The described machine learning models can be used in combination with simulation to give an indication of the resource demand on the day of operation, where flight and process delays can occur. Another approach to determine the expected demand for ground resources on the day of operation in the form of a time-inhomogeneous continuous time Markov chain model is proposed. A method of iterative state space exploration and dynamic truncation is described in order to evaluate such a model.

Preface

De luchtvaart is een van de voornaamste redenen waarom ik Operations Research ben gaan studeren. De processen op en rond een luchthaven hebben mij altijd al gefascineerd. Na een periode van twee jaar als Duaal stagiair bij KLM Decision Support, heb ik gelukkig de kans gekregen om hier ook mijn afstudeeronderzoek uit te voeren. Bij aanvang van deze opdracht was het even zoeken naar de juiste richting, maar na een lang proces met goede begeleiding mag ik tevreden zijn met het resultaat.
Deze Master thesis zou nooit tot stand gekomen zonder een groot aantal mensen die mij zowel inhoudelijk als persoonlijk begeleid en gesteund hebben. Ten eerste wil ik graag mijn directe begeleiders bedanken: Nico van Dijk (UTwente) en Matthieu Bolt (KLM), voor hun steun en advies.
Ook bedank ik de leidinggevenden van de betrokken afdelingen: Michel Mulder (Network Capacity Planning), Feodor Dekker (Ground Services - Tactical Planning), Anne Jan Beeks (Decision Support - Operations Research) en Bernard Vroom (Decision Support - Operations Research) die het mij mogelijk hebben gemaakt om aan de nodige data, bedrijfsinformatie en resources te komen om deze thesis te kunnen voltooien.
Tot slot wil ik graag al mijn collega's bij Decision Support - Operations Research bedanken voor de brainstormsessies en adviezen maar vooral ook voor de afleiding, gezelligheid en potjes tafeltennis.

Contents

Chapter 1

Introduction

KLM Royal Dutch Airlines is a large airline that serves 30 million annual passengers operating from its home base Amsterdam Airport Schiphol. The Network department of this airline aims to design a commercially interesting flight schedule that does not violate any operational constraints. For example, KLM needs to own take-off and landing rights - so called slots - in order to be able to operate the planned schedule. Another aspect that reduces flexibility of flight schedule planning is the airline's resource capacity constraints. The use of resources like fleet, manpower and ground equipment has to be taken into consideration while designing the flight schedule. To make sure that a schedule is operationally feasible, each department that is responsible for some of the resources will apply an operational check at some point in the planning process.
These operational checks often require extensive calculations. Each department is responsible for its own performance and it want to be really sure it can handle the flight schedule before they give it a green light. The results of these operational checks can come as a surprise for the Network department and occasionally bring them back to the drawing board. This problem especially arises with the demand for ground resources, that is calculated by the Ground Services - Tactical Planning (GS-TP) department.

1.1 Goal of This Thesis

The aim of this thesis is to provide KLM's Network department with more insight in the resource demand for ground equipment that is determined by the Ground Services - Tactical Planning (GS-TP) department. This includes an estimation of the demand for the most critical resources - i.e. what can cause operational infeasibility? - and an advise on how to solve problems regarding ground equipment resource problems - i.e. how to alter a flight schedule in order to turn an operationally infeasible schedule into a schedule that is likely to be accepted by GS-TP.
Currently, GS-TP determines their resource demand based on a planned schedule where no flight delays are taken into account. Therefore, an additional goal of this thesis is to provide a method to give an indication of the resource demand on the day of operation.
If these goals are achieved, KLM's Network department will have a method to quickly estimate

operational feasibility in terms of ground resources. This insight in operational constraints allows them to be able to make better scheduling decisions regarding the commercial objectives which improves the usage of the little degrees of freedom in flight network scheduling (figure 1.1). Furthermore, any insight in the resource demand on the day of operation will allow GS-TP to validate their planning process. In addition, this information can be used to substantiate the discussion between Network and GS-TP on operational feasibility.

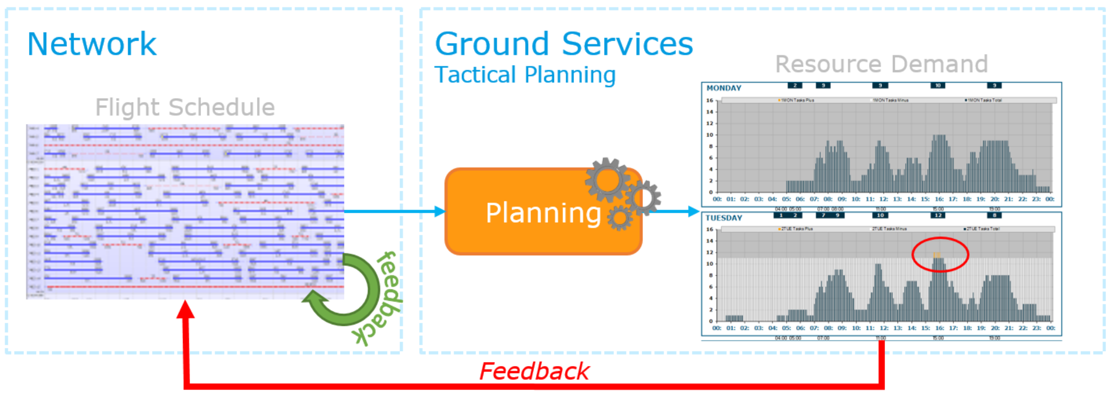


FIGURE 1.1: A small feedback cycle allows the Network department to verify operational feasibility before they send a schedule to GS-TP.

1.2 Research Questions

The goals that are aimed to be achieved in this thesis can be described by the following research questions.

1. What is the relation between a flight schedule and the resource demand for the most critical ground resources as calculated by Ground Services - Tactical Planning and is it possible to quickly verify whether a flight schedule will be accepted after an operational check?
2. Given that a schedule will not be accepted by Ground Services - Tactical Planning, how can the Network capacity planners be advised to alter the schedule such that it increases the probability of acceptation?
3. Given a flight schedule, what can be said about the demand for ground resources on the day of operation?

1.3 Methods

The first resource question aims to find a relation between a flight schedule and the demand for ground resources. Machine learning models have shown to be able to find relations between input and output and can be used to quickly predict the outcome of an input instance the model has not seen before [1]. This thesis uses Gradient Boosting in its powerful implementation XGBoost [2] to find this relation between a flight schedule and the resource demand. These models can be used to answer research questions 1 and 2.

In order to answer research question 3, an existing simulation tool named OPiuM is used to simulate delays in both flights and ground processes. The output of these simulations is then used as input for the machine learning models to estimate the demand necessary to operate such simulated schedules.
Another approach that is proposed in order to answer research question 3 is to model the movement of aircraft on Schiphol as a continuous time Markov chain. Because the number of arrivals and departures on Schiphol fluctuates over a day, this Markov chain will contain time-inhomogeneous transition rates. This requires solution methods as discretization and uniformization as summarized by Van Dijk, Van Bummelen and Boucherie [3]. Because of the large state description of a Schiphol CTMC model, state truncation is necessary in order to evaluate its characteristics. For regular state truncation of time-homogeneous Markov chains it has been proven that the error that is caused by the truncation can be bounded [4]. However, truncating this large time-inhomogeneous model requires a more dynamical approach of truncation that is based on the work of Andreychenko [5].

1.4 Outline

This thesis report is structured as follows. Chapter 2 contains a description of the KLM and the Network and GS-TP departments. Then, chapter 3 provides a problem description containing the research questions and the scope of this thesis. From there this thesis is divided in three parts.
Part I considers machine learning models in order to answer research questions 1 and 2 and starts with chapter 4 discussing the used methods. Its results are presented in chapter 4 and the discussion, conclusions and recommendations are combined in chapter 6.
Part II discusses a simulated approach in combination with the machine learning models - as described in *Part I* - in order to answer research question 3. First, simulation tool OPiuM is described as well as how machine learning can be applied on its output (chapter 7). Then, the results are presented in chapter 8, followed by chapter 9 that contains the discussion, conclusions and recommendations based on these results.
Part III contains the description of a time-inhomogeneous continuous time Markov chain model of the movements of aircraft on Schiphol airport, including a solution approach based on discretization using iterative state exploration and dynamic truncation. This part starts with an introduction (chapter 10). An overview of the background of continuous-time Markov chains is given in chapter 11 followed by chapter 12 describing the Schiphol CTMC model. Algorithms to solve such a large time-inhomogeneous CTMC are proposed in chapter 13. The results of this approach are presented in chapter 14. Chapter 15 contains the discussion, conclusions and recommendations of these results.
After the main content of this thesis the Appendices are included. These are followed by the Bibliography, a Glossary, a list of Abbreviations and a Management Summary.

1.5 Reading Guide

This thesis report describes multiple approaches in order to propose solutions to the problems regarding ground resource demand that KLM's Network department addresses. It contains detailed descriptions of the underlying processes as well as technical sections to substantiate the applied models. The thesis is divided in three parts that can be read independently. It is advised to read chapter 2 and chapter 3 before diving into one of these three part. Since *Part II* uses results from *Part I* it is strongly recommended to first read at least the first two sections of chapter 4 to have a sufficient understanding of these results.
Since this report can be read with multiple intentions, three different readers are described. For these readers the following reading guides can be used:

- **Management Reader**: For KLM managers, with sufficient understanding of the underlying processes, it is recommended to first read the Management Summary in the back of the report. After that, they can consult the results, discussion, conclusion and recommendation sections of each part this thesis for a more detailed description of the achievements of this study.
- **SOR Reader**: For technical readers with a sufficient understanding of Stochastic Operations Research, *Part III* will be the most interesting part. In order to understand the underlying problem it is recommended to first read chapter 2 and chapter 3.
- **Aviation Decision Support Reader**: For all readers working in the airline industry as Decision Support Consultant, Business Analyst or Data Scientist, *Part I* and *Part II* will be of interest. Feel free to read *Part III* as well if Stochastic Operations Research is one of you points of interest.

For all readers it is recommended to consult the Glossary and list of Abbreviations in the back of this report whenever an unknown term or abbreviation is used.

Chapter 2

Description of KLM and Key Departments

The problems that are aimed to be solved in this thesis mainly arise from the communication between two departments within KLM:

- Network department
- Ground Services - Tactical Planning (GS-TP) department

Both have their own responsibilities and objectives that they aim to achieve during their collaboration. This chapter first describes KLM as an airline in section 2.1. An overview of the two departments, Network and GS-TP, are given in sections 2.2 and 2.3 respectively. Then the process of the Operational Planning Cycle (OPC), the most important collaboration between these two departments, is explained in section 2.4.

2.1 KLM Royal Dutch Airways

KLM Royal Dutch Airlines (KLM), founded in October 1919, is the oldest airline worldwide that is still operating under its original name. The company merged with Air France (AF) into the Air France-KLM Group (AF-KL) in 2004. Together with Delta and Alitalia they form a North-Atlantic joint venture. The airline group also plays an important role in the SkyTeam Airline Alliance, one of world's largest airline collaborations. The core of the KLM group, KLM and KLM Cityhopper (KLC), has a combined fleet of around 200 aircraft, with which it connects 170 destinations in 74 countries worldwide. It employs around 32 thousand people in order to transport its 30 million passengers and 635.000 tons of cargo annually. As a result, it generated around 10 billion euros of revenue in 2016. These figures are available on KLM's website [6].
KLM can be considered as a typical hub and spoke carrier (figure 2.1), which means that it provides almost all their flights to and from their home base Amsterdam Schiphol Airport (AMS).

Their operation is mainly focused on providing connections between a variety of airports around the world with a transfer on Schiphol. Around 75% of KLM customers is a transfer passenger with a layover at their main hub. In order to achieve this many international connections, the airline works with a so called bank system.

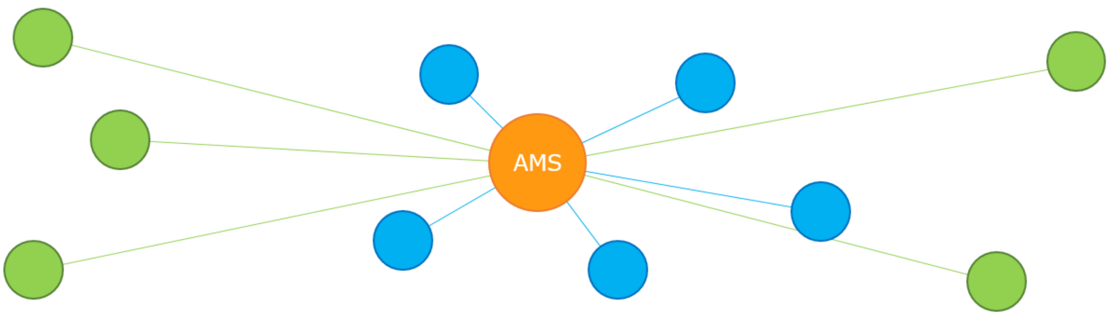


Figure 2.1: A typical hub and spoke system. KLM provides connections (spokes) to its main hub station Schiphol (AMS). In this way it can connect multiple European (blue) and intercontinental (green) destinations.

The “bank” system - also called “wave” system - (figure 2.2) defines multiple moments of peaks in arrivals and departures at an airport. In the remainder of this report the term “bank” is used to refer to this system. The idea is that a cluster of flights will arrive before, and depart after a certain moment of the day. This includes both short haul flights (European or EUR) and long haul flights (Intercontinental or ICA). This system allows for the connection of multiple destinations for transfer passengers.
For example, let there be 40 EUR destinations and 15 ICA destinations that all have both an arrival and departure on the 2nd bank of the day. Now in total there are $(40+15)\times(40+15-1) =$ 2970 unique travel paths between all connected airports using AMS as connecting hub. Adding the $40+15+40+15 = 130$ direct flights to and from AMS, there are in total $4160+130 = 3100$ possible combinations of origin and destination created with only 130 single flights.

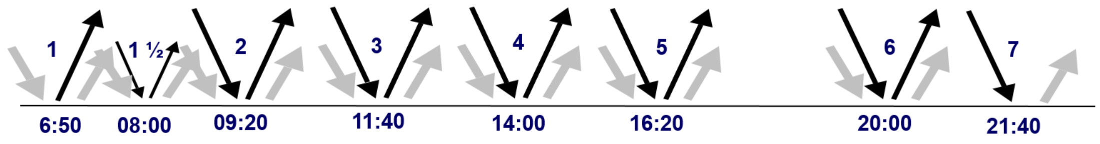


Figure 2.2: The Schiphol bank system: a single day has 8 banks, which define in- and outbound peaks, to optimize network connectivity. The arrows indicate in- and outbound of short haul (long thin black) and long haul (short bold grey).

This bank system is necessary for commercial success of the hub and spoke structure of KLM’s Network. It also contributes to the utilization and flexibility of their air resources like aircraft and cockpit- and cabin crew: because multiple flights arrive and depart at the same time, crew can be distributed over the flights in many ways. The downside is that the ground activities on Schiphol experience high demand peaks during these banks.
The great connectivity on Schiphol is one of the main reasons that this airport is expanding rapidly in terms of aircraft movements [7]. This results in a debate about whether or not the airport can grow - that has been widely discussed in the media [8–10] - and a scarcity of aircraft landing rights, so called slots [11]. This can result in fierce political debates [12]. Because of this

situation, KLM's Network department loses degrees of freedom in order to fulfill their objective: to design a commercially interesting flight network that will be operationally profitable.

2.2 Network Department

KLM's network department forms the first implementation of the airline's strategy (figure 2.3). It determines where to fly to, when to fly, how often to fly, and what the capacity of each flight should be - i.e. what aircraft and configuration of seats to use. Because their work heavily influences KLM's core business, and therefore has a direct influence on the airline's performance, the Network department maintains good contacts with the executive committee (see organizational chart in appendix A). During Network's masterclasses they describe their mission as *"to design the most attractive KLM (and partners) network for all stakeholders, based on the optimal match between commercial wishes and operational capabilities in order to develop a sustainable and profitable position within the AF-KL Group"*.

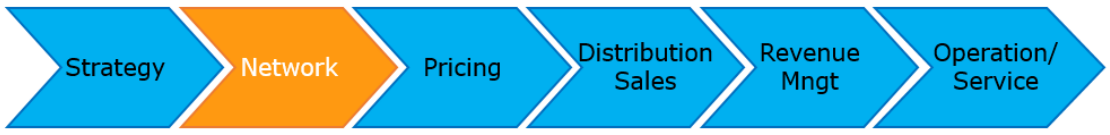


FIGURE 2.3: A typical airline value adding chain.

The department can be divided in three subdivisions: Network Planning, Network Schedule and Capacity Planning and Schedule Distribution. The network planners are driven commercially and are organized in geographical regions. They analyze the market and estimate whether flying to a destination in a certain frequency is profitable. Network Scheduling and Capacity Planning is operationally driven. They have to make sure that a schedule is operationally feasible, i.e. the flight network can be operated with the currently available resources. Schedule Distribution is responsible for distributing the schedule to the rest of the company. This last subdivision also manages code shares, which are collaborations between airlines where they can sell tickets to each other's flights in order to expand their network.

2.2.1 Flight Scheduling Tool

The Network department uses a tool to support its network planning and scheduling processes. Flights are represented in a Gannt chart as blue horizontal bars on horizontal lines. These lines, called fleet lines, represent the time line of a single aircraft. By this representation of the schedule it immediately becomes clear how the schedule might be operated by the current fleet.
This tool also shows aircraft reservations that are visualized by red bars. These aircraft reservations are necessary to have some slack in the flight schedule. The reason for this slack can be an obligatory periodical maintenance check. Another reason is to have an aircraft as stand-by to be able to use it when other aircraft are disrupted on the day of operation. These reservations are indicated with codes. For example, `#RS` is used to indicate a stand-by reservation, `##A` denotes

built-in slack for the so called A-check (a maintenance check that has to be carried out every 400-600 flight hours) and `#TO` marks reservations for small technical maintenance.

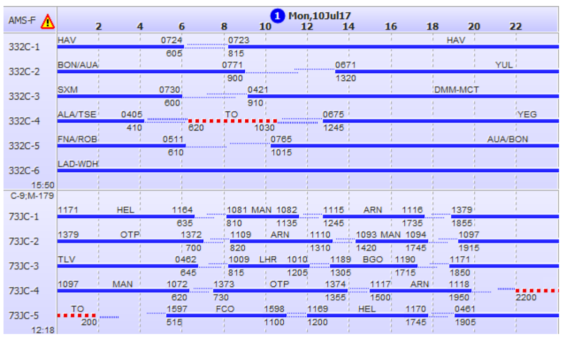


FIGURE 2.4: Flight schedule Gannt chart

Flights can be distributed over fleet lines according to a First In First Out (FIFO) or Last In First Out (LIFO) algorithm (figure 2.5). With the FIFO implementation, the current fleet line with the earliest arrival (First In) should be the fleet line with the earliest departure (First Out). With the LIFO implementation, however, the current fleet line with the latest arrival (Last In) should be the fleet line with the earliest departure (First Out). LIFO is used by network capacity planners because it shows big gaps where another flight might fit in, for example between flight KBP-AMS and flight AMS-MAD in figure 2.5. For operations and resource planning the FIFO implementation is used because it spreads slack over the fleet lines. A combination is possible where the FIFO algorithm is used for short turns and a switch to LIFO is made when a turnaround is longer than x minutes. In this case one can use the advantages of both methods.

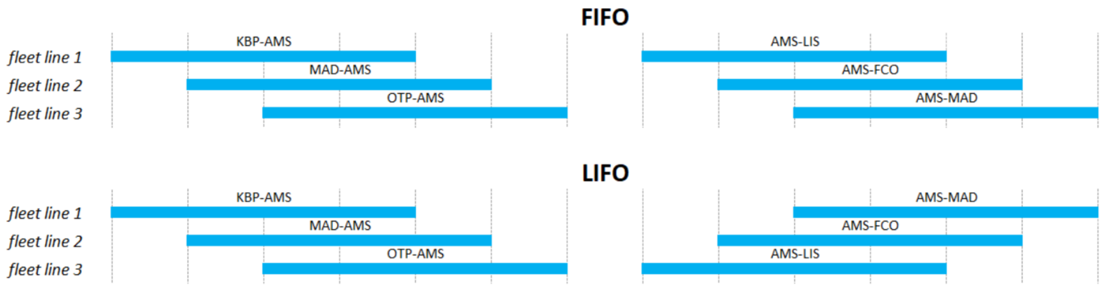


FIGURE 2.5: Example of six flights that are scheduled FIFO and LIFO. Above each flight its origin and destination are shown in three letter airport codes.

2.2.2 Gate Planning Tool

The Network department uses a gate planning tool to indicate the gate demand of a schedule. This tool assigns VOPs (vliegtuigopstelplaatsen) to incoming and outgoing flights according to some planning rules. VOPs are locations on an airport where an aircraft can be placed such as a gate, a buffer or a parking position. These VOPs are grouped by category that indicates the size of the biggest aircraft that can use the VOP. This category ranges from Cat. 1 - small private jets - to Cat. 9 - Airbus A380 jumbo jet.
When there is much time between an arrival and departure of an aircraft, it might be necessary to be towed to a parking position to create space for other aircraft that have to be handled at a gate. Therefore, this gate planning tool generates towing tasks when there is not enough space available to allocate all aircraft. It contains an optimizer to reduce the number of tows and it can swap flights in order to achieve a better gate allocation.

2.2.3 BTS Workload Tool

KLM's Network department is able to visualize the workload of BTS. BTS is the subdivision of Ground Services that is responsible for loading and unloading baggage and cargo. These BTS-graphs are based on the workload business rules of GS. Tasks are assigned to arriving and departing aircraft where the connection of flights are not taken into account. Although this calculation of workload is not always representative, it can still be used to compare different schedules and thus estimate the effect of a schedule change on the workload of BTS.

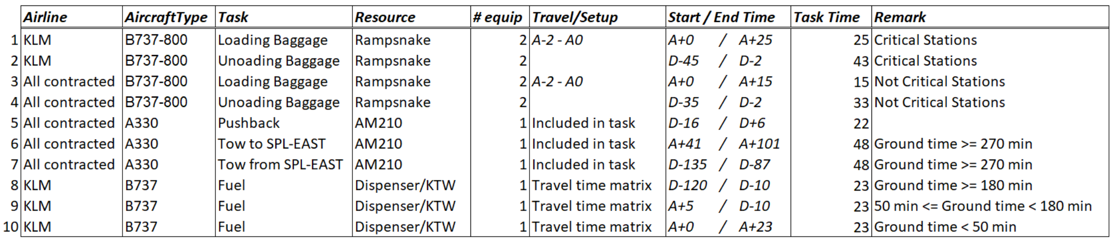
2.3 Ground Services - Tactical Planning Department

Ground Services (GS) is the department that is responsible for the ground handling of aircraft on KLM's main hub Schiphol. It consist of several sub-departments like Passenger Services, Apron Services and Baggage Services (see appendix A for organizational chart). Ground Services - Tactical Planning (GS-TP) is the sub-department that supports the planning process of GS by maintaining a rules catalogue of planning parameters, monitoring GS processes and providing a planning for ground resources like baggage operators, towing trucks and fuel trucks. An important task of GS-TP is to evaluate a flight schedule and check its operational feasibility in terms of ground resources.

GS-TP is organized in groups of planners that are responsible for some particular resources or processes like towing equipment, fuel equipment and baggage handling. Each planner maintains contact with the operators responsible for these processes, creates planning rules and updates these rules according to process changes. An inside IT-group supports GS-TP with tooling and data management.

2.3.1 Planning Rules (PUGs)

Planning rules are used as an abstraction of ground processes such that resources can be assigned to certain tasks and a planning can be realized. These planning rules can be referred to as the PUGs, which is an acronym for the dutch word "planningsuitgangspunten". Because of the differences in processes and differences in planners, PUGs occur in multiple varieties.



	Airline	AircraftType	Task	Resource	# equip	Travel/Setup	Start / End Time	Task Time	Remark
1	KLM	B737-800	Loading Baggage	Rampsnake	2	A-2 - A0	A+0 / A+25	25	Critical Stations
2	KLM	B737-800	Unoading Baggage	Rampsnake	2		D-45 / D-2	43	Critical Stations
3	All contracted	B737-800	Loading Baggage	Rampsnake	2	A-2 - A0	A+0 / A+15	15	Not Critical Stations
4	All contracted	B737-800	Unoading Baggage	Rampsnake	2		D-35 / D-2	33	Not Critical Stations
5	All contracted	A330	Pushback	AM210	1	Included in task	D-16 / D+6	22	
6	All contracted	A330	Tow to SPL-EAST	AM210	1	Included in task	A+41 / A+101	48	Ground time >= 270 min
7	All contracted	A330	Tow from SPL-EAST	AM210	1	Included in task	D-135 / D-87	48	Ground time >= 270 min
8	KLM	B737	Fuel	Dispenser/KTW	1	Travel time matrix	D-120 / D-10	23	Ground time >= 180 min
9	KLM	B737	Fuel	Dispenser/KTW	1	Travel time matrix	A+5 / D-10	23	50 min <= Ground time < 180 min
10	KLM	B737	Fuel	Dispenser/KTW	1	Travel time matrix	A+0 / A+23	23	Ground time < 50 min

Table 2.1: Example of some planning rules (planningsuitgangspunten or PUGs).

Table 2.1 shows some examples of PUGs for ground resources where most of the variants are included. The list below describes the characterizations of PUGs.

1. PUGs apply to some specific *airlines* and *aircraft types* or to groups of airlines and groups of aircraft types (first two columns). PUGS 1-4 show that this allows, for example, to apply different rules on the aircraft of different airlines.
2. PUGs define the *task* that has to be completed and the *resource* as well as the number of resources that are used to fulfill this task. Sometimes the resource that is used depends on the gate allocation (PUGs 8-10). A dispenser can only be used at gates that are connected to the Schiphol kerosene pipeline network.
3. *Start* and *End Time* are given in minutes relative to an aircraft's arrival (A) or departure (D). For example, PUG 8 has to be scheduled between 5 minutes after arrival (A+5) and 10 minutes before departure (D-10). It is possible that a task begins before an arrival or ends after a departure.
4. *Task Time* defines the time that is necessary to complete the task. For some tasks this *Task Time* is shorter than the difference between start and end time (PUGs 6, 8, 9). This means that these tasks can be shifted and scheduled somewhere such that it starts after their start-time and ends before their end-time.
5. *Travel Time* is contained in PUGs in multiple ways. Some PUGs define fixed travel times (PUGs 1 and 3). Others have the travel times included in their task times (PUGs 5-7). Others use a travel time matrix to determine the travel time of a resource from one task to another (PUGs 8-10).
6. Some PUGs require some *Remarks*, for example that the PUG only applies to aircraft that have an origin or destination for which they know that handling the baggage is a critical process (PUGs 1 and 2). Another example that the PUG only holds if the ground time - time between arrival and departure - is in a certain interval (PUGs 6-10).

2.3.2 GS-TP Tooling

In order to schedule ground resources given a flight schedule, GS-TP makes use of externally developed tooling: Gateplanner and PlanControl.

Gateplanner

Gateplanner is a tool that assigns aircraft to VOPs ("vliegtuigopstelplaatsen"), just like the gate planning tool of Network (section 2.2.2). It does not contain any optimization nor alters the schedule in order to achieve a good fit. It just takes the schedule as it is and assigns gates according to some planning rules. When a schedule does not fit it creates so called overflow gates. These gates do not physically exist but are used such that the tool can terminate its gate planning process. After a run, GS-TP planners alter the gate allocation such that there are no aircraft on overflow gates used anymore. Sometimes they conclude that, despite the usage of overflow gates, they will most likely be able to operate the schedule due to stochastic events in operations.
It is important to note that this gate planner tool only plans gates for a planned schedule. It tries to account for flight delays by adding some separation time. GS-TP uses a separation time

of 20 minutes which means that the allocation of two aircraft on the same gate should be at least separated by a 20 minute time interval.

Plancontrol

PlanControl is the tool that schedules ground resources, which uses a flight schedule and Gateplanner's gate allocation as its input. It generates tasks according to the PUGs and assigns them to resources with the objective to use as little resources as possible. The result is the demand for resources that are necessary to fulfill all tasks.
Note that this demand for resources is calculated for a fixed planned schedule. This means that no flight delays are taken into account.

2.3.3 Ground Resources Planning Process

Figure 2.6 shows the process of resource planning from Freeze [1] - i.e. flight schedule - to resource demand. The gate planning, pug assignment and task planning are done by the tools described in the previous section. The final result, the demand for all ground resources, is presented in graphs that contain the highest number of demand for these resources per 15 minutes.

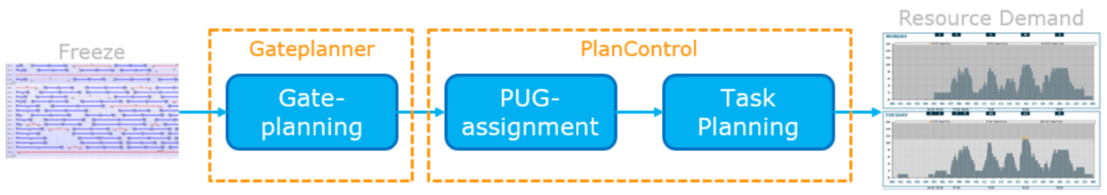


Figure 2.6: Process of determining resource planning for a given flight schedule.

This process is repeated weekly with a horizon of 4 weeks for production planning purposes. Every 4 weeks a Base Rolling Planning (BRP) is performed that looks 1 to 3 months ahead.
This resource planning process is a typical Capacity Requirements Planning (CRP) [13]. In its essence it is an administrative job of which the complexity lies in the large number of plannings rules where no complex models are involved.

2.4 Operational Planning Cycle

The Operational Planning Cycle (OPC) is used to verify the feasibility, performance and efficiency of a flight schedule. Multiple parties that are responsible for some KLM resources, like GS-TP, check the schedule that is designed by KLM Network department by calculating their expected demand. After verifying a schedule, these departments are allowed not to accept a flight schedule when they can proof it is not possible or not realistic to operate the schedule as it is.
Because of the High Performance Organization program, this process is now repeated every 4

[1]The term Freeze comes from the fact that the process of flight schedule planning is "frozen" in order to evaluate its expected performance and resource demand.

weeks to allow KLM to quickly react on market changes. Although the term "Operational Planning Cycle" is not used anymore, it is preferred in this report since it perfectly describes the process.

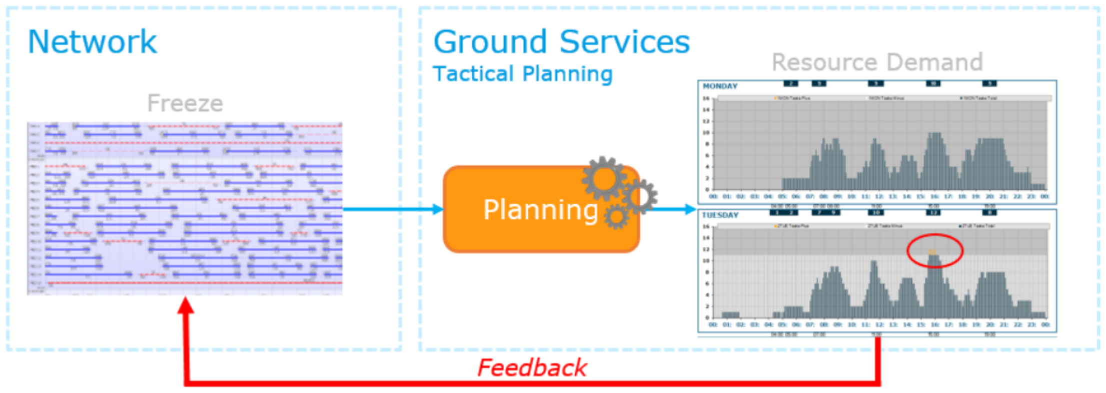


Figure 2.7: Operational Planning Cycle (OPC) between Network and GS-TP

For GS-TP this operational check means that they use their planning process (figure 2.6) to determine whether they have enough ground resources to facilitate this flight schedule. Their analysis is then used as feedback by Network to alter their flight schedule (figure 2.7). Since most of the large ground equipment - e.g. towing trucks and fuel trucks - has a lead time of 2 years, it is not possible to purchase new equipment before the schedule is due. Exceeding their capacity will result in a flight schedule that is impossible to operate without some form of delay caused by a shortage of ground equipment. Therefore Network is forced to change the schedule if it turns out that according to GS-TP calculations it will not be operationally feasible.
It takes on average around a full night for this process to run on GS-TP computers. More important runs, on which executive decision making is based, require human intervention. Therefore it can take days before the resource demand is determined.

Chapter 3

Problem Description

The problem that the Network department addresses, is that it is often surprised by the outcome of the OPC-check performed by GS-TP. Sometimes a schedule is rejected because of a shortage in resources that has not been foreseen by Network's capacity planners and their current tooling (Networks gate planning and BTS-workload graphs). Furthermore, when a schedule is rejected, Network does not really know what schedule changes have to be made such that GS-TP will likely accept it. They would like to have their own small feedback cycle in order to include ground resource capacity in their regular network scheduling process (figure 3.1). Since the OPC-check is now performed every 4 weeks instead of semiannually (section 2.4), and Schiphol starts to expand rapidly (section 2.1), the needs of such a feedback cycle are increasing.
Ideally, Network would like to agree upon Key Performance Indicators (KPIs) of a flight schedule that directly indicate operational feasibility. This would make it much easier for Network's capacity planners to take ground resource capacity constraints into account while scheduling a flight network.

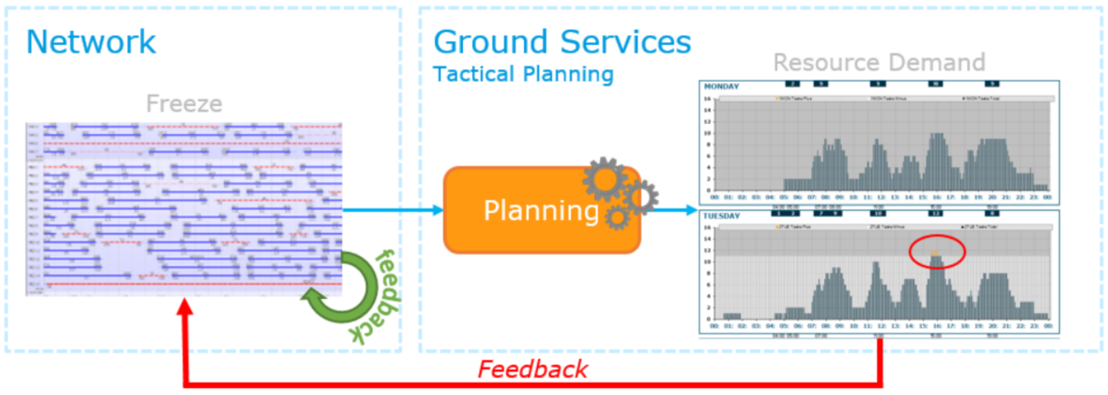


Figure 3.1: Operational Planning Cycle (OPC) between Network and GS-TP. The Network department would like to have their own small feedback cycle (green arrow) in order to evaluate a schedule before it is sent to GS-TP.

Another point of discussion is that GS-TP evaluates the resource demand given a planned schedule. This will never be the exact schedule that is operated by KLM because of stochastic events like flight delays. Sometimes it is simply impossible to operate a schedule exactly as it is planned.

For example, it often happens that more than 15 aircraft are scheduled to arrive at the same time while this violates Schiphol's runway capacity constraint. GS-TP occasionally applies corrections for these effects but it has not been studied in greater detail yet.
In order to get an indication of the resource demand on the day of operation, the current process of Capacity Requirements Planning (CRP) of section 2.3.3 is not sufficient anymore. More sophisticated models are necessary in order to include flight delays in resource planning.

3.1 Research Questions

The problems described above can be summarized into three main research questions that form the assignment of this master thesis:

1. What is the relation between a flight schedule and the resource demand for the most critical ground resources as calculated by Ground Services - Tactical Planning and is it possible to quickly verify whether a flight schedule will be accepted after an operational check?
2. Given that a schedule will not be accepted by Ground Services - Tactical Planning, how can the Network capacity planners be advised to alter the schedule such that it increases the probability of acceptation?
3. Given a flight schedule, what can be said about the demand for ground resources on the day of operation?

3.2 Scope of Thesis

This thesis aims to answer the research questions above. Before doing so, the scope needs to be clarified.
The first research question is fixed on quickly computing the resource demand as calculated by GS-TP and can be seen as a shortcut for the computationally expensive task administration. This will be done for the most critical resources only, i.e. the resources that are most likely to cause infeasibility of a schedule.. In general, these are the biggest pieces of equipment since they are expensive and have a relatively long lead time. These resources are listed in table 3.1.

Type of Resource	*Name of Equipment*		
Towing Trucks	AM500	AM210	AM110
Fuel Trucks	Dispensers	KTW	
Baggage Loaders	Rampsnakes	Powerstows	

Table 3.1: Critical Resources

Research question 2 focuses on the same resources as the first research question (table 3.1). The goal here is to provide more information to the Network department than just the demanded number of resources.
The last research question focuses on applying some stochasticity to the flight schedule. This is aimed to be achieved by two approaches: 1. A flight specific simulation that uses the the

administrative shortcut of research question 1 to evaluate simulation outcomes, and 2. By a continuous time Markov-chain (CTMC) model of movements of aircraft on Schiphol. The simulation concentrates on KLM flight delays and their influence on the resource demand for the resources described in table 3.1. The latter approach is fixed on wide body aircraft in general and only accounts for towing trucks and gates.
These research questions are distributed over this report in the following way:

Part I	Research questions 1 and 2.
Part II	Research question 3, simulated approach.
Part III	Research question 3, CTMC approach.

Part I
Estimation of Planned Ground Resource Demand

What is the relation between a flight schedule and the resource demand for the most critical ground resources as calculated by Ground Services - Tactical Planning and is it possible to quickly verify whether a flight schedule will be accepted after an operational check?

Given that a schedule will not be accepted by Ground Services - Tactical Planning, how can the Network capacity planners be advised to alter the schedule such that it increases the probability of acceptation?

Chapter 4

Methods

This chapter describes the methods that are used to find an estimation of the demand for ground resources as calculated by GS-TP. It thereby focuses on answering research question 1: whether a clear relation between a flight schedule and the resource demand for the most critical resource can be found and whether it is possible to quickly verify if a flight schedule is likely to be accepted after an operational check. Furthermore, research question 2 - how can Network be advised to alter the schedule to increase the probability of acceptation? - is discussed.

It is important to note that the goal is to estimate what the resource demand will be as calculated by GS-TP using their planning process described in section 2.3.3. At this point it is not of importance whether this planned demand is realistic. GS-TP has the authority to not accept a schedule motivated by their current planning and Network faces the problem that they lack insight in this process. Therefore it is assumed that this process, that is carried out by GS-TP, is a realistic way to estimate the demand for its resources.

Machine learning models are discussed to quickly estimate the demand for the most critical resources per 5 minute time brackets given a flight schedule. In order to apply these models, a flight schedule needs to be quantified and business rules have to be implemented to include as much known information as possible. Figure 4.1 shows that this quantification is achieved by converting a Freeze, i.e. flight schedule, to a so called ground schedule, after which characteristics of flights and tasks can be extracted. These characteristics that describe a flight schedule will be called "schedule characteristics". A complete list is of these schedule characteristics is given in Appendix B. Furthermore, this chapter gives an explanation of validation methods and the data sources that are used.

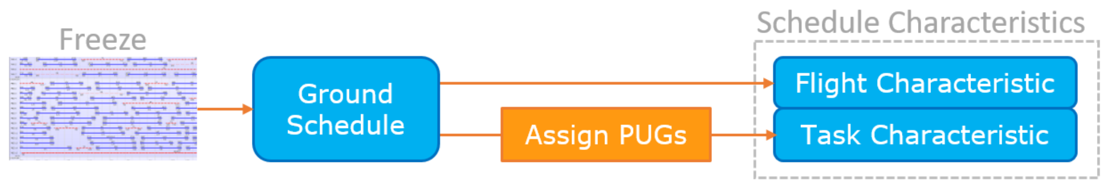


FIGURE 4.1: Process to calculate schedule characteristics in order to quantify a flight schedule (Freeze).

A motivation for the use of machine learning models is given in section 4.1, followed by some background information on the planning of critical resources in section 4.2. Then, two sections follow to describe the way a flight schedule (4.3) and the ground services business rules (4.4) are quantified. In section 4.5 it is explained how information from other time brackets can be used to add information about the resource demand in a certain time bracket. Gradient Boosting machine learning models are described in section 4.6 after which section 4.7 sketches the available data. Finally, the evaluation and validation methods are described in section 4.8.

4.1 Motivation

The Network department of KLM is looking for a way to have an indication of the resource demand as calculated by GS-TP. In fact, they want to have a shortcut for the administrative Capacity Resource Planning. Preferably, Network would like to agree upon schedule KPIs that can be used to directly imply operational feasibility.
It can be seen as if they are looking for a function $f(x)$ that takes schedule x as an input and outputs a resource demand predictor $\hat{y}$ (figure 4.2). This predictor should estimate real value y; the resource demand as calculated by GS-TP (section 2.3, figure 2.6 in general).

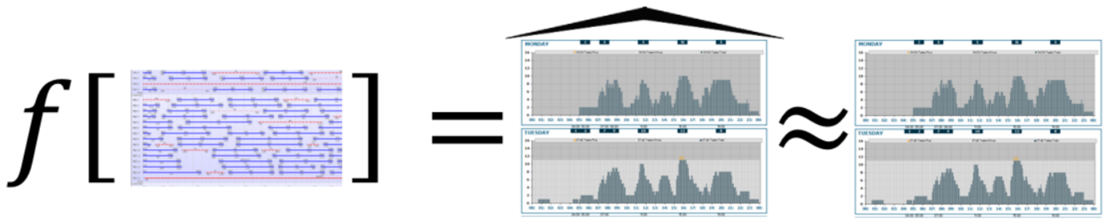


FIGURE 4.2: Function $f(\cdot)$ takes a schedule as input and returns a predictor that estimates the resource demand as calculated by GS-TP.

Machine Learning (ML) models are models that try to find the best function $f(x)$ that describes the relation between some available input and output [1]. When this relation is found, i.e. a model has been trained, it can be used to "predict" the resource demand of a Freeze that has not been processed by GS-TP (figure 4.3). The motivation to use this approach is that it is a fast way to literally estimate the GS-TP administrative process without using the computationally expensive planning tools.

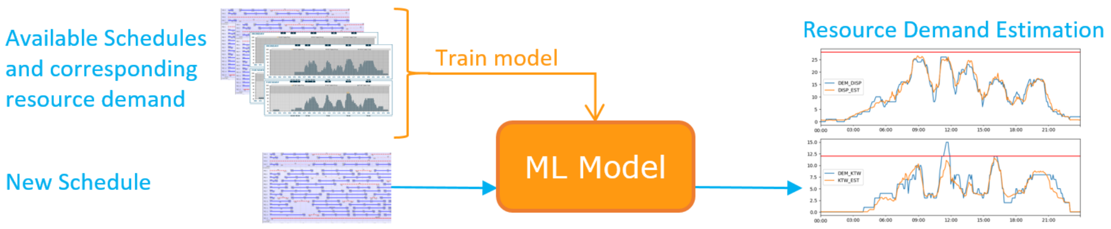

Available Schedules and corresponding resource demand
Train model
Resource Demand Estimation
New Schedule
ML Model

FIGURE 4.3: Process of training and using a machine learning model to estimate the resource demand for ground equipment.

4.1.1 ML Model Output

A ML model to indicate whether a schedule is operationally feasible can have different outputs. It can return how sure the model is that a schedule will be accepted by GS-TP. In that case the model output might be a single number between 0 and 1. This might be a good indicator for the Network department but it does not give information about at what moments the schedule is likely to be infeasible. It also does not tell what to do in the case of infeasibility (research question 2 in section 3.1). Therefore an output of the model that says something about the demand per resource per time bracket is preferred. A time bracket of 5 minutes is used since KLM schedules flights per 5 minutes.

4.1.2 The Usage of a Gate Planning

There is a high correlation between a gate planning and the corresponding demand for resources. Some types of equipment might only be used on particular gates and when a gate planning shows that all aircraft can be assigned to a gate, no towing equipment is necessary to tow aircraft to a parking location. Therefore, a gate planning will be a an important feature in a model to determine the demand for ground resources.
As discussed in chapter 2, both the Network department and GS-TP have their own gate planning tools. These tools differ in such a way that their outcomes are not comparable. This has led to a discussion between the two departments and until today they cannot agree upon its output. Therefore the gate planning tool that is used by Network will not add much information to the model and it is chosen to not include a gate allocation as a feature for the ML models.

4.2 Background on the Planning of Critical Resources

The critical resources that are focused on in this thesis are planned by ground services using their planning rules (PUGs) and tooling as reported in section 2.3. This section describes the process of planning these critical resources in more detail. Here it is important to specify two types of tasks that can be assigned by PUGs.

Definition 4.1. Fixed Task. A task for which the planned start- and end time is fixed. For example, some PUG may specify that a task will be planned to start at $D - 16$, end at $D + 5$ and will take 21 minutes to fulfill.

Definition 4.2. Shiftable Task. A task for which we know that it has to be planned in a certain time interval, but the start- and end time are not fixed. For example, some PUG may specify that a task has to be planned to start after $A + 5$, end before $D - 5$ and take 27 minutes in total. This means that this task can be shifted in a certain interval depending on the arrival and departure time of the turn.

The plannings rules that describe fixed or shiftable tasks can be referred to as a fixed or shiftable PUG respectively. For resource that only contain fixed tasks it is immediately clear what the

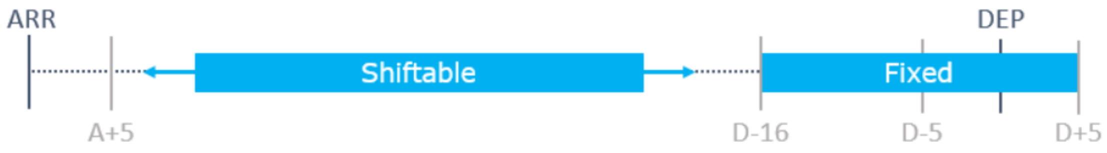


Figure 4.4: Fixed and shiftable task according to the examples in definitions 4.1 and 4.2
.

planned resource demand will be since there is no flexibility allowed at planning level. Resources that also contain shiftable tasks require additional scheduling in order to determine the planned start time of these shiftable tasks. During the scheduling of shiftable tasks, GS-TP has the objective to use as little resources as possible to carry out all planned tasks.

4.2.1 Gates

Although gates are not resources whose demand is aimed to be estimated by this thesis, the gate allocation has a big influence on the use of other resources. For example, towing an aircraft is only necessary when there is shortage of gates on that moment. As motivated in section 4.1.2, a precise gate allocation will not be used as a feature for the models but simple information about gate planning can easily be included.
The business planning rules for gates are always of the form

```
if duration of turn $\leq x$ then:
    demand for gate for total ground handling from $A - 0$ to $D - 0$
else then:
    demand for gate for arrival handling from $A - 0$ to $A + \alpha$
    demand for gate for departure handling from $D - \delta$ to $D - 0$
```

Where x, α and δ depend on aircraft type and airline, x denotes the maximum ground time in minutes at which an aircraft cannot be towed away from the gate, and α and δ are the time in minutes necessary for arrival and departure handling respectively.
During resource planning it is common to add separation times to gate demand to include some slack to manage small delays on the day of operation. GS-TP uses a separation time of 20 minutes in their gate allocation tooling. This means that a gate is considered to be occupied from 10 minutes before the planned arrival and until 10 minutes after the planned departure.
Applying these rules to all turnarounds, i.e. the period between arrival and departure of an aircraft, will result in a list of periods for which aircraft require a gate. Counting up these periods that overlap a certain time interval will give the minimum gate demand for that time interval.

4.2.2 Baggage Loaders

Baggage loaders are used to transport baggage in and out of aircraft. The baggage loaders in the scope of this research - powerstows and rampsnakes - are literally conveyor belts on wheels. In the current planning process these resources can only have fixed tasks assigned to them.

Therefore they do not require additional scheduling since the start and end time of the planned tasks are fixed.
The only additional planning rule is that if an arrival and departure handling overlap, they are combined into one single task. For example, let a 40 minute turn have an arrival handling by a rampsnake from $A - 2$ to $A - 20$ and a departure handling from $D - 35$ to $D - 2$. These tasks overlap and will therefore be merged to one task from $A - 2$ to $D - 2$.
Applying these rules to all turnarounds will result in a list of periods for which aircraft will require a ramsnake or powerstow. Counting these periods that overlap a certain time interval will give the resource demand according to GS-TP's resource planning.

4.2.3 Towing Equipment

Towing equipment is used for both push backs and towing. A push back is the operation that is performed to push an aircraft away from a gate to a position from which the pilot can safely start taxiing to the runway. A tow is carried out to bring an aircraft from the gate to a parking position or maintenance hangar or the other way around. GS has multiple types of towing trucks that vary in size and all have a specific set of aircraft types that can be handled by these trucks. From small to large the truck types that are considered to be critical are called AM110, AM210 and AM500. There are, however, some smaller truck types that are not considered in this thesis. Push backs are always planned as fixed tasks. Tows, however, can be shifted tasks as well. An additional uncertainty is that a towing task, in contradiction to tasks of other resources, only has to be planned when there are not enough gates available. Therefore the demand for this resource is highly dependent of the gate allocation. This it tricky since the idea of the machine learning model approach is to not include a gate planning at all. Travel times are included in the task times described by the PUGs.
The idea is that, in order to obtain the demand for towing tasks, the machine learning models will find relations between the number of aircraft at Schiphol, the number of possible towing tasks that can be assigned to a schedule according to the PUGs, and the number of resources planned by GS-TP.

Static Tasks
Towing and push back tasks are planned according to the PUGs and specifications of aircraft turnarounds. Additionally, GS-TP also schedules so called static tasks that are not related to specific aircraft turns. These tasks compensate for the unforeseen demand for towing equipment by, for example, additional maintenance. These shiftable tasks are of the form: plan two extra tasks that take x minutes between 1PM and 3PM daily. Determining the planned start time of these tasks goes according to GS-TP's plannings objective: to use as little resources as possible to carry out all tasks.

Overflow Towing Tasks
As described in section 2.3.2, the planners at GS-TP sometimes allow a gate planning to not perfectly fit. In this case so called overflow gates are generated. Overflow towing tasks are planned to tow aircraft to and from these virtual gates.

AM110 trucks are only used for push backs and therefore are easily planned since these only include fixed tasks. Summing up these tasks already provides the exact resource demand as calculated by GS-TP's planning process. Both AM210 and AM500 perform both push backs and tows and have additional static tasks.

4.2.4 Fuel Equipment

The fuel equipment that is considered to be critical are small fuel tucks (KTWs - "Kleine TankWagens") and dispensers. KTWs transport kerosene to an aircraft. Dispensers, however, are used as a connector between an aircraft and Schiphol's fuel pipeline network.
The business rules for planning fuel equipment mainly consist of shiftable PUGs. An additional complexity is that the resource that is used to fulfill the task dependends on the gate on which the aircraft is positioned. Almost all category 3 to category 9 gates are hydrated, which means that a dispenser can be used to tank fuel from the pipelines into the aircraft. On other positions, a KTW or GTW ("Grote TankWagen" - big fuel truck) has to be used. There are just a few GTWs in use at GS and they are not considered to be critical.
The tasks that are planned for fuel equipment do not contain travel times. These travel times are added according to a travel time matrix that indicates the time to drive between some regions on Schiphol.

Static Tasks
Just like towing equipment, additional static tasks are planned in order to compensate for unexpected demand. For KTWs these static tasks are also a reservation of equipment necessary to fill the tank at a filling station. For towing equipment, these static tasks could be shifted. For fuel trucks, however, the static tasks are a fixed reservation for some part of the day.

4.3 Quantifying a Flight Schedule

The goal of quantifying a flight schedule is to present the schedule in such a way that it can be used as an input for machine learning models. This is done via a conversion of the flight schedule to its corresponding ground schedule, which is used to determine schedule characteristics per 5 minute time bracket.
A flight schedule contains a lot of information about the planned operations of all airline activities around Schiphol. This information can be categorized into three categories.

1. Flight information - flight specific information such as departure time, arrival time, origin, destination, aircraft type and airline (Available for all airlines).
2. Information about aircraft reservations - when to reserve time for maintenance and backup (KLM only).
3. Schedule information - i.e. how these flights and aircraft reservations are distributed over the available fleet.

The first two can be considered as fixed during the evaluation of an OPC-Freeze. For KLM and partners this information is available, for other airlines the flight information is estimated using historic flights and slot availability. The schedule information depends on which algorithm (FIFO-LIFO) is used to distribute the flights and aircraft reservations over the available aircraft. GS-TP uses the FIFO implementation with a switch to LIFO when turnarounds are longer than 180 minutes.
A first step in quantifying is to obtain a ground schedule from a flight schedule. From this ground schedule the schedule characteristics regarding the flights can be directly computed.

4.3.1 Ground Schedule

Because this research focuses on what happens in between the flights, the flight schedule is converted into a so called ground schedule. This ground schedule is a tabular representation of all turns that occur on Schiphol where every record represents a turnaround. Aircraft reservations, such as maintenance and back-up aircraft, are not included as separate elements but as a characteristic of a turn. For each turnaround we require the following information:

Aircraft Type	Three letter IATA aircraft code, e.g. 73H or 744
Aircraft Category	Category (1-9) depending on aircraft size
Flight From	Flight number of incoming flight, e.g. KL0862
Flight To	Flight number of outgoing flight, e.g. KL0743
Airline	Two or three letter IATA airline code of airline, e.g. KL
Arrival Datetime	Local date and time of arrival on AMS
Departure Datetime	Local date and time of departure from AMS
Duration of Turn	Time difference in minutes between arrival and departure
Origin	Three letter IATA airport code of origin of incoming flight
Destination	Three letter IATA airport code of destination of outgoing flight
Turn type arrival	Code of first aircraft reservation after arrival*
Turn type departure	Code of last aircraft reservation before departure*

*Aircraft reservation codes that are of interest for resource planning are maintenance checks `##A`, `##C`, `##T`, and aircraft stand by reservation `#RS`.

It happens that in the flight schedule an incoming flight cannot be linked to an outgoing flight and vice versa. It then appears as if an aircraft will be at Schiphol for a very long time. This happens because the schedules of non partner airlines are estimated based on previously flown schedules and slot availability. In operations it almost never happens that an aircraft of such an airline will spend that much time on Schiphol. Therefore dummy flights are added such that every aircraft that is planned to be at AMS can be described as a turnaround in the ground schedule. These dummy flights do not require any ground handling and are only used to prevent that unrealistically long ground times occur.

4.3.2 Schedule Characteristics for Flight Information

The ground schedule can be used to calculate schedule characteristics per 5 minute time brackets. These characteristics form a quantitative representation of the flight schedule. The idea is to describe all events using only three metrics; number of arrivals, number of departures and maximum number of aircraft on AMS. These metrics can be calculated for different subgroups of turnarounds that are likely to have some effect on the demand for ground resources. For example, we would like to know these metrics for all aircraft categories separately since aircraft category determines the gates on which aircraft can be allocated. Next to KLM, other airlines such as Air France (AF) and Delta (DL) have gate preferences. Including the three metrics for all turnarounds of these airlines will add information about the gate allocation. The same argument holds for Low Cost Carriers, like EasyJet and RyanAir, and Freighters, who only transport cargo.
In total the following metrics and subgroups are defined to quantify the flight schedule.

Metrics

$\texttt{AC\_AMS}_t$	Maximum number of aircraft on AMS in time bracket t.
$\texttt{ARR}_t$	Number of arrivals in time bracket t.
$\texttt{DEP}_t$	Number of departures in time bracket t.

Subgroups

KL	KLM aircraft
AF	Air France aircraft
DL	Delta aircraft
F	Freighters (cargo aircraft)
LC	Low Cost carriers
1	Category 1 aircraft
2	Category 2 aircraft
⋮	⋮
9	Category 9 aircraft

For `AC_AMS` it is explicitly mentioned that the *maximum* number of aircraft per time bracket is taken because this metric is not a count of events, like `ARR` and `DEP`, but a count of intervals that happen to overlap a time bracket. However KLM only schedules flight with 5 minute precision, other airlines can schedule flight per minute. $\texttt{AC\_AMS}_t$ is thus the maximum of the number of aircraft on AMS of all minutes in interval t.
These metrics and subgroups can be combined using an underscore (`_`) to form schedule characteristics. For example $\texttt{ARR\_KL}_t$ denotes the number of arrivals of KLM aircraft in time bracket t. Combinations of subgroups, like $\texttt{ARR\_KL\_4}_t$, are also possible. See table 4.3 for an example of some of these schedule characteristics and how they describe the situation on AMS. The most left column indicates the start time of a bracket. The difference of the situation on AMS between the selected time periods 09:00 - 09:15 and 20:05 - 20:20 is clearly visible. For example, in the first period there are relatively more wide body aircraft (cat 6-9) than smaller medium body

(cat 4-5) and narrow body (cat 1-3) relative to the second period. This is already an indication for the usage of gates and other resources.

t	$\texttt{AC\_AMS}_t$	$\texttt{AC\_AMS\_123}_t$	$\texttt{AC\_AMS\_45}_t$	$\texttt{AC\_AMS\_6789}_t$	$\texttt{AC\_AMS\_AF\_123}_t$	$\texttt{AC\_AMS\_DL}_t$
09:00	118	21	45	52	2	9
09:05	119	20	46	53	2	10
09:10	121	20	48	53	2	11
⋮	⋮					⋮
20:05	111	35	55	21	2	1
20:10	110	35	54	21	1	1
20:15	105	34	50	21	1	1

t	$\texttt{ARR}_t$	$\texttt{ARR\_F}_t$	$\texttt{ARR\_LC}_t$	$\texttt{DEP}_t$	$\texttt{DEP\_F}_t$	$\texttt{DEP\_LC}_t$
09:00	13	0	1	5	2	1
09:05	2	0	0	1	0	0
09:10	5	0	2	3	0	1
⋮	⋮					⋮
20:05	2	0	2	1	0	0
20:10	4	0	2	1	0	1
20:15	1	0	0	6	0	0

Table 4.3: Example of schedule characteristics per 5 minute time brackets.

Note on `AC_AMS`

The flight schedule of KLM, AF and partners is well known. For other carriers, however, it is a good guess what their schedule will be. Because of this guess it sometimes happens that aircraft arrive on AMS without ever departing again or the other way around. In the ground schedule this problem is tackled by adding dummy flights. For the `AC_AMS` characteristic, these aircraft are only included from the moment they have a planned task assigned to them (see next section).

4.4 Adding Ground Services Business Rules

GS-TP business rules (PUGs), as described in section 2.3.1, contain a lot of information about the resource demand. The idea is to use this information by assigning tasks to turns in the ground schedule. Each PUG specifies both the type of turnarounds on which it applies, and the task that has to be scheduled for this turnaround - i.e. task duration and necessary equipment. Therefore, PUGs can be directly assigned to turns that are included in the ground schedule after which they can be converted to task related schedule characteristics to give additional information about the resource demand for every time bracket.

4.4.1 Quantifying PUGs

Fixed PUGs (definition 4.1) can simply be used to assign planned tasks to turns, after which the number of tasks per equipment that lie in a certain time interval can be counted. This is not possible for PUGs that describe tasks that can be shifted. However, it is possible to give an indication of the “uncertain” resource demand generated by these shiftable PUGs by breaking down the task and define a probability distribution that a task will be planned at any point in

time. Here "uncertain" refers to the fact that one cannot be sure that a task will be schedule at that moment. These probability distributions differ from the mathematical definition in the sense that they indicate the probability that some part of the task will be planned at a certain moment instead of the probability that some event happens. Therefore, the area under the graph will not be equal to 1 - as with the regular mathematical definition - but it has to be equal to the total duration of the task.
Figure 4.5 shows a task that takes 25 minutes that has to be planned between $D-30$ and $D+5$. It is clear that, independent of the actual start time of this task, this task will definitely be scheduled to be executed between $D-20$ and $D-5$. This is true since it has to start after its earliest start ($D-30$) and cannot be started after $D-20$ because otherwise the task will end after its latest end ($D+5$). For the other parts, between the earliest start and the latest start and between the earliest end and the latest end, a probability of .5 that the task will be planned at those point in time is used. When there is not a certain interval that will definitely contain at least a part of the task, i.e. the time difference between the earliest start and latest end is more than double the duration of the task, the probability can be uniformly distributed over the total task interval.

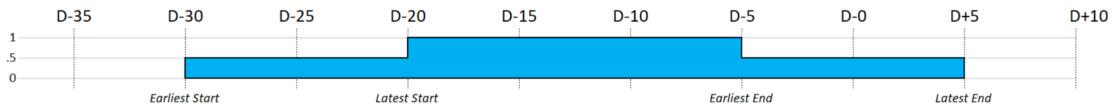


FIGURE 4.5: Example of a probability distribution of a 25 minute shiftable task that has to be scheduled between $D-30$ and $D+5$.

One can think of other methods to distribute the task probability over the uncertain intervals. A reasonable approach is sketched in figure 4.6 where it is assumed that a task is randomly planned between its earliest start and latest end. This results in a probability distribution with the same certain part between the latest start and earliest end, but with an increasing and decreasing slope between the earliest and latest start and the earliest and latest end respectively. This seems like a fair distribution but in the planning process shiftable tasks are often started at - or close to - their earliest start or their latest start. This is a direct result of planning with the objective to use as little resources as possible: when there is a peak in resource demand at one of these two intervals, the task will be shifted such that it will not higher this peak even more. Therefore the task probability distribution from figure 4.5 will be used.

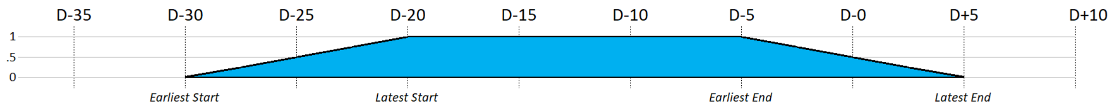


FIGURE 4.6: Example of an alternative probability distribution of a 25 minute shiftable task that has to be scheduled between $D-30$ and $D+5$.

4.4.2 Schedule Characteristics Based on PUGs

Per equipment a schedule characteristic can be created for every type of task this equipment can fulfill. This, of course, differs per equipment and depends on how GS-TP plans these resources (section 4.2). For every type of task, two types of metrics per time interval can be defined: one

metric that is the count of all fixed tasks and all shiftable tasks that have a probability of 1 of being planned in that time interval, and another metric that is the sum of all probability distributions for all tasks of that type of equipment. The latter will be indicated with suffix `_unc`.
This section describes per resource group the schedule characteristics that are generated. The schedule characteristics that are based on shiftable tasks have an additional schedule characteristic with suffix `_unc`. This uncertain characteristic is not described for every resource independently. A full list of schedule characteristics can be found in Appendix B.

Gates
For gates, two schedule characteristics are generated. They sum up the gate demand per time interval

$\texttt{AC\_GATE}_t$	Number of aircraft that require a gate for time interval t.
$\texttt{AC\_GATE\_SEP}_t$	Number of aircraft that require a gate including a separation time of 20 minutes. 10 minutes before and 10 minutes after each gate occupation.

Baggage Loaders
Since baggage loaders only have fixed tasks that are assigned to them, they will only have schedule characteristics that count the number of these tasks per time interval.

$\texttt{RAMP}_t$	Number of rampsnake tasks per time interval.
$\texttt{POWER}_t$	Number of powerstow tasks per time interval.

Towing Equipment
As indicated before, towing tasks have the additional complexity that these tasks only have to be planned when there is a shortage of gates. At this point the focus is on assigning towing tasks without considering whether they actually have to be planned or not. This will already give an indication of the maximum number of towing tasks that can be scheduled.
Because maintenance and other aircraft reservations are already included in the flight schedule, and we know that these events will trigger towing tasks that have to be realized, these tasks are separated from other towing tasks. For these tasks it is known to which region on Schiphol these aircraft have to be towed, which gives a pretty clear indication of the towing time. For other towing tasks, the parking location is not known since a gate planning has not been realized. Therefore two types of towing PUGs will be assigned to every turn that might have a tow. One for relative short tows to parking positions that are close to Schiphol's gates, and one for relative long tows to Schiphol East and parking positions that are further away.
Static tasks are planned after the demand for other tasks of tow equipment is determined. Therefore these tasks are not described by schedule characteristics but added afterwards.

$\texttt{PB}_t$	Number of push backs in time interval.
$\texttt{TOW\_MAIN}_t$	Number of maintenance tow tasks in time interval.
$\texttt{TOW\_NORM}_t$	Number of normal length tow tasks in time interval. Note that these tasks do not necessarily have to be planned.

`TOW_EAST`$_t$	Number of tow tasks to Schiphol East or far parking locations in time interval. Note that these tasks do not necessarily have to be planned.

These schedule characteristics can be split for the three towing trucks, in which case they will have a suffix `_AM500`, `_AM210` or `_AM110`.

Fuel Equipment
Because the resource that is necessary to fulfill a fuel task dependents on the gate allocation, it is not possible to generate schedule characteristics per equipment type. Tasks are therefore included using schedule characteristic `FUEL` without specifying equipment. For freighters, however, the PUGs differ for hydrant and non-hydrant positions. This is indicated by the suffixes `_Hydrant` and `_NonHydrant`. Because the planned start and end time of static tasks are fixed for fuel equipment, they can be included as schedule characteristics directly.

`FUEL`$_t$	Number of fuel tasks in time bracket.
`FUEL_Hydrant`$_t$	Number of freighter fuel tasks in time bracket in case that aircraft is located at a Hydrant position.
`FUEL_NonHydrant`$_t$	Number of freighter fuel tasks in time bracket in case that aircraft is not located at a Hydrant position.
`DISP_STATIC`$_t$	Number of dispensers that are reserved for a static task.
`KTW_STATIC`$_t$	Number of KTW's that are reserved for a static task.

4.5 Adding Information From Other Time Brackets

In the previous section it is described how schedule characteristics can be constructed in order to provide quantitative information that might indicate the resource demand for some ground equipment in some time bracket. Besides the information about the schedule and resource planning rules per time bracket, models may improve from knowing what happens around this time bracket. It makes sense that the resource demand at a certain moment does not only depend on what happens on that moment, but also on what happens before and after that moment. This information is added in two forms: time shifted schedule characteristics and uncertain shift characteristics.

4.5.1 Time Shifted Schedule Characteristics

The idea of time shifted schedule characteristics is to add information about the value of schedule characteristics directly around every time bracket. This is done by summing some schedule characteristics for the next or previous x minutes relative to the start of a time bracket. These time shifts are added as schedule characteristic and can be recognized by suffixes `_Px` and `_Nx` for previous and next x minutes respectively.

`_Px`	Suffix to denote the sum of a schedule characteristic of the previous x minutes
`_Nx`	Suffix to denote the sum of a schedule characteristic of the next x minutes

4.5.2 Uncertain Shift Characteristics

Figure 4.5 shows the probability distribution of a shiftable task. This task contributes to the total workload of the time intervals it overlaps. However, the probability that it will start at its earliest start depends on the resource demand in the time intervals between its earliest end- and latest end times. If the demand is high in these time intervals, the task will more likely be started at its earliest start in order to decrease the demand in these time intervals.
The uncertain shift characteristics try to add information to account for this effect. The idea is to add a schedule characteristic for the time intervals at the left end of this probability distribution that indicate the resource demand at the right end of the distribution and vice versa. These characteristics can be recognized by the `_UNCSHIFT` and are applied to all schedule characteristics with some uncertain part.

`_UNCSHIFT`	Suffix of an uncertain schedule characteristic that sums up the value of this characteristic on complementary time brackets of the uncertain parts of task probability distributions that cover the current time bracket.

4.6 Machine Learning Models

Machine Learning (ML) models are used in order to find relations between a flight schedule and the demand for ground equipment [1]. These models can be trained on available data such that they can be applied on new schedules for which this demand is not yet known. The concept is to find a function that takes the schedule characteristics per 5 minute time bracket as an input and returns an estimated demand per resource for that time bracket.
This section first describes the necessary models that have to be trained in order to give an indication of the resource demand for critical resources. After that the Gradient Boosting algorithm is briefly explained.

4.6.1 Model per Critical Resource

As defined in the scope of this thesis (section 3.2), the resource demand for towing trucks, fuel trucks and baggage loaders has to be estimated. For each type of equipment a separate model is built. Since all resources follow different planning rules they require a slightly different approach.

Baggage Loaders

Both rampsnakes and powerstows do not have shiftable PUGs that assign planned tasks to them. Therefore the demand for these resources can be simply calculated when the GS-TP planning rules are implemented. The demand for these resources therefore does not have to be estimated by a machine learning model.

Towing Trucks

AM110 towing truck is only used for push backs. Since every departing aircraft needs a push back, and these tasks can not be shifted, the resource demand for truck type AM110 can simply be calculated just like the the demand for baggage loaders. The type AM210 and AM500 trucks, however, are also planned to carry out towing tasks. The demand for these resources for the uncertain towing tasks will be estimated by a ML model. Also the demand for overflow tows, as introduced in section 4.2.3, will be estimated by a ML model. Finally, the certain demand for push backs and the result of the ML models regarding the demand for tows and overflow tows will be combined. After that the demand for static tasks is added according to the static task planning rules (see section 4.2.3).

Fuel Trucks

Both KTWs and dispensers do not have travel time included in their PUGs. This travel time is added separately according to a travel time matrix and depends on the gate allocation. For these resources, two approaches are used to model the total resource demand. The first approach is to build a ML model that tries to estimate the total demand for these resources. An alternative is to separate the travel time part of the resource demand from the demand for actually performing fuel tasks and to build two separate ML models per resource to estimate these separately. This last model is referred to as the “splitted” approach.

4.6.2 Gradient Boosting

Gradient Boosting is a decision tree based machine learning technique that has been developed since the late nineties [14–16]. Boosting refers to the fact that multiple weak learning models are combined to form a strong well performing model. It uses gradient descent to iteratively build decision trees that try to predict the error of its preceding tree. Figure 4.7 shows how these trees are built iteratively for a regression problem. In this figure, tree 1 shows a rather bad estimation of the ground truth. The errors of tree 1 are estimated by tree 2. Tree 3, on its turn, tries to estimate the errors of tree 2. It might be clear that this technique allows to find non-linear relations.
Figure 4.8 shows an example of a regression tree that uses schedule characteristics `AC_AMS` and `TOW_NORM_AM500` to estimate the demand for towing truck AM500. It contains 3 leave nodes with corresponding weight vector $w = [0, 3, 6]$. This tree on itself is probably bad in estimating the demand but the idea of Gradient Boosting is that another tree is added to the model that tries to account for the errors that this initial tree makes.
The main advantages of Gradient Boosting are that it can be used on heterogeneous data, it

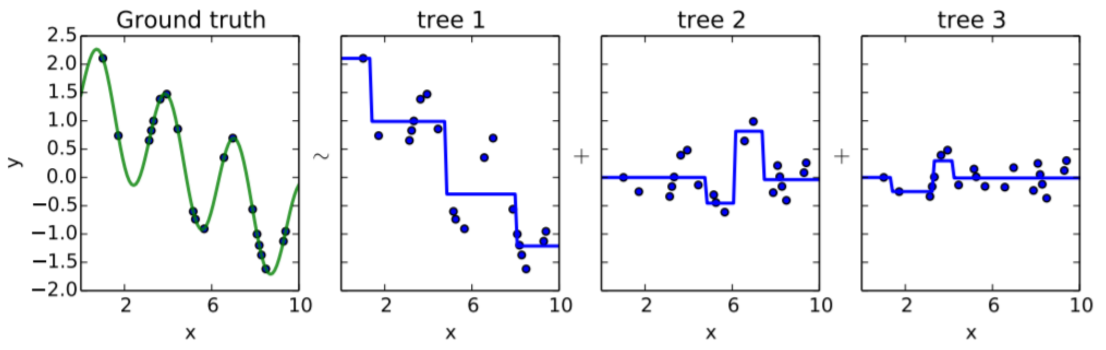


FIGURE 4.7: Gradient Boosting, trees are built sequentially. Each tree tries to predict the error of its predecessor [17].

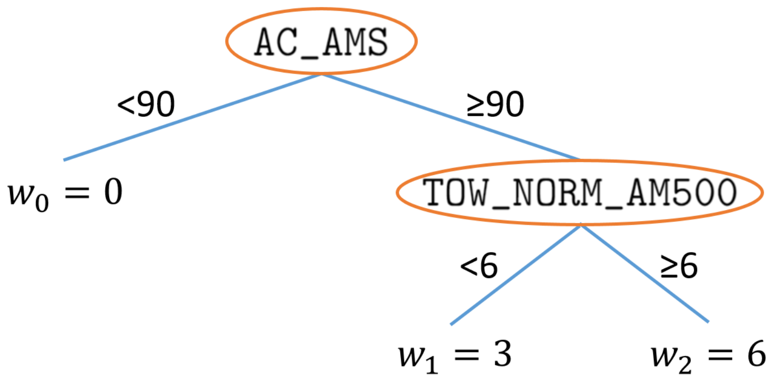


FIGURE 4.8: Example of a decision tree using schedule characteristics as an input.

supports different loss functions - as long as the the first and second derivative can be calculated - and it automatically detects non-linear feature interactions. The biggest disadvantage is that it cannot extrapolate like for example linear regression can [17].
A powerful implementation of this algorithm, XGBoost, is developed by the University of Washington [2]. It has proven to have a great performance in the well known Kaggle data science competitions [18]. XGBoost excels in the usage of computational resources by using efficient parallelization. It also uses a regularized objective function in order to control overfitting, the phenomenon that models tend to find relations from noise in the training data.
The XGBoost paper by Chen and Guestrin [2] describes the regularized objective of model ϕ, with predictor $\hat{y}$ of dependent variable y, with n observations, and K sequentially added decision trees by

$$\mathcal{L}(\phi) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{4.1}$$

where l is a differentiable convex loss function that measures how well $\hat{y}$ predicts y and the second term is a regularization to control the complexity of the k trees that are used in the model. The complexity of each tree is penalized using

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2 \tag{4.2}$$

where f is a decision tree, T is the number of leaves of this tree with corresponding weight vector w. Hyper parameters γ and λ are used to tweak the regularization.

Given the performance of a current model, a new tree is added greedily to improve (4.1). Adding a tree at the t-th iteration will change this objective by

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i + f_t(x_i)) + \Omega(f_t)$$

Here, f_t is used to denote the tree of iteration t as well as a function $f_t(x)$ that outputs the leave weight of observation x. Chen and Guestrin [2] explain exact and approximate algorithms to find a new best split and to assign weights to leaves of newly generated trees by only using hyper parameter settings and the objective's gradient and hessian functions

$$\begin{aligned} g_i &= \frac{\partial}{\partial \hat{y}^{(t-1)}} \left[l(y_i, \hat{y}^{(t-1)}) \right] \\ h_i &= \frac{\partial^2}{\partial (\hat{y}^{(t-1)})^2} \left[l(y_i, \hat{y}^{(t-1)}) \right] \end{aligned} \tag{4.3}$$

This means that every objective function - in data science also referred to as "loss" function - can be used by Gradient Boosting algorithm as long as the gradient and hessian (4.3) can be calculated. In this thesis the standard Mean Squared Error (MSE) is used as loss function. This objective aims to minimize the error between ground truth y and estimator $\hat{y}$ and punishes harder for large errors. This makes sense since a large error in estimating the resource demand can result in a wrong feedback signal to the network capacity schedulers.

4.7 Available Data and Data Source

The data that is used to train the models is exported from GS-TP planning administration tool. This system provides access to the results of the Base Rolling Planning (BRP) that were used in the Operational Planning Cycle (OPC) from the fifth 4 week period of 2017. Before this period the results of the BRP where not stored systematically but were provided in separate excel sheets and figures. This has some inconvenient consequences from which follows that the data of these BRPs is useless:

1. Only the total resource demand is available. Nowadays every task that is assigned to a resource is known. This is necessary for validation of the applied PUGs and separating the uncertain part of the resource demand. For example: tow tasks need to be separated from push backs in order to be able to build a model to estimate the demand for resources to carry out these tasks.
2. Flight schedules are stored in separate Freezes. This means that the distribution of flights over the fleet lines is not fixed an thus may differ from the distribution of flights that is used by GS-TP to calculate the resource demand.

At the start of this project the Freezes of periods 5 to 10 of 2017 were available. These Freezes will be referred to as `F17P05` - `F17P10`. In total these Freezes contain 124 weeks, corresponding to around 250 thousand time brackets of 5 minutes for which the resource demand is calculated. This should be enough data to train the models.

4.8 Validation and Evaluation Methods

In order to develop a model and verify whether it provides the desired results, evaluation metrics and a method of validation have to be defined. Evaluation metrics are used to compare models and quantify its performance. A validation is used to give an indication of the performance of the models when they are put into practice.

4.8.1 Evaluation Metrics

The evaluation of the models is done by using two evaluation metrics, the Mean Absolute Percentage Error (MAPE) and the Root Mean Squared Error (RMSE) which are defined by

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{\hat{y}_i - y_i}{\bar{y}}\right| \tag{4.4}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \tag{4.5}$$

where y_i and $\hat{y}_i$ are the actual value and forecast value of observation i and n is the number of observations. The regular definition of the MAPE says that y_i has te be used in the denominator of (4.4). Because this value might be zero or close to zero the average $\bar{y}$ is used instead.
The RMSE metric is directly related to the MSE loss function that is used to train the models. It therefore is a direct reflection of the model's performance. The MAPE is included to provide another relative performance metric.
Furthermore, since schedules are rejected if the resource demand exceeds its capacity, there is a special interest in the performance at the peaks of the demand. This performance can be evaluated by applying the metrics defined by (4.4) and (4.5) to all observations that have a resource demand that is higher than a certain percentile. In this study both metrics are evaluated from the 95-th and 99-th percentile resulting in the additional measures $MAPE95$, $MAPE99$, $RMSE95$ and $RMSE99$. An additional measure that accounts for peak performance but does not cancel out the performance on other observations is the Weighted Root Mean Squared Error defined by

$$WRMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left[(\hat{y}_i - y_i)\frac{y_i}{\bar{y}}\right]^2} \tag{4.6}$$

4.8.2 Validation Method

The classical way to validate machine learning models is to split the data in three sets [1]. A ***training set*** is used to train the models by providing them information on the input (schedule characteristics) and the desired output (resource demand). These models can then be validated using a ***validation set***. This provides information about how well the models perform on data that it has not seen before. This process allows to choose the right model and to adjust model parameters in order to improve its performance. When a final model is chosen, its expected performance in real life can be tested using a ***test set***. This part of the data is not used to train the model nor to make modeling decisions. Figure 4.9 shows the iterative process of training and validating in order to find the right model, after which the performance of this model is verified using the test set.

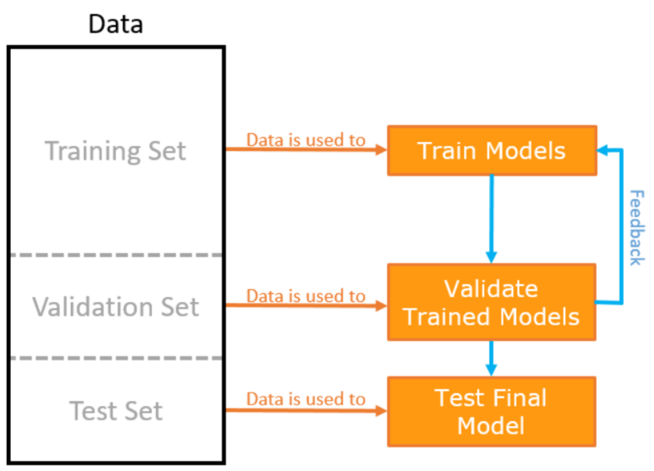


FIGURE 4.9: Testing and validation of machine learning models.

Because the data are time dependent, it makes sense to split the data in a way obeying this time dependency. Let's suppose that `F17P10` has not been calculated yet by GS-TP. This Freeze will function as test Freeze. The other Freezes, `F17P05 - F17P09`, will be used for testing and training. There will always be one single test Freeze where a model is tested that has been trained on Freezes of earlier periods. This is a time-dependent form of cross-validation [1].

Chapter 5

Results

The results of the Gradient Boosting models as described in chapter 4 are presented in this chapter. The goal of this chapter is only to present the results, a discussion can be found in chapter 6. The results are presented per resource or per type of task that a resource could be assigned to. The performance of the models is given in table 5.1 according to the performance metrics as defined in section 4.8. Furthermore, a visual result is shown that is representative for the performance of each model on an average day. In these figures the current capacity is shown by a horizontal red line. Finally, some examples are given where the models show very poor performance.
As explained in section 4.8, all results are obtained from applying the models on the data of test Freeze `F17P10`, i.e. the 10-th four weekly Base Rolling Planning of 2017, while the models are trained and validated on `F17P05 - F17P09`.

	Tow AM500	**Tow AM210**	**Overflow AM210**	**KTW**	**KTW Split**	**DISP**	**DISP Split**
mape	25.019	59.714	121.739	17.603	17.945	11.102	11.846
mape95	13.320	28.493	58.206	16.652	15.987	5.397	5.146
mape99	16.057	29.007	71.339	19.130	17.472	3.885	3.245
rmse	0.541	0.591	0.264	1.148	1.245	1.663	1.856
rmse95	0.993	1.250	0.870	2.197	1.738	1.613	1.543
rmse99	1.410	1.521	1.622	2.576	1.946	1.302	1.124
wrmse	1.631	1.937	5.334	2.036	2.053	2.361	2.457

Table 5.1: Results of Gradient Boosting models.

5.1 Visualization of Estimated Demand

Towing Equipment

Towing equipment can be assigned to multiple tasks that all require a different approach of modeling (section 4.2.3). Figures 5.1 and 5.2 show the estimation of the demand for towing tasks for towing trucks AM500 and AM210 respectively. Figure 5.3 shows the demand for overflow towing tasks for towing truck AM210. The demand for overflow towing tasks for the AM500 towing truck is very sparse and therefore it is almost impossible to model this demand using a machine learning approach.

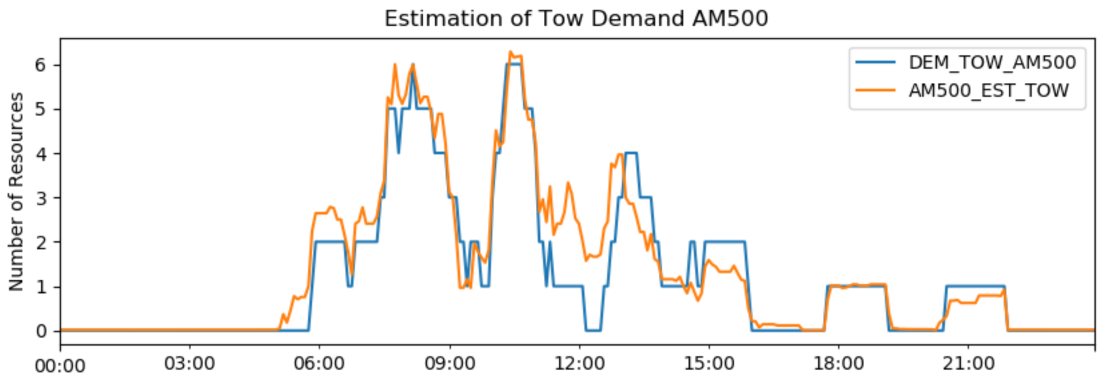


FIGURE 5.1: Result of Gradient Boosting model (AM500_EST_TOW) that estimates the demand for towing tasks planned to be carried out by an AM500 towing truck as calculated by GS (DEM_TOW_AM500).

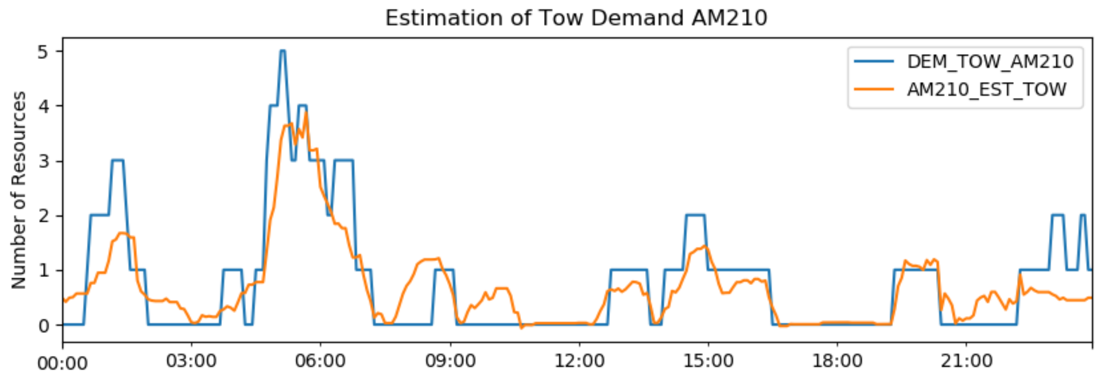


FIGURE 5.2: Result of Gradient Boosting model (AM210_EST_TOW) that estimates the demand for towing tasks planned to be carried out by an AM210 towing truck as calculated by GS (DEM_TOW_AM210).

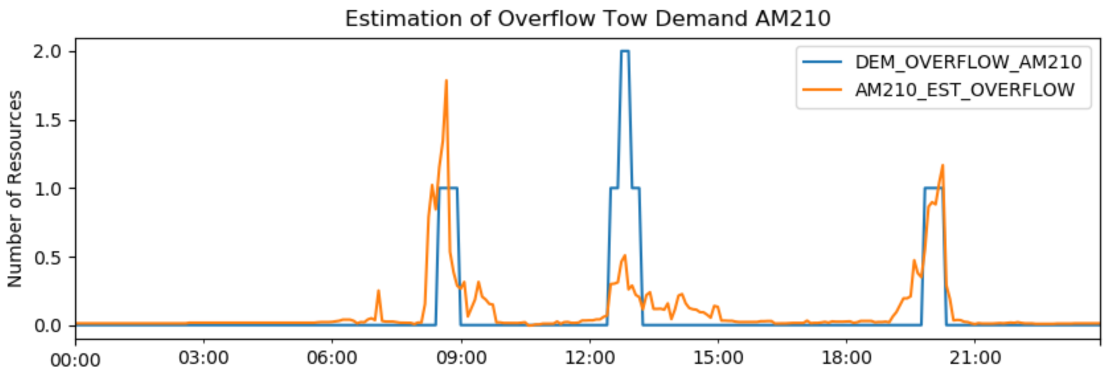


FIGURE 5.3: Result of Gradient Boosting model (AM210_EST_OVERFLOW) that estimates the demand for overflow towing tasks planned to be carried out by an AM210 towing truck as calculated by GS (DEM_OVERFLOW_AM210).

These towing tasks form the biggest part of uncertainty in the demand for these resources. The demand for push backs or static tasks for these resources as determined by GS does not require any machine learning techniques in order to achieve a good estimation. These tasks can relatively easily be calculated and the predictive error for these components should therefore be zero if the PUGs are implemented correctly. Figures 5.4 and 5.5 show the results of the estimation of the total demand for the AM500 and AM210 equipment. These figures also show the calculated demand for push backs and static tasks, which are almost certainly included in the total demand of GS.

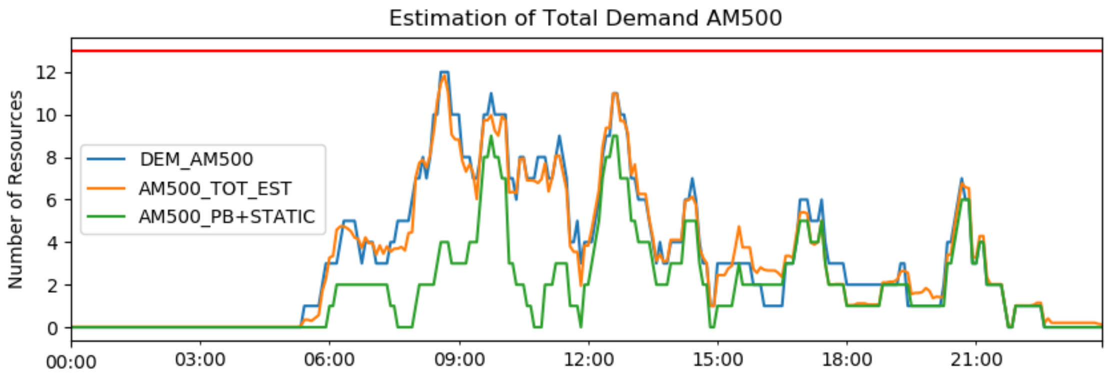


FIGURE 5.4: Total demand for AM500 towing truck equipment according to GS (`DEM_AM500`) with the estimation of this total demand (`AM500_TOT_EST`) and the part of the total estimation that is almost certainly included in the GS planning (`AM500_PB+STATIC`).

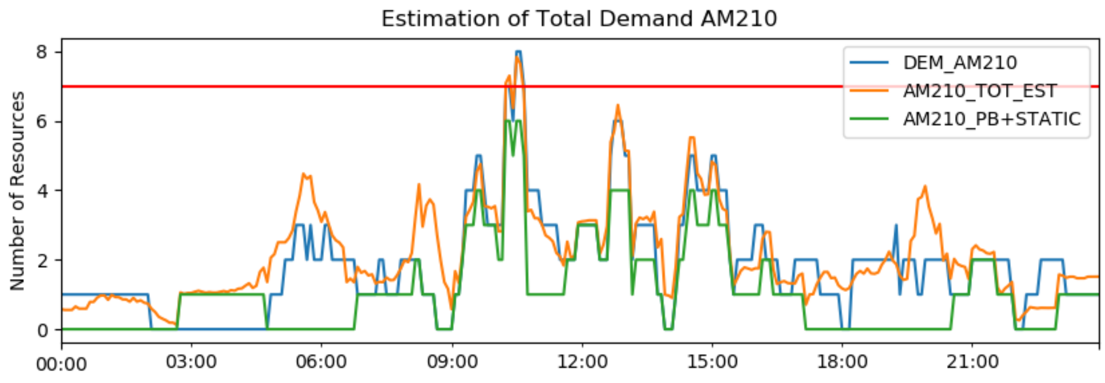


FIGURE 5.5: Total demand for AM210 towing truck equipment according to GS (`DEM_AM210`) with the estimation of this total demand (`AM210_TOT_EST`) and the part of the total estimation that is almost certainly included in the GS planning (`AM210_PB+STATIC`).

The total demand for towing equipment AM110 is much easier to determine since this equipment only carries out push back tasks. Figure 5.6 shows the demand for this resource as calculated by GS and as estimated by the methods as described in this study. It is clear that exactly the same demand should be found if the business rules that are used coincide.

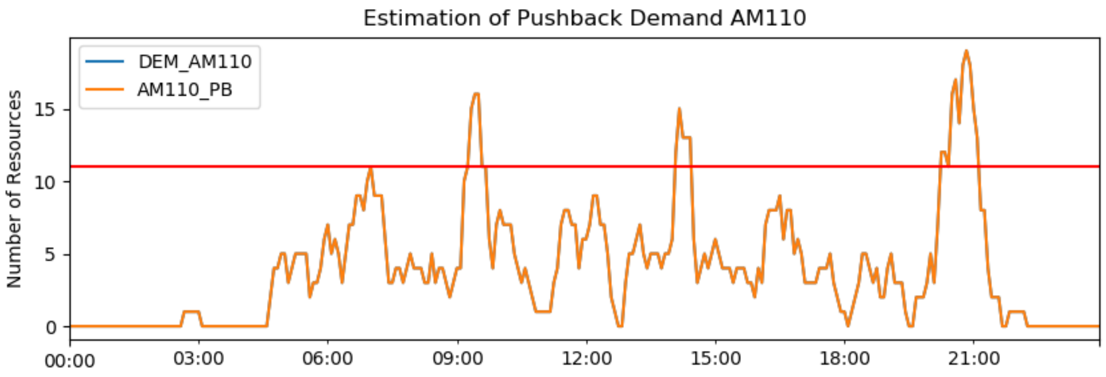


FIGURE 5.6: Result of calculating the demand for tow truck AM110 (`AM110_PB`) compared to the demand as calculated by GS (`DEM_AM110`). These two should be exactly the same when the PUGs are implemented correctly.

Fuel Equipment

The demand for fuel equipment is estimated by two approaches as described in section 4.6.1. The result of the models that try to estimate the sum of the demand for both fuel tasks and their travel time is shown in figures 5.7 and 5.9. The results of the models that estimated these two components separately and summed the results for the same day are shown in figure 5.8 and 5.10.

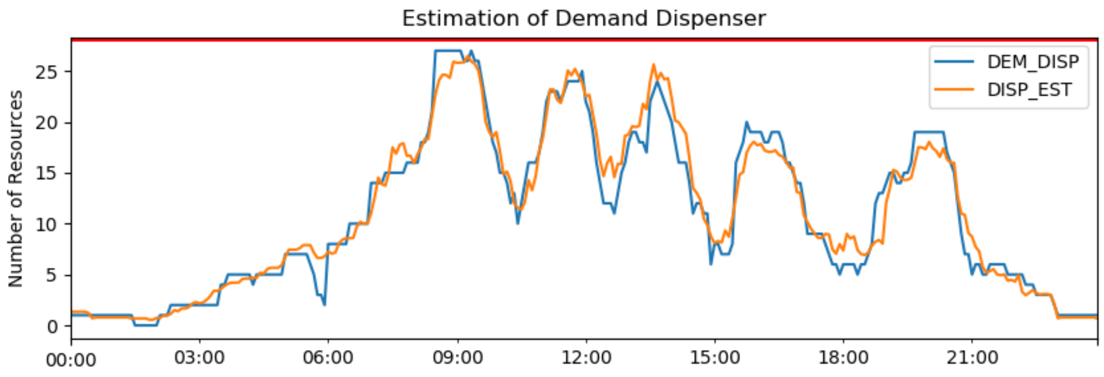


FIGURE 5.7: Total demand for Dispensers according to GS (`DEM_DISP`) with estimation (`DISP_EST`).

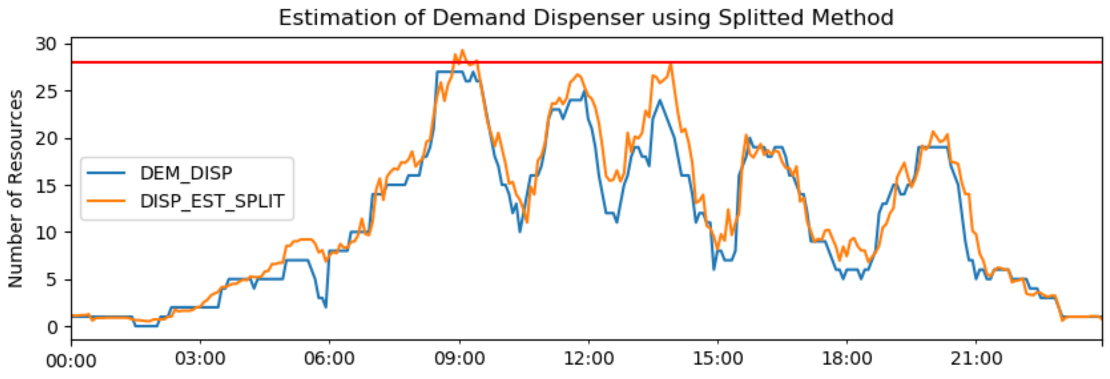


FIGURE 5.8: Total demand for Dispensers according to GS (`DEM_DISP`) with estimation using the splitted method (`DISP_EST_SPLIT`).

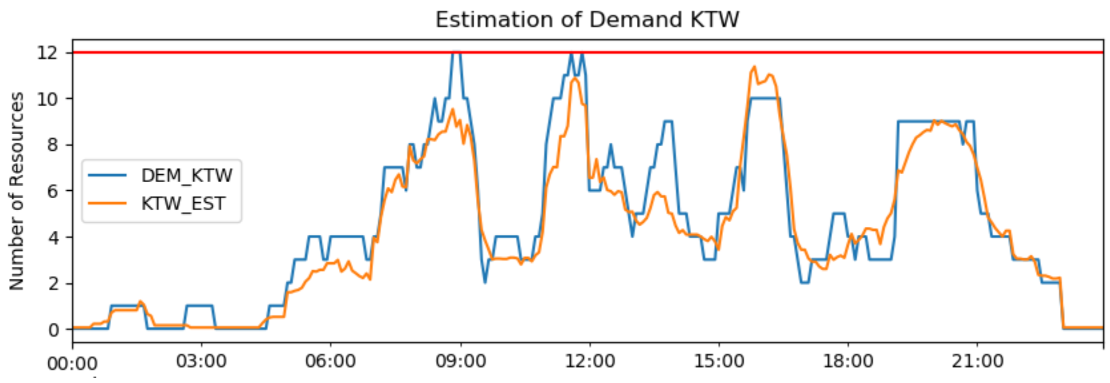


FIGURE 5.9: Total demand for KTWs according to GS (`DEM_KTW`) with estimation (`KTW_EST`).

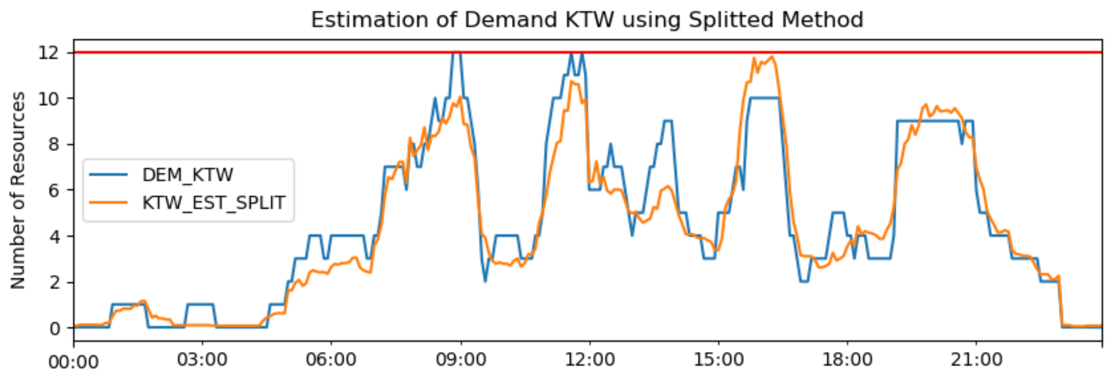


FIGURE 5.10: Total demand for Dispensers according to GS (`DEM_KTW`) with estimation using the splitted method (`KTW_EST_SPLIT`).

Baggage Loaders

As described in section 4.2.2, baggage loading equipment (rampsnakes and powerstows) will always have tasks assigned to them that cannot be shifted. These tasks always have to be planned, unlike towing tasks that only have to be planned when there are not enough gates available. Therefore, the demand for these resources can be exactly determined when the correct business rules are used, just like the demand for tow truck AM110 in figure 5.6.
Figures of the demand for rampsnakes and powerstows are not included since it will only show that the demand according to GS-TP and as estimated by the implementations of PUGs as described in this thesis will be exactly the same.

Examples of Poor Performance

The following figures (5.11, 5.12, 5.13, 5.14, 5.15 and 5.16) contain some examples of days where the obtained models show poor performance.

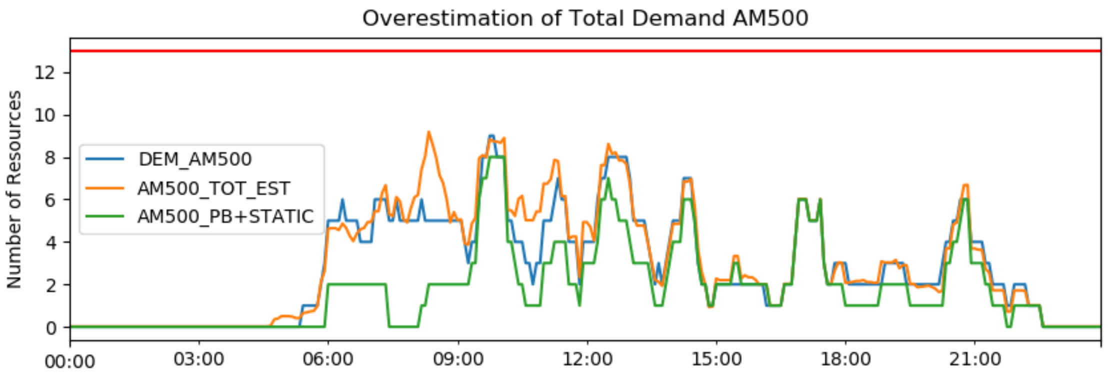


FIGURE 5.11: Total demand for AM500 towing truck equipment according to GS (DEM_AM500) with an overestimation of this total demand (AM500_TOT_EST) and the part of the total estimation that is almost certainly included in the GS planning (AM500_PB+STATIC).

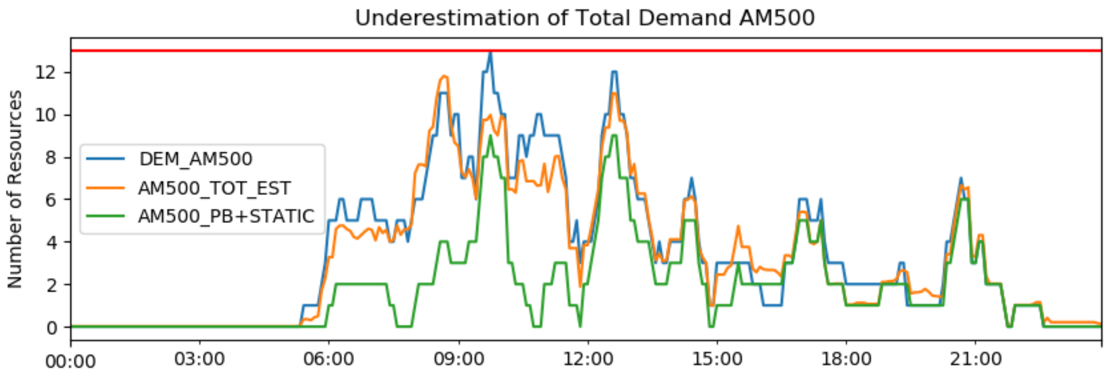


FIGURE 5.12: Total demand for AM500 towing truck equipment according to GS (DEM_AM500) with an underestimation of this total demand (AM500_TOT_EST) and the part of the total estimation that is almost certainly included in the GS planning (AM500_PB+STATIC).

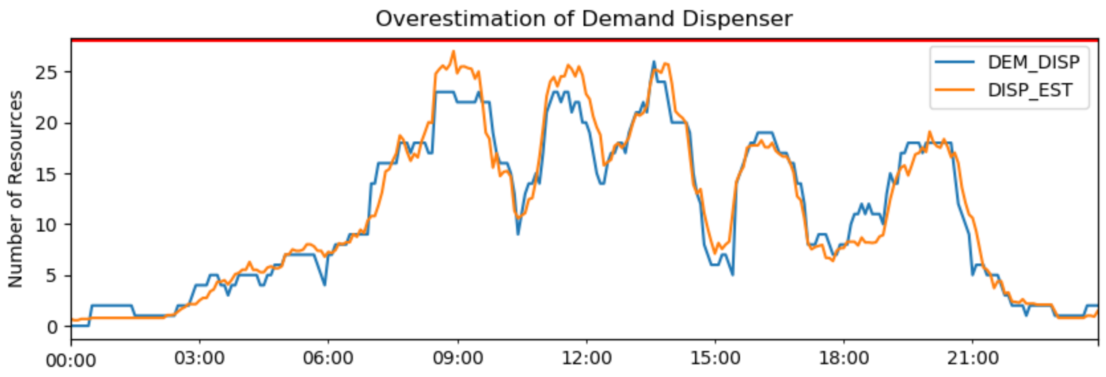


FIGURE 5.13: Total demand for Dispensers according to GS (DEM_DISP) with overestimation (DISP_EST).

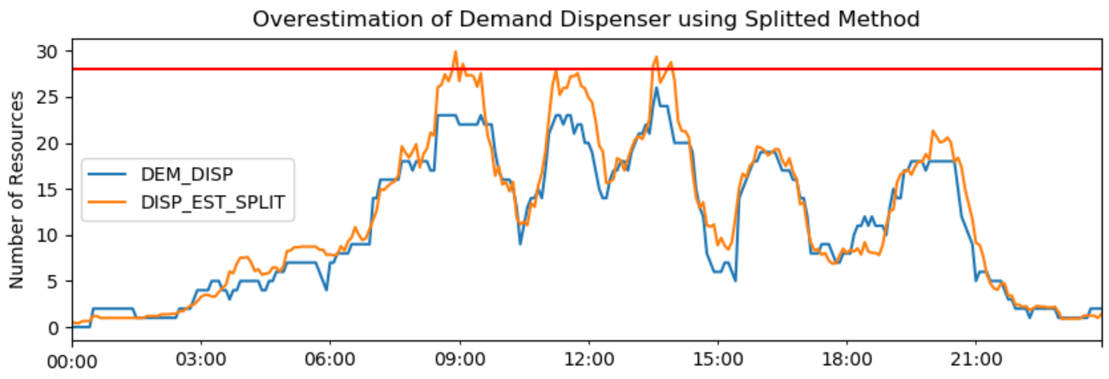


FIGURE 5.14: Total demand for Dispensers according to GS (DEM_DISP) with overestimation using the splitted method (DISP_EST_SPLIT).

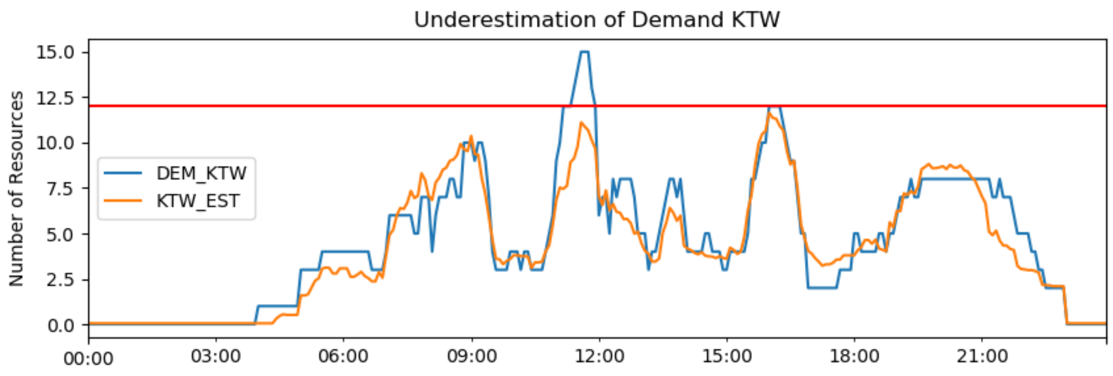


FIGURE 5.15: Total demand for KTWs according to GS (DEM_KTW) with underestimation (KTW_EST).

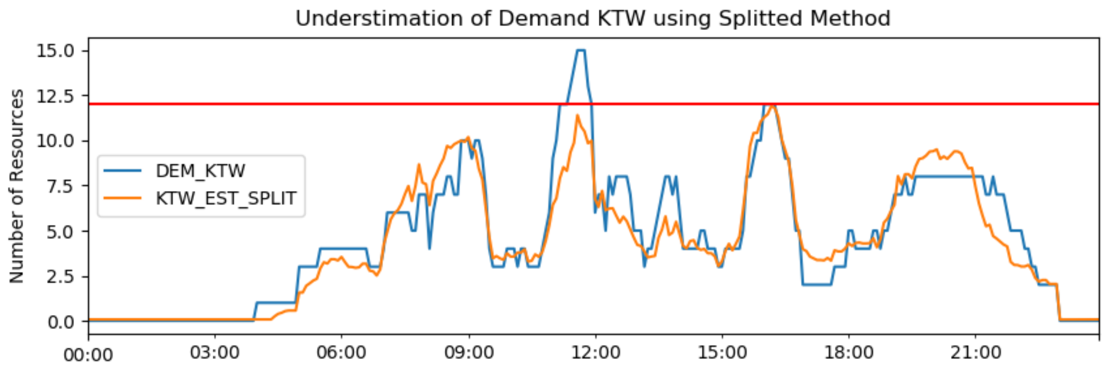


FIGURE 5.16: Total demand for Dispensers according to GS (DEM_KTW) with underestimation using the splitted method (KTW_EST_SPLIT).

5.2 Lists of Generated Tasks

The plots that are shown in the previous section can be used to quickly visualize the estimated demand according to the gradient boosting models. This can be used to find moments for which the demand for ground resources might exceed its capacity. For example, figure 5.17 is a copy of figure 5.5 and shows that the machine learning models indicate a capacity violation of the AM210 equipment.

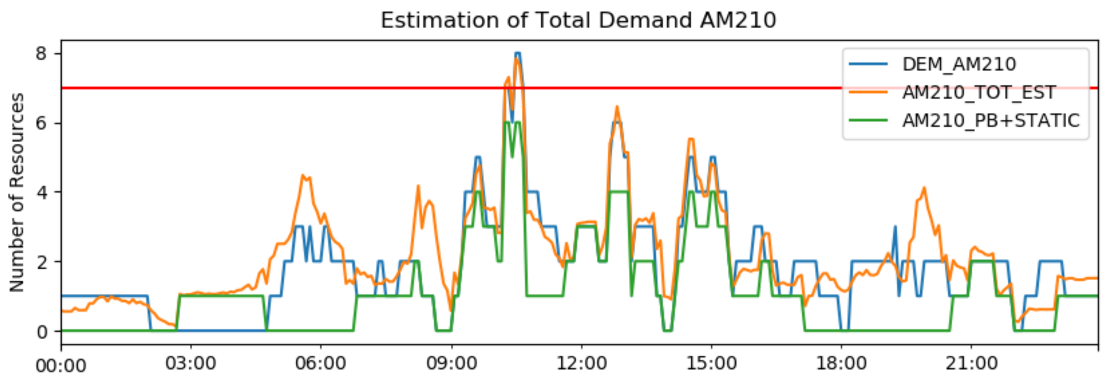


FIGURE 5.17: Total demand for AM210 towing truck equipment according to GS (`DEM_AM210`) with the estimation of this total demand (`AM210_TOT_EST`) and the part of the total estimation that is almost certainly included in the GS planning (`AM210_PB+STATIC`).

Table 5.2 lists all tasks that are generated for the time interval where the demand for AM210 exceeds its capacity. Note that all tasks are listed that could possibly be performed. For push backs it is known that they always have to be carried out. For tows, both a task for a tow to Schiphol East as well as a normal tow task are included. Only one of these has to be carried out when there are not enough gates available. In figure 5.17 it can be seen that the model `AM210_TOT_EST` expects that these tasks have to be carried out. In this case, the model is correct since `DEM_AM210` shows indeed a demand that is higher than `AM210_PB+STATIC`.
Network can use this table in order to take action to decrease the demand for the AM210 for this time interval. It is clear that a lot of this demand comes from Delta (DL) flights that need a push back and Transavia (HV) flights that need to be towed from a parking position to a gate. The only thing KLM can do in this case is to reschedule flight KL0535 that requires a push back from 10:19 to 10:41. The question is whether KLM wants to reschedule flights when other airlines are responsible for a resource capacity violation.

Task	**StartTime**	**EndTime**	**TaskDur**	**FltFrom**	**FltTo**	**ArrDatetime**	**DepDatetime**	**Org**	**Des**	**AC**
PB	09:59	10:21	22	KL0422	KL0723	30/08/2017 07:15	30/08/2017 10:15	DMM	HAV	332
TOW_EAST	10:12	11:21	58	HV5564	HV6331	29/08/2017 15:35	30/08/2017 12:05	KTW	VLC	73H
PB	10:14	10:36	22	DL0072	DL0161	30/08/2017 08:30	30/08/2017 10:30	ATL	MSP	333
TOW_NORM	10:15	11:16	50	HV5564	HV6331	29/08/2017 15:35	30/08/2017 12:05	KTW	VLC	73H
PB	10:19	10:41	22	KL0724	KL0535	30/08/2017 08:05	30/08/2017 10:35	HAV	KGL	332
PB	10:19	10:41	22	DL0134	DL0135	30/08/2017 08:10	30/08/2017 10:35	DTW	DTW	332
PB	10:19	10:41	22	DL0160	DL0125	30/08/2017 08:30	30/08/2017 10:35	MSP	BOS	333
PB	10:19	10:41	22	DL0178	DL0073	30/08/2017 08:30	30/08/2017 10:35	PDX	ATL	333
TOW_EAST	10:22	11:31	58	HV5216	HV5739	29/08/2017 18:55	30/08/2017 12:15	CTA	SOF	73W
TOW_NORM	10:25	11:26	50	HV5216	HV5739	29/08/2017 18:55	30/08/2017 12:15	CTA	SOF	73W
PB	10:34	10:56	22	DL0142	DL0049	30/08/2017 08:50	30/08/2017 10:50	SEA	JFK	333

TABLE 5.2: Example of a list of tasks for AM210. Some task specific information is specified as well as some information about the flights corresponding to these tasks.

Chapter 6

Discussion and Conclusions

This chapter contains the discussion (section 6.1) and conclusions (section 6.2) of the results of *Part I* that were presented in chapter 5. Furthermore, recommendations for the usage of these results for KLM and future studies are given in section 6.3.

6.1 Discussion

The discussion of the results to use machine learning models in order to create a shortcut for extensive resource planning starts with debating the performance of these models in section 6.1.1. Then section 6.1.2 contains a discussion of the continuity of implementing these models. Finally, section 6.1.3 discusses how Network's capacity planners can use the models to react on possible resource capacity violations.

6.1.1 Model Performance

In general the gradient boosting method seems to be quite able to estimate the demand as calculated by ground services for average schedules. Though the models show some examples of bad performance, they can still be used as an indication for the ground research demand.

Towing Equipment

The estimation for towing equipment follows the trends of the demand for tow tasks as calculated by GS-TP - see figures 5.1 and 5.2 - and most of the time the absolute error is not more than 1 resource. The demand for overflow tasks seems to be harder to estimate (figure 5.3). This is probably caused by the sparsity of this demand but, because of this sparsity, it might not be a big problem that the estimation of this demand is not that good since there is only a demand of 1 or 2 resources for overflow towing every once in a while.
Figures 5.4, 5.5 and 5.6 show that the total demand for towing equipment can be quite well estimated by gradient boosting models. Figure 5.5 would have correctly alarmed the Network department that more resources are needed than available according to GS-TP planning process

after which table 5.2 could be used to explore possible schedule changes in order to decrease this demand. A helpful aspect of these graphs is that the demand of which Network can be almost sure that it will be planned by GS-TP plannings process (green line) is shown. From figure 5.12 it becomes clear that sometime the model is that far off that a demand peak is not shown by the estimation at all. This is a flaw of the model and such differences will question the credibility of these models.

Fuel Equipment

For estimating the demand for fuel equipment, both the regular and splitted approach show a higher absolute error than the tow equipment models (table 5.1). Figures 5.7 and 5.8 show that the splitted approach seems to overestimate the demand, while the model that estimates the total demand for dispensers directly tends to underestimate at some points. For KTWs (figures 5.9 and 5.10) this effect is much smaller. Though the demand seems to be higher according to the splitted method, both models underestimate the first two peaks and overestimate the third peak.
Figures 5.13 and 5.14 show an example of a day where both the splitted and regular model overestimate the demand for dispensers by a large amount. A negative consequence could be that Network reschedules flights according to this estimation while initially there was no resource capacity problem at all. Figures 5.15 and 5.16 show a regular occurring underestimation of the KTW models. This means that if Network decides to consult the gradient boosting models, they might conclude that there is no resource capacity problem at all.
The demand for fuel resources seems to be harder to predict than the demand for tow trucks. The reason for this might be that the demand for fuel trucks is more dependent of a gate planning and the model does not take a gate planning as input. Also, the demand for fuel resources considers travel times according to a travel time matrix. Though this is a more realistic method to plan these resources, it is harder for the machine learning models to estimate its result.

Baggage Loaders

For rampsnakes and powerstows the demand as calculated by GS-TP can be precisely recalculated since these resources are planned with only fixed tasks. So, if the business rules are implemented correctly, Network's capacity planners will know exactly what the planned demand for these resources will be and can therefore adequately respond to a capacity violation.

6.1.2 Continuity of the Implementation of ML Models

Although the machine learning models seem to perform well enough to be used by the Network department as an indication for the ground resource demand, it might be far off after changes in GS-TP's planning process or infrastructural changes on Schiphol. Therefore, the continuity of the implementation of the ML models is discussed in this section.
When GS-TP planning rules (PUGs) change, the implementation of these rules that is used for the machine learning models has to be updated. It is expected that small changes in the planning process, like a change in task duration or planned start time, will not have that much of an influence on the performance since this information is included in the models. More structural

changes in the planning process, like changes in the gate planning tool, will most likely cause the models to show bad performance. This information is not used as an input for the model and therefore is ingrained in the decisions that the gradient boosting algorithms make.
For infrastructural changes on Schiphol the same problem occurs. This information is not given to the gradient boosting algorithm simply because this information is the same for all available data. If this information does not differ for some data sample points, it cannot be used to distinguish between the outcome of these data points. If at some point, for example, more gates become available, the models will be unaware.
There are two solutions to the problems of structural process changes: the models can be trained on example data that contains structural changes or the models have to be retrained every time a structural process change occurs. The first one should produce models that are robust for process changes. However, these process changes do not occur regularly and there will not be enough data available in order to train reliable models. Therefore the latter solution is favourable.

6.1.3 How can Network's capacity planners be advised to reschedule?

Most of *Part I* of this thesis is focused on answering research question 1: whether it is possible to estimate the resource demand as calculated by GS-TP. Research question 2, that asks how Network's capacity planners can be advised to alter the flight schedule if it is likely to be operationally infeasible, has been less discussed.
In the current process, when the feedback of GS-TP shows a capacity issue, the capacity planners reschedule the flights they think cause the problem. They are aware of what resources are used to handle each aircraft type, but the amount of rescheduling of the flights that are flown by these types is just a good guess. The machine learning models that estimate the total resource demand for the most critical resources can already show an indication of the effect of these good guesses. The planners can use the model's output in an iterative process of altering the flight schedule and analyzing the effect on the resource demand.
Furthermore, the process of creating the schedule characteristics that form the input for the machine learning models (figure 4.1) require that GS-TP's business rules are assigned to turnarounds in the schedule. The result of this assignment is an extensive list of (possible) tasks that are planned by GS-TP (see table 5.2 for an example of such a list). These lists can guide the capacity planners towards a good rescheduling decision. Therefore, it can be said that the methods used to answer research question 1 already provides a solution for research question 2.

Local Search Solver

Attempts to build a local search solver that reschedules flights to make a schedule operationally feasible, with the objective of rescheduling the least amount of flights possible, have been made. This is a tricky approach since the gradient boosting methods probably do not perform well enough to be used as a method to evaluate each candidate solution. Furthermore, there is a lot of information regarding Network scheduling that is very hard to include in such a solver. For example, flights to some specific airport might never be rescheduled because a lot of transfer possibilities might be lost. After conversations with KLM's Network schedulers it became clear that a local search solver will not be able to outperform nor overrule the schedulers decision. It might however be possible to use such a model to support the schedule planners in their decision making.

6.2 Conclusion

Part I of this thesis focuses on answering the following resource questions:

1. What is the relation between a flight schedule and the resource demand for the most critical ground resources as calculated by Ground Services - Tactical Planning and is it possible to quickly verify whether a flight schedule will be accepted after an operational check?
2. Given that a schedule will not be accepted by Ground Services - Tactical Planning, how can the Network capacity planners be advised to alter the schedule such that it increases the probability of acceptation?

It has been shown that the planning rules (PUGs), that GS-TP uses to schedule their resources, can be used to give a quick indication of the possible tasks for ground equipment for a new schedule without the realization of a gate planning.
The implementation of these PUGs and other characteristics of the flight schedule can then be used to quantify a flight schedule. This quantification, in the form of “schedule characteristics” can be used as an input for machine learning methods that are proven to be able to find the relation between a flight schedule and the resource demand as calculated by GS-TP. Therefore it is possible to quickly give an estimation of whether a schedule is likely to be accepted by an operational check (research question 1). For some types of tasks it is not even necessary to rely on machine learning models, the demand as calculated by GS-TP follows directly from the implementation of the PUGs.
The resource demand estimation can already be used as an advice for Network’s capacity planners on how to alter a schedule when it is operationally infeasible (research question 2). Furthermore, the capacity planners have access to more information in the form of the implementation of PUGs that can generate a list of tasks or possible tasks for every type of equipment (see table 5.2).
Though the machine learning models, as described in *Part I* of this thesis, do not show a constant reliable performance for all resource types, they can still be used by planners to give an indication of the demand. The added value of the machine learning models is that within a couple of minutes, without creating a gate allocation, an extensive planning process that takes a full night to fulfill can be estimated. This allows KLM’s Network capacity planners to quickly evaluate a schedule in terms of operational feasibility and to adequately react on resource capacity violations.

6.3 Recommendations

It is shown that the resource demand as calculated by Ground Services - Tactical Planning can be estimated by assigning their business rules to a flight schedule and applying the machine

learning technique Gradient Boosting. This can be done without the use of a gate allocation planning tool. It is recommended that KLM's Network department uses these models in order to gain insight in the GS-TP planning process and to be able to adequately alter a schedule in the case of operational feasibility. Although the machine learning models failed to show a consistent performance for some resources, they still add information at the network scheduling phase.
Some resources do not require a machine learning model since the demand according GS-TP can easily be calculated with a 100% accuracy if their PUGs are implemented correctly. For other resources, the same statement holds for a significant part of the total demand. The lists of ground tasks that can be generated can support Network's capacity planners in rescheduling flights. Furthermore, these lists also give insight in the expected number of resources. Therefore it is recommended to at least implement the GS-TP plannings rules at Networks site. They will provide a lot of information even without the usage of machine learning models.

Future Studies

In order to improve the machine learning models, future studies can include that of ensemble methods in order to be able to provide a better estimation of the resource demand as calculated by GS-TP. An ensemble method takes a weighted vote of predictions of multiple classifiers [19]. In its essence, Gradient Boosting is already an ensemble method, but it can be combined with other models, that might be better in finding some specific relations, in order to improve its performance.
Another interesting application of machine learning is the usage of convolutional neural networks. These models are often used for image recognition and the idea is to move a frame of a couple of pixels high and a couple of pixels wide over an image [20]. The network learns to recognize patterns in this frame and gives meaning to the relative ordering of these patterns over the image. In the same way such a "frame" of a couple of minutes wide can move over the time span of a day of a flight schedule to find patterns in this frame. These patterns can be combined in another layer of the network in order to achieve an estimator for ground resource demand.

Part II
Indication of Ground Resource Demand on the Day of Operation

Given a flight schedule, what can be said about the demand for ground resources on the day of operation?

A Simulated Approach

Chapter 7

Methods

This chapter describes the methods that are used in order to model the demand for ground resources on the day of operation using simulation. Existing flight simulation software OPiuM, that has been developed by KLM internally, is used to produce multiple possible flight schedules on the day of operation. These "operational" schedules can be evaluated using the machine learning (ML) models as described in *Part I* of this thesis report (figure 7.1). The result will be an indication of how the demand for ground resources is distributed on the day of operation.

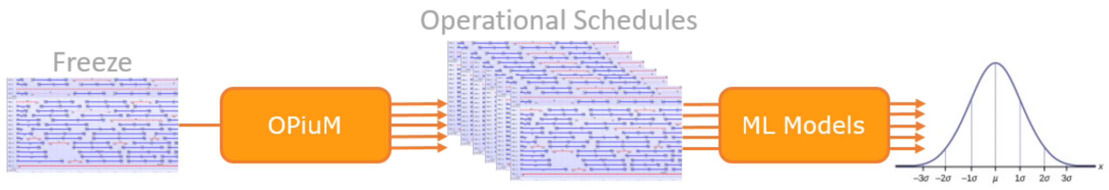


FIGURE 7.1: OPiuM can be used to simulate multiple "operational" schedules. Applying the ML models as described in *Part I* on each "operational" schedule can give a distribution of the expected resource demand.

Section 7.1 describes the motivation for this simulated approach. The simulation software OPiuM is explained in section 7.2. Then, section 7.3 dives more in the way OPiuM can be used in combination with ML models in order to determine resource demand on the day of operation. Finally, section 7.4 discusses the validation of this method.

7.1 Motivation

The initial question that was asked by GS-TP was whether it was possible to generate an average "operational" schedule that includes operational uncertainties such as flight delays. This is hard to achieve since it asks for a deterministic representation of stochastic events. This can be compared with the average number of eyes of the role of a single die. On average this is 3.5 but the result will never be observed when a die is rolled.
The simulated approach uses existing simulation - that has already been used and validated by KLM - to generate multiple of these "operational" schedules. ML models that are built to

try to achieve the same result as GS-TP's planning process will then be used to evaluate these schedules. So instead of generating an average schedule for which GS-TP can determine the output in the form of resource demand, multiple "operational" schedules are generated that are evaluated with an estimation of their extensive planning process.
This approach is a combination of existing simulation tooling and the result of *Part I*: ML models that can quickly estimate resource demand as calculated by GS-TP. Without the ML models it would not have been possible to evaluate multiple "operational" schedules in a short amount of time.

7.2 OPiuM Schedule Simulation

OPiuM is a simulation model that is used to evaluate the performance and robustness of a proposed flight schedule. It breaks all activities that are carried out by an aircraft in so called building blocks. The length of each building block is simulated according to a realistic probability distribution of the duration of the activities corresponding to this building block. If the simulated duration of an activity causes a schedule disruption - for example, the departure handling finishes after the succeeding flight is scheduled to depart - this disruption is fixed.

7.2.1 Building Blocks

Building blocks are an interpretation of the sequence of tasks that has to be carried out by a single aircraft (figure 7.2). These building blocks can now be seen as separate processes of flight schedule operations.

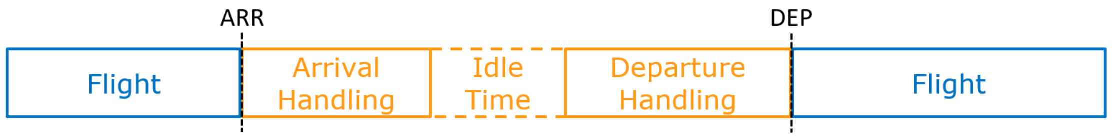


FIGURE 7.2: Building blocks are used by OPiuM to separate processes of regarding flight operation.

Each flight is considered as one single process. All arrival and departure related ground processes are grouped in arrival and departure handling building blocks. When the duration between arrival and departure is relatively short, arrival and departure handling processes intertwine. In those cases, arrival and departure handling is modelled as one single building block.

7.2.2 Process Probability Curves

In order to simulate the duration of each building block, OPiuM uses cumulative probability curves that are derived from operational data. These so called "OPiuM curves" or "S-curves" are described in the form of percentiles of the duration of a building block. These curves can be defined for subgroups of aircraft type, origin, destination and validity periods (table 7.1). Some examples of the percentiles for these subgroups are shown in table 7.2. For every building block in a given flight schedule, the duration of the process corresponding to this building block is drawn from the probability distribution described by these percentiles.

For each process, the curve that matches the characteristics of the process the most is used. For example, for a departure handling building block on Schiphol (AMS) of an Airbus 332 that is about to fly to destination ACC (Kotoka International Airport, Accra, Ghana), the curve with CurveId 2 from table 7.1 should be used. When a curve that matches on that level of detail cannot be found a more general curve, like the curve with CurveId 1, is used. Sometimes, a specific process matches multiple curves. In that case the curve with Time AMS (minutes from midnight) is used that is closest to the start time of the process.

For arrival and departure handling curves, the Standard Processing Time (SPT) is given. This indicates the standard duration of these processes. The curves with ids 4 and 5 show that the duration of similar processes can have slightly different curves for different seasons. CurveIds 4 and 6 show that the direction of a flight might require a different probability distribution.

CurveId	ValidFrom	ValidTo	Block	Subtype	Origin	Destination	Station	SPT	Time AMS
1	26-Mar-17	04-Apr-17	DEP	332			AMS	84	
2	26-Mar-17	04-Apr-17	DEP	332	AMS	ACC	AMS	84	
3	26-Mar-17	04-Apr-17	DEP	332	AMS	ALA	AMS	84	
4	30-Oct-16	25-Mar-17	Flight	332	ACC	AMS			290
5	26-Mar-17	28-Oct-17	Flight	332	ACC	AMS			275
6	29-Oct-17	24-Mar-18	Flight	332	AMS	ACC			805

Table 7.1: Examples of subgroups of OPiuM curves that are used to simulate the duration of building blocks. The percentiles of the duration of the processes for these subgroups are given in table 7.2.

CurveId	P10	P20	P30	P40	P50	P60	P70	P80	P90	P95	P98
1	68	74	77	79	81	84	86	90	96	105	113
2	76	79	81	82	84	85	88	92	100	121	153
3	65	70	74	76	77	78	81	83	87	94	107
4	394	400	405	408	411	414	418	421	428	433	438
5	397	402	405	407	410	413	415	421	425	432	440
6	391	396	401	405	407	410	413	419	425	430	434

Table 7.2: Examples of percentiles of OPiuM curves that are used to simulate the duration of building blocks. A description of the subgroups on which these curves apply is given in table 7.1.

Figure 7.3 shows two OPiuM-curves from which the duration of building blocks is drawn. This is done by randomly selecting a probability p between 0 and 1. The x-value of the intersection between $y = p$ and the curve is then used as the duration of the process.

7.2.3 Disruption Decision Management

When the duration of flights and ground processes are drawn from an OPiuM-curve, the initial flight schedule might be disrupted. OPiuM checks the simulated schedule every 15 simulated minutes for such disruptions. Whenever a disruption occurs, e.g. a departure process has not been finished before a planned departure, OPiuM can take one of the following actions in order to solve this disruption:

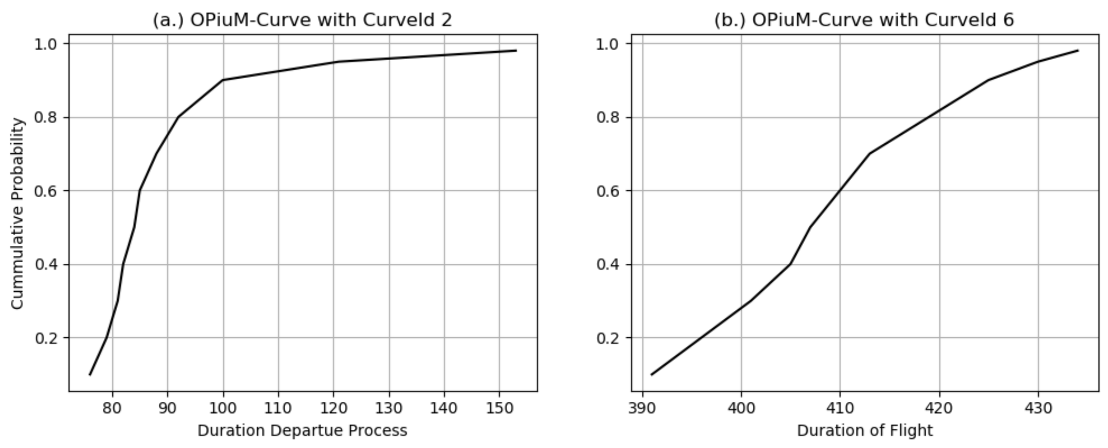


FIGURE 7.3: Example of OPiuM-curves corresponding to (a.) duration of departure process of an Airbus 332 on Schiphol (AMS) that is about to fly to ACC, and (b.) duration of a flight operated by an Airbus 332 from AMS to ACC.

- Delay Flight.
- Swap Flights, i.e. two aircraft switch flights they were initially assigned to.
- Break Maintenance, i.e. stop current maintenance in order to make an aircraft available for operation.
- Cancel Flight.

Each sanction generates some kind of cost. This cost is used by OPiuM to decide what action is best to take.

7.2.4 Shortcomings of OPiuM

One of the shortcomings of OPiuM is that it currently only can simulate the processes of the *wide* and *medium* body fleet of KLM. The processes corresponding to the *narrow* body fleet of KLM Cityhopper and aircraft from other airlines are not included since there has not been an interest in this performance or because necessary data are not yet available. This means that only a part of the flight schedule can be simulated. Because KLM makes up a large portion of the flights on Schiphol airport, and even a larger portion of the turnaround that require the usage of ground resource from GS, it might be enough to simulate just the processes of KLM medium and wide body aircraft to give insight in the resource demand on the day of operation. OPiuM samples processing and flight times for each process independently. Realistically, these processes might be dependent. For example, because of weather conditions a lot of flights that originate from the same region might all be delayed by some amount. This independent way of simulating delays also has the result that resource capacities are not taken into account. If some ground equipment is delayed it might mean that other aircraft have to wait for this resource and will therefore be delayed as well.

7.3 Using OPiuM Simulation to Determine Operational Resource Demand

As shown in figure 7.1, OPiuM can be used to simulate a flight schedule such that multiple possible “operational” schedules arise that can be evaluated by the ML models from *Part I*. This can be done using the standard OPiuM curves as described in the previous section (section 7.2). These curves, however, allow for delays in the ground handling processes. This results in an analysis of the resource demand on the day of operation while the processes underlying this demand might be delayed. A conclusion may be that there are enough resources available while maybe more resources would have been necessary if they would not have been delayed.
Therefore, except from exclusively simulating using the regular OPiuM curves, additional simulations are run that do not allow ground handling delay. To achieve this result, the Standard Processing Time (SPT) is used as a maximum for the duration of ground handling processes. Figure 7.4 a. shows the OPiuM-curve of a departure process without the possibility of having a processing time higher than the SPT. For flight time (figure 7.4 b.) delays are still allowed and therefore the regular OPiuM curve can still be used.
When the ML methods are used to estimate the demand of simulated schedules, the models use information that might not always be available on the day of operation: for the ML models the delay is known at the moment of resource planning. Realistically this is not the case. A delay might occur at the moment a resource is already at an aircraft. This effect is hard to analyze since from an OPiuM output it is not clear what caused the delay.

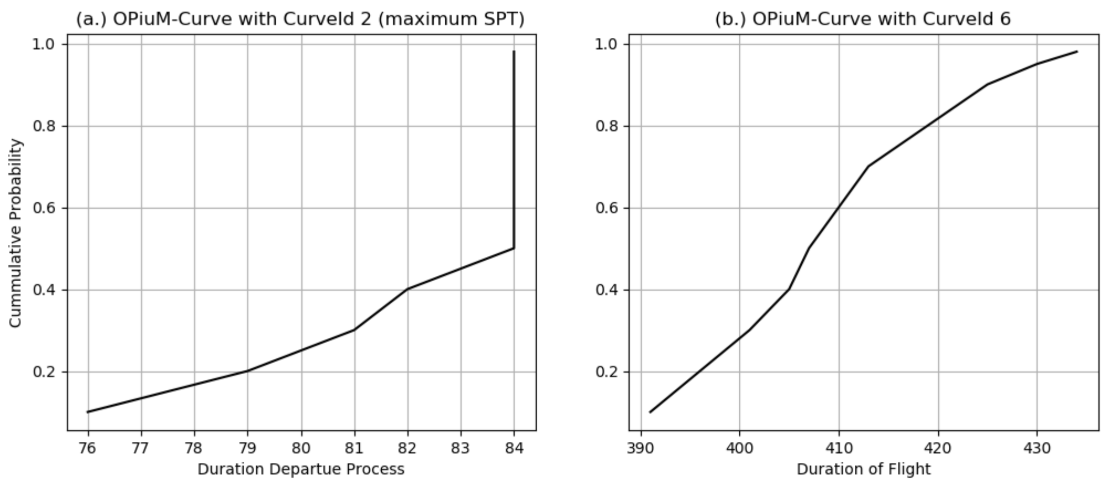


Figure 7.4: Example of OPiuM-curves corresponding to (a.) duration of departure process of an Airbus 332 on Schiphol (AMS) that is about to fly to ACC with a maximum departure process duration of its SPT (= 84), and (b.) duration of a flight operated by an Airbus 332 from AMS to ACC.

7.4 Validation Methods

This section describes the way the simulated approach of retrieving an indication of ground resource demand on the day of operation can be validated. First the validation of the used simulation tool OPiuM is discussed. Then the system that the KLM uses to track the usage of resources on the day of operation named CHIP is introduced and a retrospective way of GS-TP planning is described. Finally the method of validating the output of applying ML models on OPiuM simulations is discussed.

7.4.1 OPiuM Validation

OPiuM is validated for KLM's punctuality performance. This means that the percentage of flights that arrive/depart on time, or at least within 15 minutes of its planned arrival/departure, corresponds to the value of these measures that are observed in operation. This validates that OPiuM can be used for its purpose: to evaluate the performance and robustness of a proposed flight schedule. However, this does not directly imply that this simulation tool can be used to estimate the usage of ground equipment.

7.4.2 CHIP and Ground Equipment Control Room

CHIP is a system that KLM uses to track multiple ground processes. One of its functionalities is that operators of ground equipment can use a handheld to let this system know that they are currently starting or ending a certain task. This information is transferred to planners in the ground resources control room. In this control room decisions are made regarding the usage of equipment on that moment. CHIP also contains an optimizer to reschedule tasks according to delays. Because of a lack of faith in this system, this functionality is not used by the planners in the control room.
Unfortunately, CHIP is still in development and it does not yet track all critical resources that are included in this thesis. Also the resources and tasks are grouped in a different way than GS-TP does. For example, CHIP does not distinguish what truck is used for a tow task. It thus includes all towing trucks, even the ones that are not considered in this thesis.

7.4.3 GS-TP Historic Planning

Ground Services - Tactical Planning has developed planning rules in order to determine the demand for resources of a schedule that has already been flown. This "Historic Planning" is used to give an indication of the demand for resources of a realized schedule. These planning rules take the difference between planned and actual departure and arrival times into account. They assume that, when a small departure delay occurs, the equipment that was already at the aircraft is delayed as well. This might not always be the case since the cause of the delay is not known. For example, a dispenser might be done with its tanking task after which some mechanic finds out about a technical defect. In that case the flight is delayed but the dispenser can already be used to fulfill another task. Though the Historic Planning is just a rough indication that is based on somewhat unrealistic assumptions, it might still be used as a comparison for the simulated resource demand.

7.4.4 Validation of the Simulated Approach to Indicate Resource Demand on the Day of Operation

In order to draw conclusions from the results of this simulated approach, it has to be validated. OPiuM is only validated at an aggregate level and the ML models are only validated on planned schedules (chapter 5). This performance does not tell what the performance will be on simulated "operational" schedules.
CHIP data can be used to show the usage of ground equipment on the day of operation. Because this data depends on the human capability and willingness of tracking its own performance, it is not very reliable for the total resource demand. From a visit at the control room it became clear that most operators of GS equipment simply forget or deny to register their performance on their handheld. Despite this knowledge of the quality of the data, it is still the best actual information. Furthermore, GS-TP historic planning can be used for validation since this resource demand calculation takes known delays into account.
The Rooted Mean Squared Error is presented as a performance metric to indicate the predictive value of the the planned resource demand. Furthermore the probability that the actual demand lies within a certain interval that is generated by the ML models on OPiuM simulation can be used as an indication for the performance of this simulated approach.

Performance Metric

Though there is no reliable operational data and the "historic" planning of GS-TP might be just a retrospective indication, they can both be used to be compared with the result of the resource demand as calculated for simulated schedules. This can point out similar trends and provoke the discussion about the reliability of this simulated approach and its results.
In order to define performance metrics a distinction has to be made between planned demand and actual demand:

Planned Demand	
GS-TP Planned	Resource demand according to GS-TP regular plannings process.
ML on Planned	Resource demand according to Machine Learning models that try to estimate the regular GS-TP planning process.
Mean of ML on OPiuM	Mean of the resource demand according to Machine Learning models on 30 OPiuM outputs.
Actual Demand	
Historic	Demand according to GS-TP retrospective resource planning.
CHIP	Demand according to CHIP registration.

An indication of how well a planned demand ($\hat{y}$) can be used as a forecast for the actual demand (y) can be given by the Rooted Mean Squared Error (7.1).

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \tag{7.1}$$

Confident Interval of ML on OPiuM Output

It can be tested whether actual demand lies in a confidence interval generated by applying the ML learning models on multiple OPiuM simulation runs. Assuming that the demand at a certain 5 minute bracket according ML models that are applied on OPiuM output is normally distributed, the t-test can be used in order to define its 95%-confidence interval. Let the bounds of this interval be described by:

$$\bar{x} \pm t_{(n-1,\alpha)}\frac{s}{\sqrt{n}} \tag{7.2}$$

With n the number of observations, s the standard deviation of the demand and $t_{(n-1,\alpha)}$ the t-statistic corresponding to a student's t-distribution with $n-1$ degrees of freedom and $\alpha = 2.5\%$. A more empirical interval can be defined by taken the 5-th en 95-th percentiles of the resource demand as calculated by ML models on OPiuM estimation. With 30 simulations runs, the lower bound can be set by the 2-nd lowest demand and the upper bound can be given by the 2-nh highest demand for that interval.

Chapter 8

Results

This chapter presents the results of 30 OPiuM simulation runs on the 9th Base Rolling Planning of 2017 (`F17P09`). The goal of this chapter is only to present the results. A discussion can be found in chapter 9. Simulations are run with full OPiuM curves, as well as with OPiuM curves that do not allow ground delays. First, visual results are presented in the form of graphs showing representative sample days. Then, tabular performance metrics - as discussed in section 7.4.4 - are given.
The graphs on the next pages can contain the following information:

1.	GS-TP Planned	(blue line)	Result of GS-TP resource demand on initial schedule.
2.	GS-TP Historic	(green line)	Retrospective "Historic" planning of GS-TP.
3.	ML on Planned	(red line)	Result of ML models on the initial planned schedule.
4.	ML on OPiuM	(grey lines)	Result of ML models on OPiuM simulation runs. These are shown as multiple grey lines.
5.	CHIP	(black line)	Resource demand according to CHIP.

The graphs can contain a lot of information and therefore may look a bit messy. However, in this representation, all information is combined into one single graph, which allows comprehensive analysis.

Towing Equipment

The results of calculations for the demand for AM110 trucks on OPiuM simulation with and without simulated ground delay are shown in figures 8.1 and 8.2 respectively. Note that “ML on Planned” matches the demand as calculated by GS-TP because this resource is only planned by fixed tasks. Unfortunately, CHIP does not group push back tasks on type of equipment.

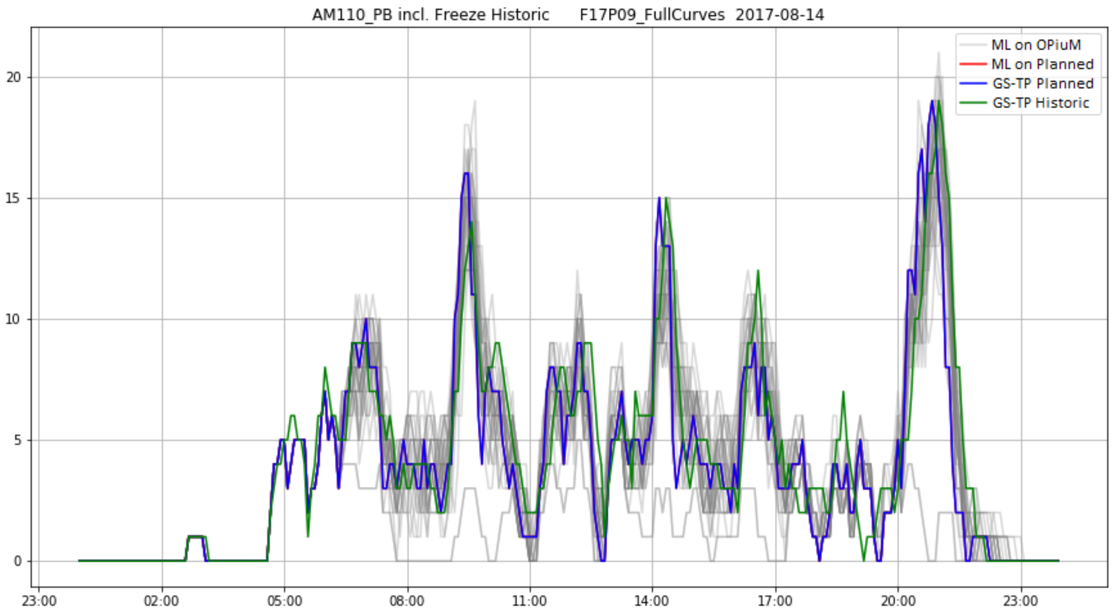


FIGURE 8.1: Result of calculations for PB AM110 tasks on OPiuM simulation using regular OPiuM-curves.

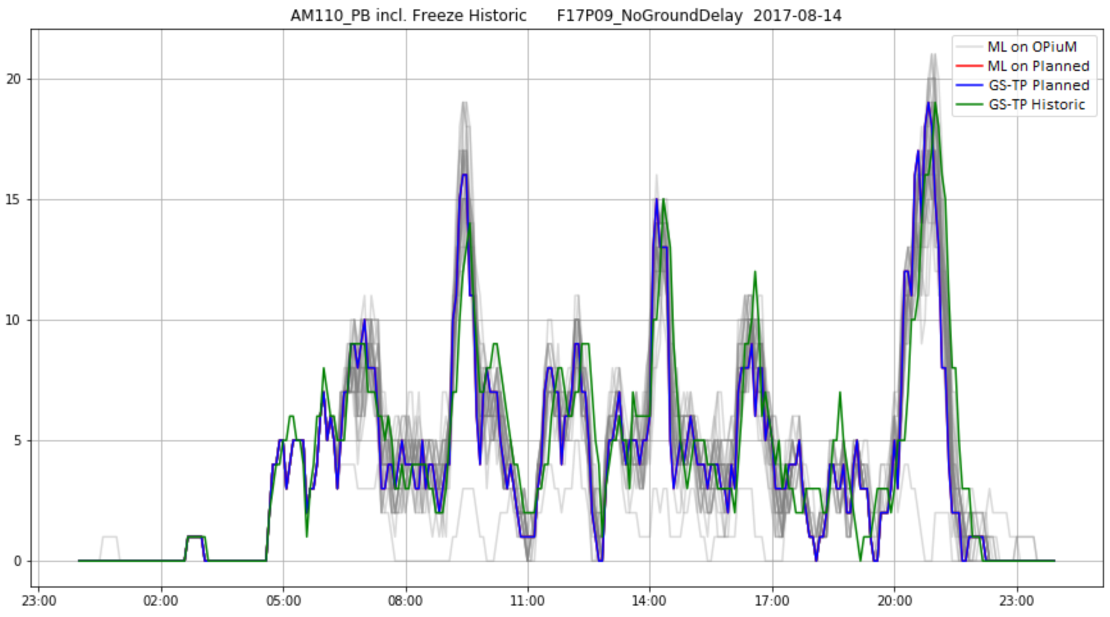


FIGURE 8.2: Result of calculations for PB AM110 tasks on OPiuM simulation using OPiuM-curves without simulated ground delay.

Figures 8.3 and 8.4 show the calculations of the total demand for push backs for all trucks combined. This is done to make comparisons with CHIP data possible because CHIP does not group push back tasks on type of equipment.

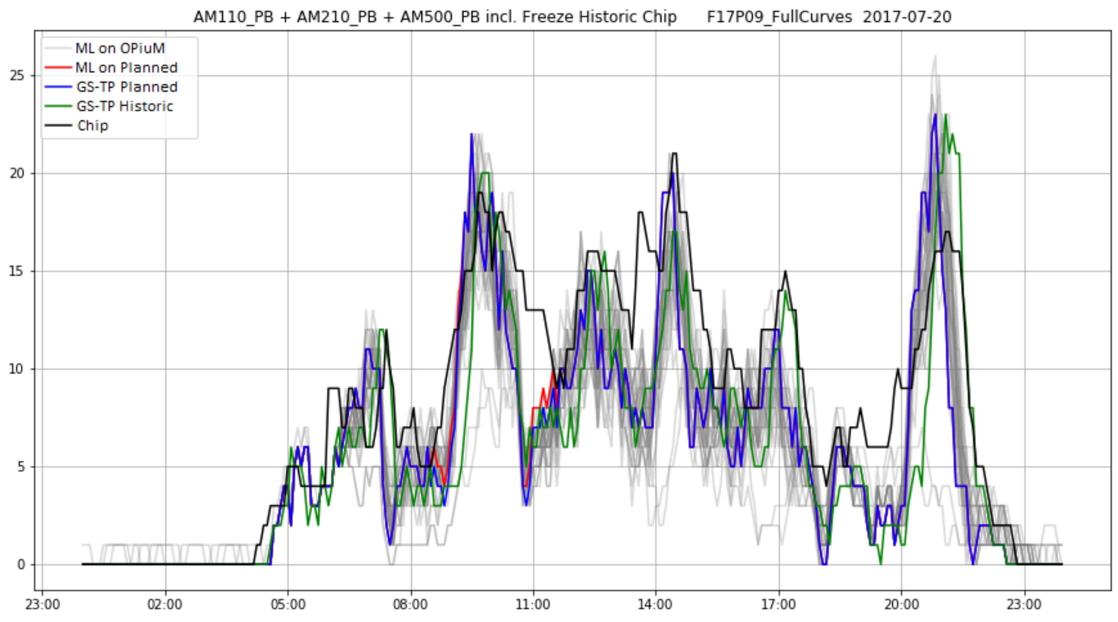


FIGURE 8.3: Result of calculations for the total push back demand for all towing trucks combined on OPiuM simulation using regular OPiuM-curves.

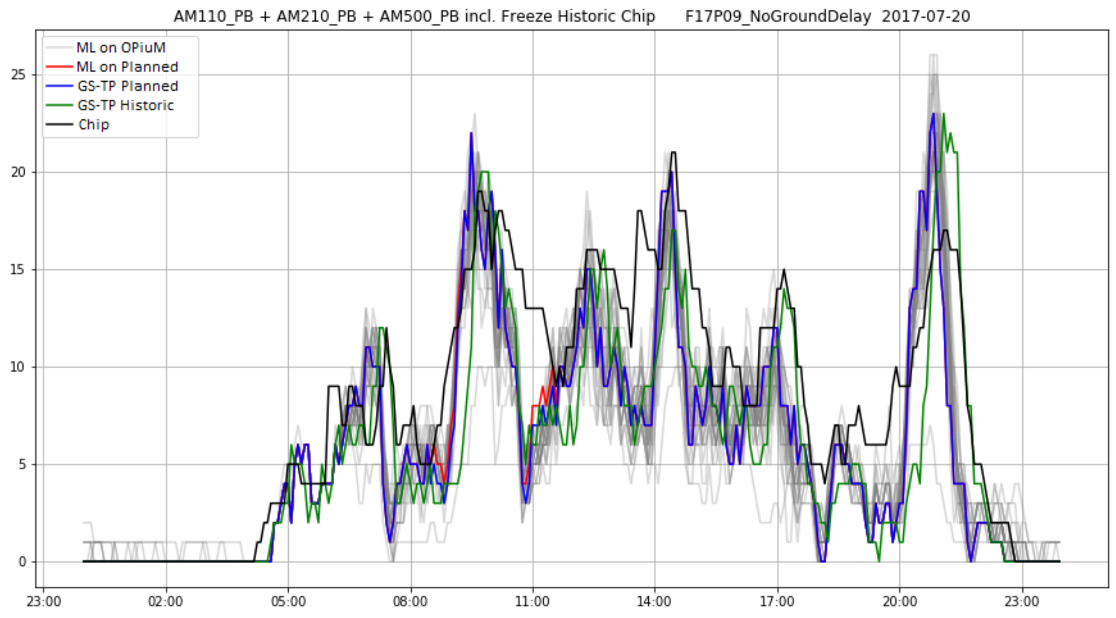


FIGURE 8.4: Result of calculations for the total push back demand for all towing trucks combined on OPiuM simulation using OPiuM-curves without simulated ground delay.

For the total demand for the AM500 towing truck, the results are shown in figures 8.5 and 8.6. Again, CHIP data could not be included since CHIP data does not allow for filtering on this equipment type.

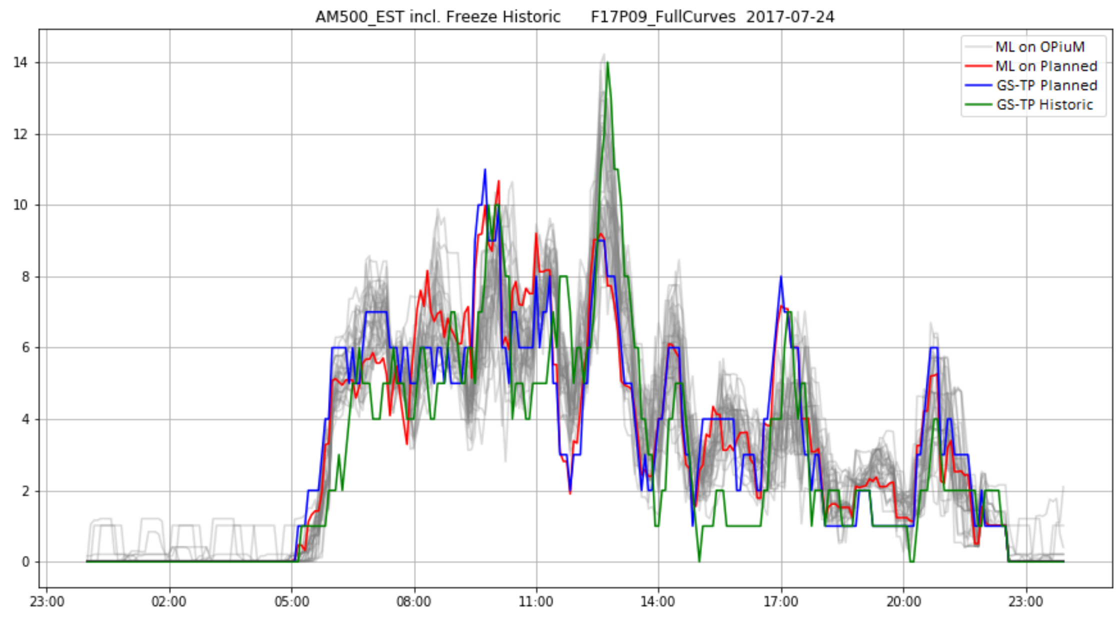


Figure 8.5: Result of ML model for the total demand for AM500 towing trucks on OPiuM simulation using regular OPiuM-curves.

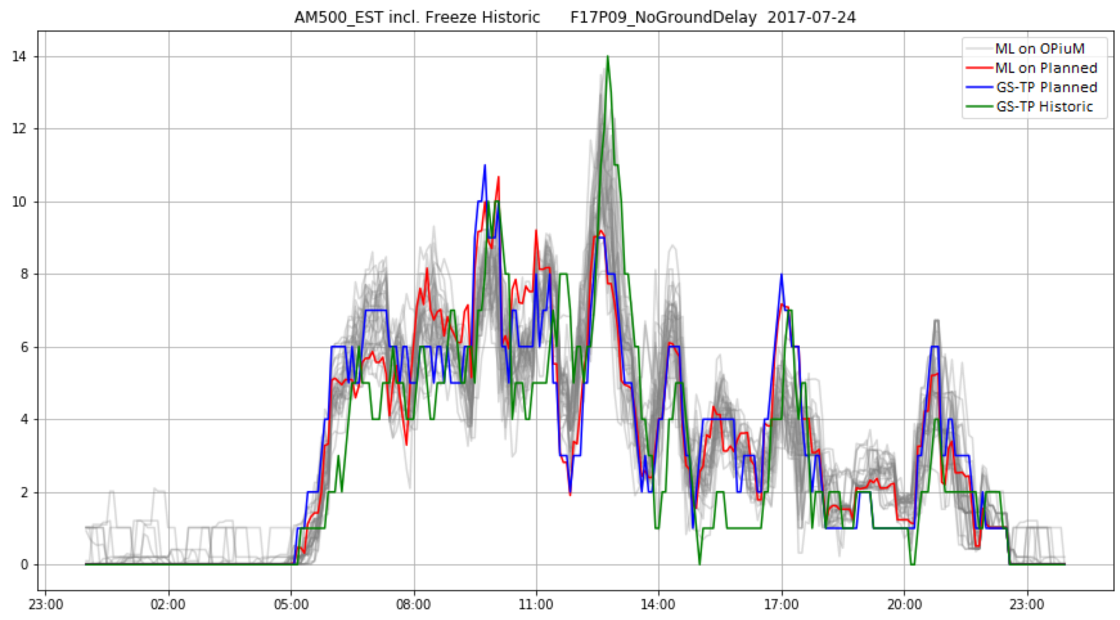


Figure 8.6: Result of ML model for the total demand for AM500 towing trucks on OPiuM simulation using OPiuM-curves without simulated ground delay.

Figure 8.7 shows the result of the simulated approach to estimate the demand for towing tasks for the AM210 truck on the day of operation.

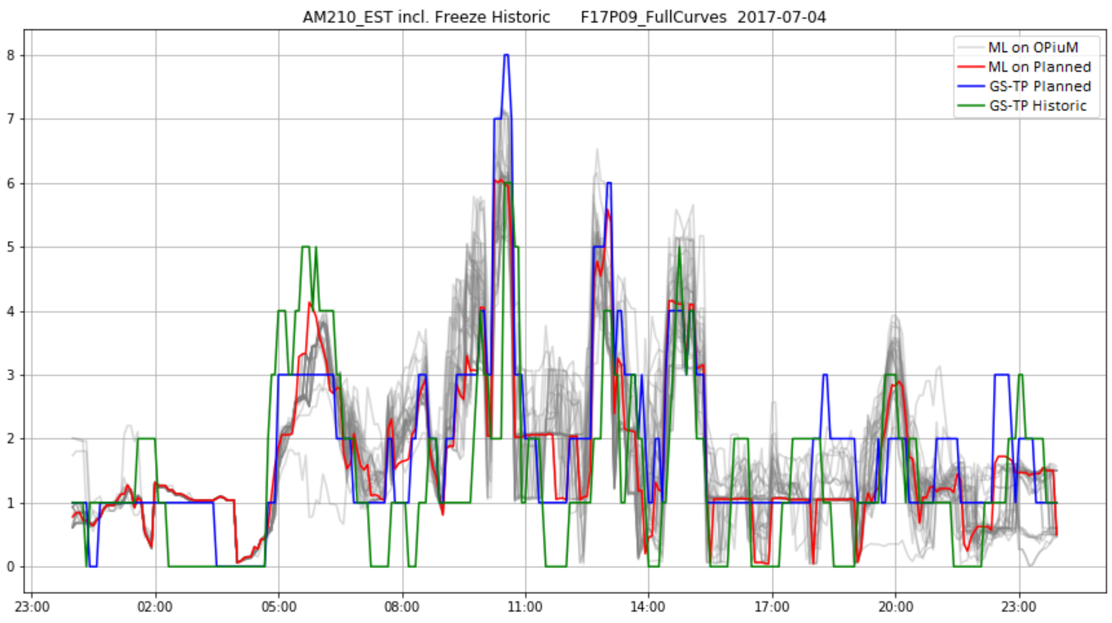


Figure 8.7: Result of ML model for demand for AM210 tow tasks on OPiuM simulation using regular OPiuM-curves.

Fuel Equipment

Figure 8.8 shows the result of the simulated approach for KTWs. Though this resource is only used to fuel aircraft that cannot be simulated by OPiuM, it might be still of interest to analyze the difference between GS-TP planning, retrospective planning and CHIP.

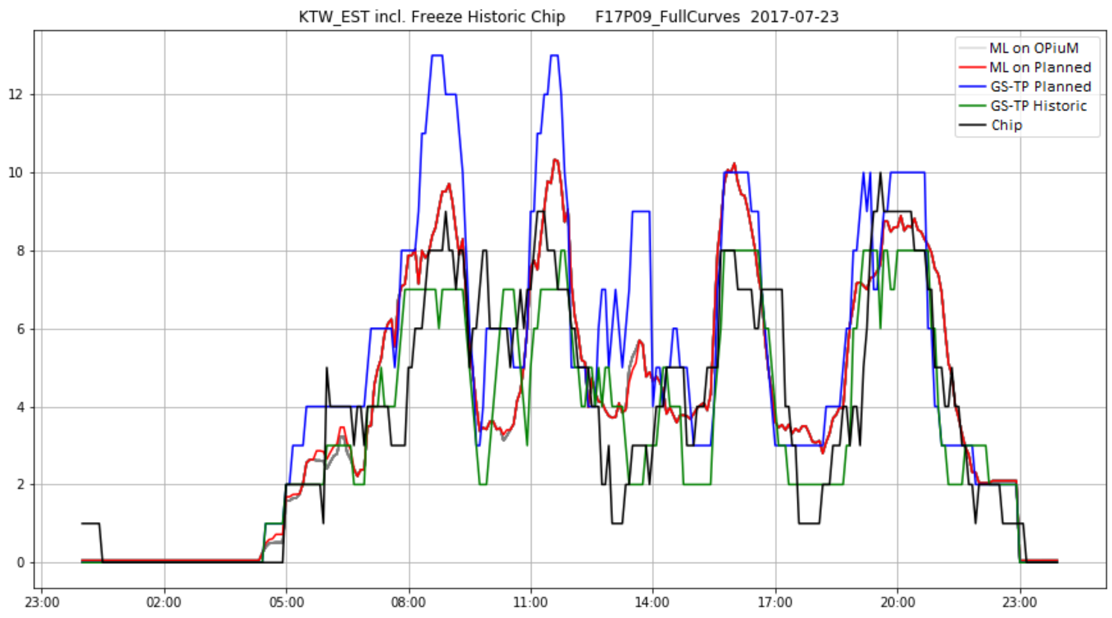


Figure 8.8: Result of ML model for demand for KTW fuel trucks on OPiuM simulation using regular OPiuM-curves.

The simulated demand for dispensers is shown in figures 8.9 and 8.10. These are interesting figures since dispensers are the only resources of which the flights can be simulated by OPiuM, have a retrospective planning and are available in CHIP on the right level of detail.

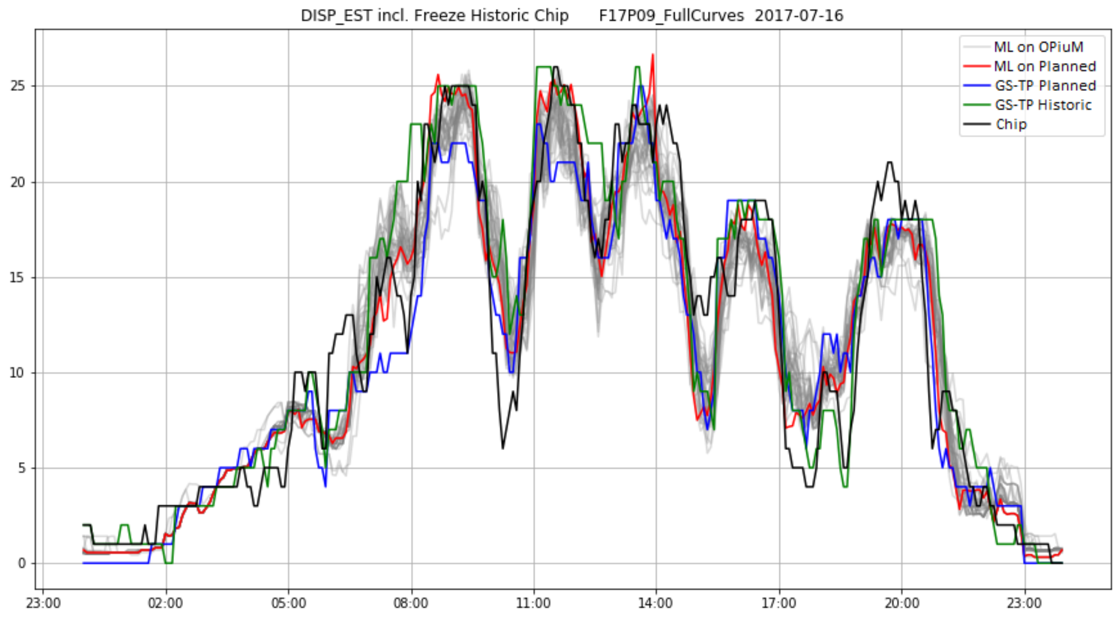


FIGURE 8.9: Result of ML model for demand for dispenser on OPiuM simulation using regular OPiuM-curves.

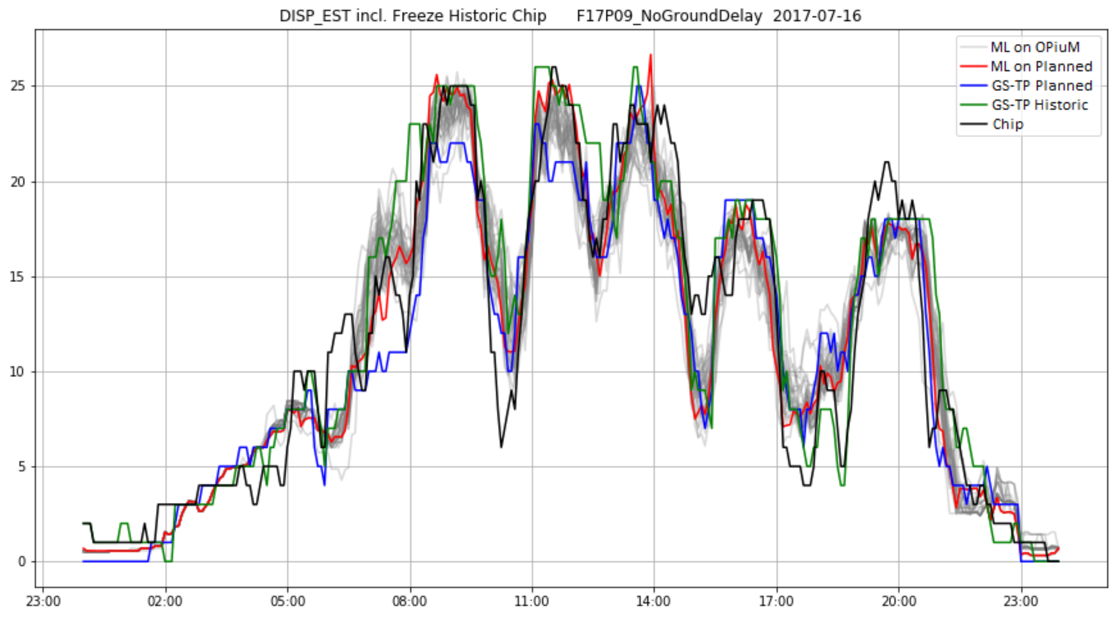


FIGURE 8.10: Result of ML model for demand for dispenser on OPiuM simulation using OPiuM-curves without simulated ground delay.

Baggage Loaders

The result of the calculations for the demand for rampsnakes are shown in figures 8.11 and 8.12. It is not interesting to show the result for powerstows since these resources are used to (un)load aircraft that cannot be simulated by OPiuM and the usage of these resources is not included in CHIP.

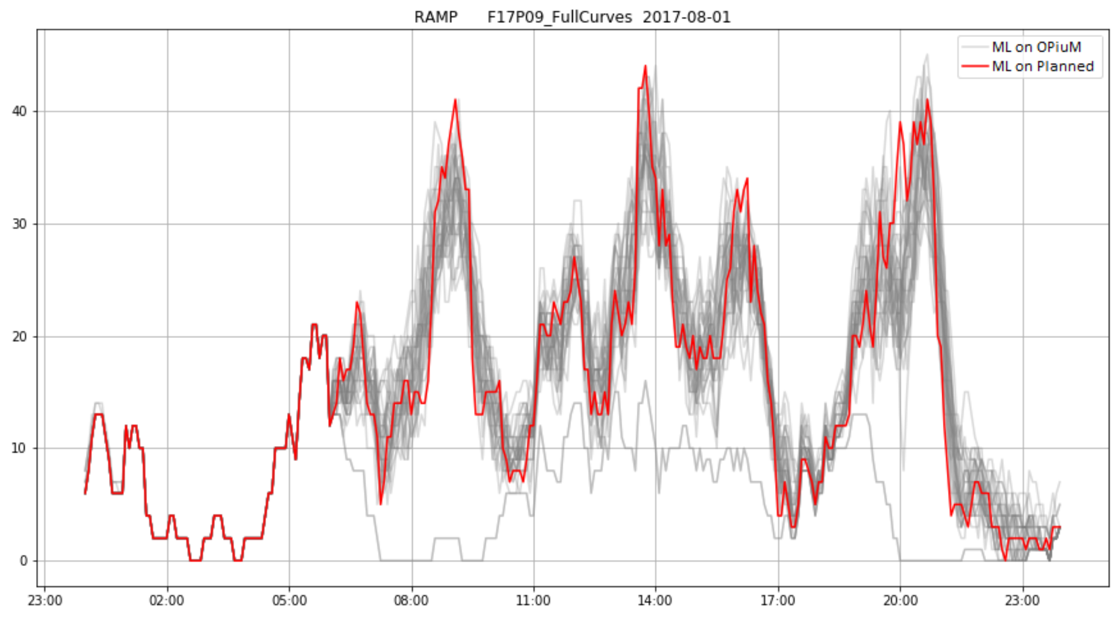


FIGURE 8.11: Result of calculation of the demand for rampsnakes on OPiuM simulation using regular OPiuM-curves.

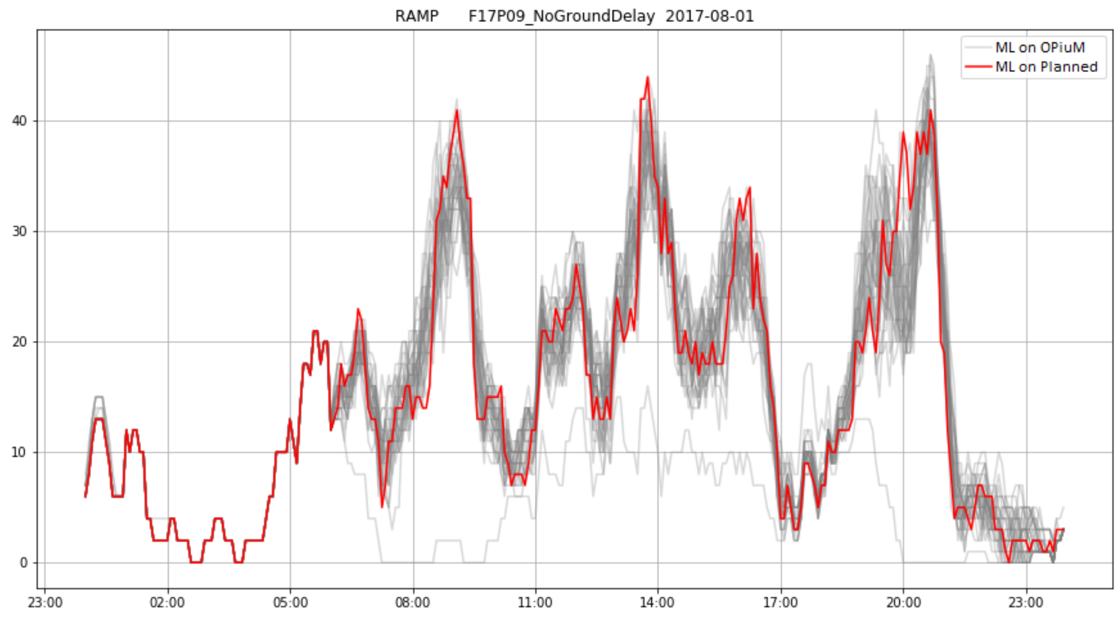


FIGURE 8.12: Result of calculation of the demand for rampsnakes on OPiuM simulation using OPiuM-curves without simulated ground delay.

Tabular Presentation of Performance Metrics

The following tables contain performance metrics as discussed in section 7.4.4. Just like the visual representation, these tables display the results of 30 OPiuM runs on the 9th Base Rolling Planning of 2017 (`F17P09`). The measures are an average of weeks 27 to 34 (3 July to 27 August). The average over all time points is given, as well as for selected interesting time intervals where the demand for this resource peaks.

For each time interval and for each resource the Rooted Mean Squared Error (RMSE) for a planned demand against an actual demand is given. Here we distinguish the following planned demand:

1.	*GS-TP Planned*	Resource demand according to GS-TP regular plannings process.
2.	*ML on Planned*	Resource demand according to Machine Learning models that try to estimate the regular GS-TP planning process.
3.	*Mean of ML on OPiuM*	Mean of the resource demand according to Machine Learning models on 30 OPiuM outputs.

Depending on the available data, the following actual demand can be used:

1.	*Historic*	Demand according to GS-TP retrospective resource planning.
2.	*CHIP*	Demand according to CHIP registration.

Furthermore, the percentage of resources that lies in the statistical confidence interval (between *p5* and *p95*) and the empirical interval (between the 2nd lowest en 2nd highest OPiuM observation) are presented.

	AM110		
	Full Day	**Bank2 (09:00- 10:00)**	**Bank6 (20:00 - 22:00)**
RMSE	*Historic*	*Historic*	*Historic*
GS-TP Planned	2.510	4.554	5.261
ML on Planned	2.510	4.554	5.261
Mean of ML on OPiuM	1.913	3.075	3.431
% Above P95	32.8	35.8	44.7
% In Between	38.8	20.3	20.9
% Below P5	28.6	43.9	34.3
% Above 2nd Highest	12.7	13.1	18.3
% In Between	77.9	74.7	69.7
% Below 2nd Lowest	9.4	12.2	12.0

Table 8.3: Results of comparison of planning, simulation and actuals for the total demand for AM110.

	AM500		
	Full Day	**Bank2 (09:20- 10:40)**	**Bank3-4 (12:00 - 13:30)**
RMSE	*Historic*	*Historic*	*Historic*
GS-TP Planned	1.770	3.412	2.301
ML on Planned	1.803	3.130	2.209
Mean of ML on OPiuM	1.656	2.573	1.928
% Above P95	24.4	37.1	35.6
% In Between	17.8	15.5	20.4
% Below P5	57.8	47.4	44.0
% Above 2nd Highest	11.5	17.0	14.4
% In Between	33.6	52.1	65.1
% Below 2nd Lowest	54.9	30.9	20.6

Table 8.4: Results of comparison of planning, simulation and actuals for the total demand for AM500.

	AM210				
	Full Day	**Bank1 (04:50 - 07:20)**	**Bank2-3 (10:00 - 11:20)**	**Bank3-4 (12:30 - 13:30)**	**Bank4-5 (14:30 - 15:30)**
RMSE	*Historic*	*Historic*	*Historic*	*Historic*	*Historic*
GS-TP Planned	1.492	2.034	2.351	2.084	1.684
ML on Planned	1.342	2.005	2.022	1.728	1.525
Mean of ML on OPiuM	1.356	2.161	1.754	1.715	1.458
% Above P95	29.9	68.7	33.9	20.4	35.0
% In Between	9.4	5.5	12.2	11.3	19.7
% Below P5	60.7	25.8	53.9	68.3	45.3
% Above 2nd Highest	21.5	61.3	18.3	9.5	19.3
% In Between	26.1	18.9	36.3	37.7	46.5
% Below 2nd Lowest	52.4	19.8	45.5	52.7	34.1

Table 8.5: Results of comparison of planning, simulation and actuals for the total demand for AM210.

	KTW							
	Full Day		**Bank1-2 (07:00- 09:30)**		**Bank3 (10:55 - 12:30)**		**Bank5 (15:15 - 17:00)**	
RMSE	*Historic*	*CHIP*	*Historic*	*CHIP*	*Historic*	*CHIP*	*Historic*	*CHIP*
GS-TP Planned	3.277	2.113	4.739	2.534	5.225	2.856	4.791	2.956
ML on Planned	2.727	1.666	3.868	1.843	4.244	1.974	4.298	2.427
Mean of ML on OPiuM	2.771	1.675	3.998	1.870	4.324	1.982	4.342	2.445

Table 8.6: Results of comparison of planning, simulation and actuals for the total demand for KTW.

	Dispenser					
	Full Day		**Bank1.5-2 (08:00- 10:00)**		**Bank3-4 (11:00 - 14:15)**	
RMSE	*Historic*	*CHIP*	*Historic*	*CHIP*	*Historic*	*CHIP*
GS-TP Planned	6.845	3.256	11.240	3.660	10.522	3.678
ML on Planned	6.842	3.004	11.138	3.091	10.766	3.541
Mean of ML on OPiuM	6.735	2.965	10.608	3.475	10.406	3.098
% Above P95	43.9	47.4	60.7	57.2	49.7	44.6
% In Between	7.6	8.3	5.7	10.0	8.9	11.9
% Below P5	48.5	44.3	33.6	32.8	41.5	43.5
% Above 2nd Highest	30.8	33.0	48.0	37.5	14.4	27.5
% In Between	27.8	32.4	23.6	42.7	65.1	45.5
% Below 2nd Lowest	41.4	34.7	28.4	19.9	20.6	26.9

Table 8.7: Results of comparison of planning, simulation and actuals for the total demand for Dispenser.

Chapter 9

Discussion and Conclusions

This chapter contains the discussion (section 9.1) and the conclusions (section 9.2) of the results of *Part II* as presented in chapter 8. Furthermore, recommendations for the usage of these results for KLM and future studies are given in section 9.3.

9.1 Discussion

The results as described in chapter 8 are hard to analyze for a couple of reasons. First, the reliability of the CHIP data does not allow for good validation. Furthermore, the stochasticity of operational events and the correlation between delays of aircraft and resources - something that is not included in OPiuM simulation - might cause outliers that will never be observed after simulation. Another problem is that the demand for resources according to CHIP can never be higher than the number of active resources at that moment. This does not allow the analysis of what would have happened if the number of active resources would have been higher. Despite the flaws of validation, it it still possible to find some patterns and draw some conclusions.

AM110 Push Backs

As shown in figures 8.1 and 8.2, the simulated demand for push backs with an AM110 does not differ that much from its planned demand. Though the RMSE as presented in table 8.3 might appear to be high, this is mostly due to a shift of the peak. In other words: the vertical error in the graphs - the error that is measured by this RMSE - might be high, it can be seen that the horizontal error is not that big. The demand on the highest peaks around 09:30 and 21:00 seems to have a high variance on the day of operation according to the simulations. In general, the retrospective planning of GS-TP lies in the grey area of simulated demand (80% in the empirical interval). This indicates that the combination of OPiuM and ML models results in a realistic resource demand on the day of operation.
The departure delays are visible in the downward slopes after demand peaks. Here the simulated demand is almost always higher than the planned demand. This makes sense since flights often delay but almost never depart much earlier. Of course this effect is larger for the simulations that

allow for ground process delays (figure 8.1) than for simulations where ground process delays cannot occur (figure 8.2).
The demand in the latest bank (21:00) shows signs of delayed arrivals: most simulated demand (grey lines) peaks at a later moment than the planned demand (blue line). This is the effect of the delay that has been built up during the day and that is visible on the last bank since a lot of aircraft that require a push back from the AM110 are on Schiphol during that bank.
Unfortunately, all that can be discussed is the performance measured using GS-TP's historical planning. There is no reliable operational data available on this level of detail to support the arguments in the discussion of these results. Nevertheless, all effects as discussed above make sense according to KLM's strategy and operational performance.

Total Demand for Push Backs

For the total demand for push backs, the same conclusions can be drawn as for the simulated demand for push backs of AM110, though on this aggregation level CHIP data can be used for comparison.
In general the CHIP data seems to follow the patterns of both resource planning, retrospective resource planning and simulated resource demand. Only in the last peak (around 21:00), the demand according to CHIP is much lower than the planned and simulated demands. Here it might be the case that flights are canceled and therefore do not require a push back anymore. It can also be the case that these trucks have been delayed or that they were occupied for towing tasks and that as a consequence a lot of flights have been delayed. The delay is observed by the difference of the green/black and blue lines at the downward slope of the last peak. This delay is more than the simulated delays as shown by the grey lines. These indicates that there were not enough resources available in order to operate all push backs.

Total Demand for AM500

The total simulated demand is shown in figures 8.5 and 8.6. Here it can be seen that the simulated demand does not differ that much from the retrospective GS-TP planning. This observations is substantiated by the performance metrics in table 8.4. One peak has even been correctly pointed out as a possibility by the simulation runs (around 12:30). On this peak, 65.1% of all historic planned demand lies in the empirical confidence interval. Of course this does not mean that this demand has been observed on the day of operation, but it indicates that the simulation runs are capable of showing similarities to the retrospective planning.

Total Demand for AM210

The total demand for AM210 towing trucks (figure 8.7) shows that planned, retrospectively planned and simulated demand might differ a lot. This is a direct consequence of the small number of available AM210 resources and its correlation with gate occupation. Table 8.5 shows that for Bank2-3 and Bank3-4 the mean of ML models on OPiuM simulations is a better predictor for historic demand that the initial GS-TP planning.
A recurring observation is that the demand as initially planed by GS-TP for the the peak at 10:30 is almost always higher than the ML estimation, the retrospective planning and the simulated

demand. This error can also be observed in table 8.5. This might be an indication that the demand on this peak is planned too high by GS-TP.

Total Demand for KTWs

For KTWs a recurring pattern is observed that both the retrospective demand and the actual demand according to CHIP are lower than the demand as planned by GS-TP. It is remarkable that the ML estimation (red line) finds it really hard to estimate the planned demand by GS-TP (blue line) but it is pretty close to the retrospective planning (green line) and actuals (black line). Table 8.6 shows that both the *ML on Planned* and the *Mean of ML on OPiuM* are better predictors for the actual demand on the day of operation than the *GS-TP on Planned.* This might indicate a flaw in the resource planning by GS-TP. The ML models probably could not predict the demand as calculated by GS-TP because gate planning is an important feature for the demand for KTWs. From comparison with actual data it seems like the gate planning of a planned schedule results in an unrealistic high demand for KTWs.

Total Demand for Dispensers

For the total demand for dispensers (figures 8.9 and 8.10) a same pattern is observable as for the total demand for KTWs as described in the previous subsection. Though this pattern is less clear in table 8.7. On the peaks, the GS-TP planning (blue line) shows a lower resource demand than the retrospective planning (green), actual CHIP data (black), the ML estimation (red) and the simulated delays (grey). The ML model tends to overestimate the planned demand for dispensers but it turns out that this result is a bit closer to retrospective planning and actual CHIP data. This might be an indication that operational delay is underestimated by the GS-TP planning process for dispensers or that the gate allocation of a planned schedule is too restrictive.

Total Demand for Rampsnakes

Figures 8.11 and 8.12 include the demand for rampsnakes and show that most of the demand peaks are lower and wider when they are simulated. This is caused by the fact that aircraft that arrive earlier than scheduled, can already be unloaded by rampsnakes. Also delayed aircraft might require that a rampsnake remains available depending of the cause of the delay. Only the peak on the last bank shows that the peak of the simulated demand is higher than originally planned. This may be caused by flight delay that builds up over the day. In this last bank a lot of flights arrive delayed, more rampsnakes are demanded in order to handle these aircraft. Unfortunately, there is no actual data to support these observations.

9.2 Conclusions

Part II of this thesis focused on using a simulated approach to answer the resource question

3. Given a flight schedule, what can be said about the demand for ground resources on the day of operation?

It has been shown that in some cases OPiuM simulation output combined with machine learning models from *Part I* might be used to give an indication of the demand for ground resources on the day of operation. The results show that most of the times this combination shows to be a better estimation of operational demand according to CHIP or GS-TP's historical planning. However, because of the incompleteness and questionable quality of the actual data, it is hard to prove this statement.
Here the machine learning models show their value as a shortcut for the extensive capacity requirement planning of GS-TP. Without this shortcut, all OPiuM simulation output have to be evaluated with GS-TP planning.
The discussion from section 9.1 can give GS-TP a better insight in the operational fluctuation of the resource demand. For example, the demand for resources that are required to handle medium body aircraft, like AM110 and rampsnakes, is often higher in the last bank because of delay that has been build up during the day.
For some resource the machine learning models failed to show good performance of estimating the GS-TP initial resource planning process for a planned schedule. However, it has been shown that these models were closer to the actual demand on the day of operation. This can indicate that GS-TP underestimates the influences of operational delays on the gate planning and thus on the resource demand.

9.3 Recommendations

It is shown that the result of flight delay simulation (OPiuM), in combination with machine learning models to estimate the resource planning process of GS-TP (*Part I* of this report), shows patterns of resource demand that are comparable with the available actual data (CHIP) and retrospective planning. However, there is not enough data of sufficient quality available in order to substantiate this statement. Therefore it is recommended to improve the quality of the data and to make it possible to filter on all resource types and tasks in order to make such an analysis possible.
Apart from the simulation results, the comparison with actual data and retrospective planning showed that the initial resource planning of GS-TP at some points consistently differs from what is observed in operations. This might be caused by a gate planning on a planned schedule that is too conservative compared to the gate occupation on the day of operation. It is recommended to study these differences in greater detail in order to achieve a more realistic planned resource demand.

Future Studies

As indicated above, it is interesting to study the cases where the GS-TP planning consistently differed from the observed resource demand. Is this indeed caused by the gate allocation of a planned schedule? It is possible that GS-TP planning shows a worst-case scenario that occurs when the flight schedule is carried out as it is scheduled. In that case the current planning would suffice but the high planned demand for such resources should not be an important point of discussion since this worst-case scenario will almost never be observed.

Furthermore, the construction OPiuM and machine learning models does not allow for an analysis of the cause of delays. Do flights delay because the flight time was too long, because there were not enough resources available or because the processes of these resources were delayed? A more detailed simulation can answer these questions, show the robustness of a schedule and can clarify the source of delays.

Part III
Aircraft Movements on Schiphol as a CTMC

Given a flight schedule, what can be said about the demand for ground resources on the day of operation?

A Time-Inhomogeneous Continuous Time Markov Chain Approach

Chapter 10

Introduction Part III

In *Part III* of this thesis it is aimed to build a continuous-time Markov chain (CTMC) model that can give an indication of the demand for towing equipment on the day of operation and the robustness of a flight schedule. In contrast to the simulated approach, as described in *Part II*, this model will be more conceptual and independent of GS-TP's detailed plannings process. The goal is not to give a detailed overview of possible tasks and their delays but to analyze the general effect of arrival and departure peaks on the resource demand.
This introductory chapter is included to separate *Part III* from the rest of this thesis because of its different approach. The motivation for this approach is given in section 10.1. Section 10.2 introduces the methods that are used and section 10.3 sketches the outline of *Part III*.

10.1 Motivation

In *Part II* it has been shown that a simulated approach can be used to give an indication of the resource demand on the day of operation. However, this approach requires multiple simulation runs in order to provide insight in the distribution of the number of necessary resources and it is hard to verify that ML models, that are trained on planned flight schedules where no uncertainty is included, can be applied on disrupted schedules. Though this simulated approach has shown to give some satisfying results (chapter 8) and provokes some discussions (chapter 9), a model that include dependencies between processes is desired.
OPiuM simulation does not include resource capacity restrictions and correlation between the delays of processes. If, for example, some arrival handling processes are delayed, the demand for towing equipment might increase because gates availability drops. OPiuM does not account for such dependencies while in a CTMC this can be included since the description of states may forbid some events to happen.

10.2 Methods

A continuous-time Markov chain will be described that models arrivals, departures and ground processes of aircraft on Schiphol. An important assumption that reduces complexity is that all aircraft are considered to be equal independent of the airline, aircraft type and origin or destination. Only *wide* body aircraft are considered (Category 6 - 9).
Schiphol's bank system (figure 2.2 in section 2.1) results in fluctuating arrival and departure rates. This time-homogeneity will be tackled by applying a time-discretization of the continuous process. Methods of iterative state expansion and dynamic state space truncation are introduced to evaluate the large time-inhomogeneous CTMC.

10.3 Outline

First, a background of continuous-time Markov chains is provided in chapter 11. This overview includes time-discretization, uniformization, a Markov reward approach, truncation and the terminology of transition classes. Then, a description of movements of *wide* body aircraft on and around Schiphol in the form of a CTMC is given in chapter 12. Finally, chapter 13 presents an algorithm to solve large state time-inhomogeneous CTMCs.

Chapter 11

Background on Continuous Time Markov Chains

This chapter describes the theoretical background of Continuous Time Markov Chains (CTMCs). This is necessary to understand the Schiphol Markov model that will be introduced in chapter 12 and the proposed method to solve time-inhomogeneous CTMCs with a large state space (Chapter 13).
First the basics and core definitions of CTMCs are given in section 11.1. Then the techniques of time-discretization and uniformization are discussed in sections 11.2 and 11.3 respectively. A Markov reward approach is described in section 11.4 followed by section 11.6 concerning state space truncation. Finally, transition classes are explained in section 11.7.

11.1 Basics and Definitions

A continuous-time Markov chain is a memoryless stochastic process with a discrete state space that is analyzed in continuous time.

Definition 11.1. Continuous-Time Markov Chain (CTMC). A stochastic process $\{X_t : t \in [0, \infty)\}$, with values in a countable state space S, is called a continuous-time Markov chain if for any $0 \leq t_1 < t_2 < \ldots < t_{n+1}$ and corresponding set $\{i_1, \ldots, i_{n-1} : i_k \in S, 1 \leq k \leq n-1\}$ and $i, j \in S$ such that $P(X_{t_n} = i_n, X_{t_{n-1}}, \ldots, X_1 = i_1) > 0$, we have

$$P(X_{t_{n+1}} = j | X_{t_n} = i_n, X_{t_{n-1}} = i_{n-1}, \ldots, X_1 = i_1) = P(X_{t_{n+1}} = j | X_{t_n} = i_n) \tag{11.1}$$

Where (11.1) can be referred to as the Markov property - or memoryless property - which states that the probability of a transition from a current state only depends on that state and is independent of its previous states [21].
A CTMC $\{X_t\}$ can therefore be described by its countable state space S and transition probabilities

$$\mathbf{P}_t(i,j) := P(X_t = j | X_0 = i) \qquad \forall i, j \in S \tag{11.2}$$

Definition 11.2. Time-Homogeneous CTMC. A Markov Chain is time-homogeneous if its transition probabilities are not dependent of time:

$$P(X_{t+h} = j|X_t = i) = P(X_{s+h} = j|X_s = i) \qquad \forall s, t \in [0, \infty),\ h > 0 \tag{11.3}$$

Instead of working with transition probabilities between states for some time length t, it is more convenient to use transition rates

$$q(i,j) = \lim_{h \to 0} \frac{\mathbf{P}_h(i,j)}{h} \qquad j \neq i \tag{11.4}$$

for all $i, j \in S$, where the total rate out of state i is defined by

$$q(i) = \lim_{h \to 0} \frac{1 - \mathbf{P}_h(i,i)}{h} = \sum_{j \neq i} q(i,j) \tag{11.5}$$

and for $i \in S$ we have that

$$q(i,i) = -q(i) \tag{11.6}$$

These rates form the infinitesimal generator matrix $\mathbf{Q}$. In the time-inhomogeneous case, $\mathbf{Q}_t$ is used to indicate the transition rates at time t. The elements of $\mathbf{Q}_t$ - $q_t(i,j)$ for $i, j \in S$ - are the time dependent variants of (11.4) and (11.5).
Continuous-time Markov models are widely studied and have multiple applications such as population models, telecommunications and queueing analysis [22–25]. The limit behaviour of these models is often analyzed and a typical point of interest is the steady state probability distribution of its states.

Definition 11.3. Steady State Distribution. The steady state distribution π of a CTMC is the probability distribution vector that solves the global balance equations.

$$\begin{aligned} 0 &= \pi \mathbf{Q} \\ \sum_{i \in S} \pi(i) &= 1 \end{aligned} \tag{11.7}$$

It has been shown that under certain circumstances this steady state distribution can be found in a simple and elegant way by, for example, a simple function described by a model's transition rates [26] or a product form of the steady state distributions of its state dimensions [27].
If a model does not meet the requirements for such elegant solutions, steady state analysis can be done by methods like time-discretization or uniformization. For these techniques it is assumed that the transition are uniformizable [3]

Definition 11.4. Uniformizable The transition rates of a Markov process are said to be uniformizable if they can be uniformly bounded by some finite constant $B < \infty$

$$q(i) = \sum_{j \neq i} q(i,j) \leq B \qquad \forall i \in S \tag{11.8}$$

11.2 Time-Discretization

A uniformizable CTMC, i.e. a Markov chain for which its transition rates satisfy (11.8) for some B, can be analyzed using time-discretization. The idea of this approach is to partition the continuous time interval with horizon T into small steps of size $1/B$ such that one-step state transitions between these discrete time points can be evaluated by a single one-step transition matrix. This allows for analysis of state probabilities for the total of T/B time points. Here B can be taken arbitrarily small as long as it satisfies (11.8). From now on the smallest value of B for which (11.8) still holds is used and therefore

$$B = \min_{i \in S} q(i) = \min_{i \in S} \sum_{j \neq i} q(i,j) \tag{11.9}$$

Note that for this value of B, the expected duration between two transitions is equal to $1/B$. Therefore, evaluating a CTMC at discrete time point with step $1/B$ seems like a good way to approximate the process in continuous time.

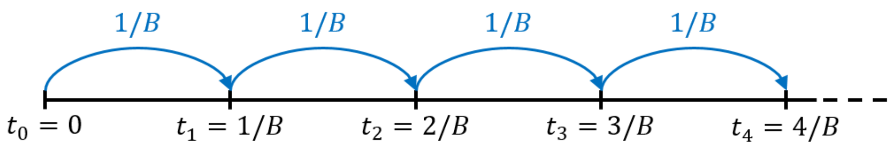


Figure 11.1: Time discretization with discretization step size B.

Equations 11.4 and 11.5 state that the continuous transition rates are approached when the discretization step goes to 0. Therefore, a smaller step size will result in a process that is a better approximation of the original process. Now take the step size for which it is expected that one transition happens $1/B$, and divide this step in β more small steps. This results in step size

$$0 \leq h = \frac{1}{\beta B} \leq \frac{1}{B} \tag{11.10}$$

Here, $\beta > 1$ can be seen as a discretization factor that is used to decrease the minimum step size B. Increasing β will result in a smaller discrete step size h and will therefore be a better approximation of continuous time Markov process with the trade-off that more steps have to be analyzed. Because it is expected that an event occurs every $1/B$ time units, there is a relative high probability that no event occurs for a step of size h.

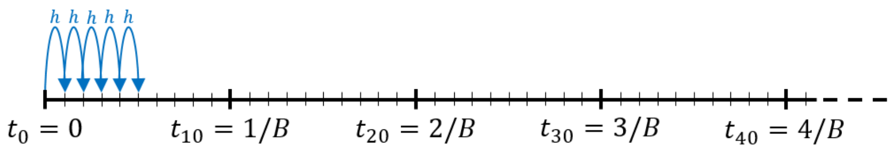


Figure 11.2: Time discretization with discretization step size h where $\beta = 10$ is used.

The CTMC with infinitesimal generator matrix $\mathbf{Q}$, i.e. the matrix that contains the transitions rates as in (11.4) and (11.5), can then be approached for small time steps by its time-discrete approximation with one step transition matrix

$$\mathbf{P}_d = \mathbf{I} + h\mathbf{Q} \tag{11.11}$$

where it can happen that after one step of time length h, the state of the system does not change. The state probability vector $\pi_{c,t}$ of the original continuous process at time t can now be iteratively approximated by its discrete variant using

$$\begin{aligned} \pi_{d,0} &= \pi_0 \\ \pi_{d,t+h} &= \pi_{d,t}\mathbf{P}_d \end{aligned} \tag{11.12}$$

This approach can be shown to provide error bounds of order $O(h)$.
As $h \to 0$ this time-discrete process approaches the original CTMC. A lower value of h gives a better approximation but requires more computational time. An important step in order to solve the Schiphol CTMC instance is that this approach can be easily extended for time-inhomogeneous CTMCs by simply replacing the one step transition matrix (11.11) by a time dependent version

$$\mathbf{M}_t = \mathbf{I} + h\mathbf{Q}_t \tag{11.13}$$

Here $\mathbf{Q}_t$ denotes the transition rate matrix of transition rates at time t. $\mathbf{M}_t$ is then the one step transion matrix for step size h according to the transition rates at time t.
If $\mathbf{M}_t$ changes between two steps of the iterating process (11.12), i.e. for some n, $t_n = nh$ and $t_{n+1} = nh + h$ we have that $\mathbf{M}_{t_n} \neq \mathbf{M}_{t_{n+1}}$, it can be decided to use either one of the two or to construct some sort of average transition rate. If $\mathbf{M}_t$ is not changing continuously it might be possible to choose h such that transition changes only occur at time points $t_0 = 0, t_1 = h, t_2 = 2h, \dots$ at which the model is evaluated.

11.3 Standard Uniformization

Uniformization was first introduced by Jensen [28] and has been successfully implemented for performance analysis of time-inhomogeneous CTMCs, e.g. see [4]. It can be seen as a randomized discretization and is well suited to be applied to time-inhomogeneous systems [29]. Following [3] we have that, using uniformization rate B from (11.8), the one step probability matrix $\mathbf{P}$ can be defined by

$$\mathbf{P} = \mathbf{I} + \frac{1}{B}\mathbf{Q} \tag{11.14}$$

with elements

$$\mathbf{P}(i,j) = \begin{cases} q(i,j)/B & j \neq i \\ 1 - \sum_{l \neq i} q(i,l)/B & j = i \end{cases} \tag{11.15}$$

Let π_c be the steady state of the CTMC with infinitesimal generator matrix $\mathbf{Q}$ and π_d the steady state of the discrete time Markov chain (DTMC) with transition matrix $\mathbf{P}$, derived as in (11.15). It can be proven that these steady state vectors coincide [3] and thus

$$\pi_c(i) = \pi_d(i), \qquad i \in S \tag{11.16}$$

Furthermore, the transition matrix of the CTMC $\mathbf{P}_t(i,j) = P(X_t = j|X_0 = i)$ and one step transition matrix of the DTMC $\mathbf{P}$ are related by

$$\mathbf{P}_t(i,j) = \sum_{k=0}^{\infty} \frac{(tB)^k}{k!} e^{-tB} \mathbf{P}^k(i,j), \qquad \forall i,j \in S, \quad t > 0 \tag{11.17}$$

where $\mathbf{P}^k$ is the k-th matrix power of $\mathbf{P}$. Note that for $k = 0, 1$ this process is similar to time discretization (section 11.2). This equation can be interpreted as follows. The probability of k transitions in time interval t follows a Poisson distribution with parameter tB, i.e. $P(K = k) = \frac{(tB)^k}{k!} e^{-tB}$. In the case of k transitions in time interval t, the probability of going from state i to j follows from the k-th matrix power of one-step transition matrix $\mathbf{P}$. It can therefore be seen as time discretization with exponentially distributed discretization step (figure 11.3).

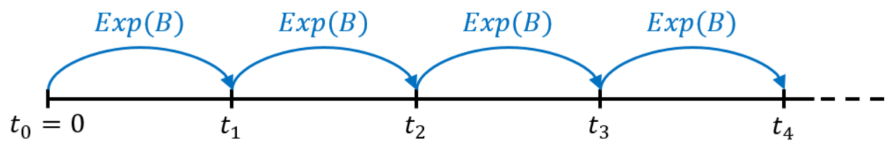


FIGURE 11.3: Time uniformization with randomized discretization step size that follows an exponential distribution with parameter B.

For numerical evaluation, the sum of (11.17) is often truncated by value K such that

$$1 - \sum_{k=0}^{K} \frac{(tB)^k}{k!} e^{-tB} < \xi \tag{11.18}$$

In words, if the probability of having more than K steps in time interval t is lower than some truncation threshold ξ, these steps are not included in an approximation of (11.17).
For time-inhomogeneous uniformization one has to consider the uniformized time dependent variant of one step transition matrix (11.15) by

$$\mathbf{M}_t(i,j) = \begin{cases} q_t(i,j)/B & j \neq i \\ 1 - \sum_{l \neq i} q_t(i,l)/B & j = i \end{cases} \tag{11.19}$$

with

$$\begin{aligned} q_t(i,j) &= \lim_{h \to 0} \tfrac{P(X_{t+h}=j|X_t=i)}{h}, & j \neq i, \quad & t \geq 0 \\ q_t(i) &= \textstyle\sum_{j \neq i} q_t(i,j) \leq B, & \forall i \in S, \quad & t \leq T \end{aligned} \tag{11.20}$$

Again, following [3] the probability to go from state i in time s to state j in time t with $i, j \in S$ and $s \leq t \leq Z$ has proven to be

$$\mathbf{P}_{s,t}(i,j) = \sum_{k=0}^{\infty} \frac{(t-s)^k B^k}{k!} e^{(t-s)B} \underset{\{t_1 \leq t_2 \leq \ldots \leq t_k\}}{\int_s^t \int_s^t \ldots \int_s^t} \mathbf{M}_{t_1} \mathbf{M}_{t_2} \ldots \mathbf{M}_{t_k}(i,j) d\bar{\mathbf{H}}(t_1, t_2, \ldots t_k) \tag{11.21}$$

where $d\bar{\mathbf{H}}(t_1, t_2, \ldots t_k)$ is the density of a k-dimensional uniform distribution of $t_1 \leq t_2 \leq \ldots \leq t_k$ at $[s,t] \times [s,t] \times \ldots \times [s,t] \subset \mathbb{R}^k$.
Since (11.21) is numerical impractical it can be an approximated using either

1. A discrete approximation of the integrals.
2. Monte Carlo sampling of time points $t_1, \ldots t_k$.
3. Piecewise evaluation of n time-independent intervals $(t_1, t_2), (t_2, t_3), \ldots, (t_{n-1}, t_n)$ for which the one step transition probabilities (11.19) do not change.

This approach, using uniformization, will not be used in this thesis to achieve any results. The reason why this section is included is to provide the reader a more complete overview of solution approaches. Instead, time-discretization will be used that is described in section 11.2. This is a more direct approach and computationally less expensive though it will be sufficient in order to evaluate the Schiphol CTMC instance.

11.4 Markov Reward Approach

A common method to keep track of the performance of a CTMC is to define so called reward vectors [4]. A reward vector r describes the reward for all states in the analysis. It assigns a reward rate of $r(i)$ to state i. In other words, the dot product $\langle r, \pi_t \rangle$ of the reward vector and the state probability vector at time t results in the expected reward at time t.

$$E[reward]_t = \langle r, \pi_t \rangle \tag{11.22}$$

The expected total reward over time interval $[0, T]$ for a time-homogeneous CTMC with initial state i at $t = 0$ can be described by

$$\int_0^T \sum_{j \in S} \mathbf{P}_s(i,j) r(j) ds \tag{11.23}$$

Where $\mathbf{P}_s(i,j)$ is the probability of being in state j at time s while being in state i at time 0 as defined by (11.2).

11.5 Time-Inhomogeneous Markov Reward Approach

For time-inhomogeneous CTMCs, one has to use the variant of the probability matrix that accounts for rate changes

$$\int_0^T \sum_{j \in S} \mathbf{P}_{0,s}(i,j) r(j) ds \tag{11.24}$$

Where $\mathbf{P}_{0,s}$ is defined as in (11.21). This probability can be approached by uniformization methods or by discretization in the form of

$$\sum_{t=0}^{T/h} h \left\langle r, \pi_{d,t} \right\rangle \tag{11.25}$$

Here $\pi_{d,t}$ follows from iterative process (11.12).
When there is an interest in a reward for a process that shows big differences between peak

and off-peak behaviour, it is more interesting to analyze how this reward progresses over time using (11.22). This progress can be approximation using discrete time state distribution $\pi_{d,t}$ that follows from discretization.

11.6 State Space Truncation

Markov models with a large or infinite state space can be numerically hard to analyze. A particular field that has to deal with such problems is the field of Markov Population Models (MPM) [25]. A method to approximate such models is to truncate the state space by simply removing some states from the analysis [4]. What is left is a truncated state space $\bar{S} \subset S$ with transition rates

$$\bar{q}(i,j) = \begin{cases} 0, & i = \zeta \text{ or } j \notin \bar{S} \\ q(i,j), & j \in \bar{S} \\ \sum_{k \notin \bar{S}} q(i,k), & j = \zeta \end{cases} \tag{11.26}$$

for all $i \in \bar{S}$, where $\zeta \in \bar{S}$ might be some absorbing state representing the loss of information of truncation $\bar{S}$.
With this static form of truncation, states are eliminated from the analysis in advance. This elimination causes a bias in both the state probability distribution vector and the reward that follows from this distribution compared to the non-truncated variant. It has been shown that this bias can be bounded analytically [4].

11.7 Transition Classes

Transition classes can be used to represent transitions of CTMCs. This is typically done for models with large or infinite state space such as MPMs [25]. Transition classes are an elegant way to represent transitions from one state to another since each transition class corresponds with an event that alters the current state of the system.

Definition 11.5. Transition Class. A transition class η of a CTMC with state space S is a triple (G, ν, α) with

- Guard $G \subset S$, the set of states on which the transition applies.
- Change vector $\nu \in \mathbb{Z}^{|S|}$, the effect of the event on the current state.
- Transition rate function $\alpha(n,t)$ that might depend on state $n \in G$ and time t.

In other words, transition class η describes that all states $n \in G$ will move to state $n + \nu \in S$ with rate $\alpha(n,t)$.

11.8 Algorithmic Description

Algorithm 2 and algorithm 3 in appendix D describe time-homogeneous and time-inhomogeneous discretization respectively. Both algorithms take a CTMC with state space S and transition class description η as an input. Algorithm 1 is used to generate a transition matrix from the given CTMC. The time-inhomogeneous variant uses algorithm 4 to determine when it has to update the transition rate matrix and corresponding one-step discrete transition matrix..
A pseudo-code of uniformization is not included in this report because this technique is not used to achieve any results.

Chapter 12

Schiphol as a Continuous Time Markov Chain

In this chapter it is described how arrivals, departures and the movement of aircraft at Schiphol airport can be modeled by a Continuous Time Markov Chain (CTMC). The goal of this approach is to estimate the expected demand for towing equipment on a day of operation. The motivation for this approach is given in chapter 10. First, the model is described by a step-by-step introduction of all events in section 12.1. Section 12.2 contains a description of how transition rates can be deducted from flight schedule data and OPiuM curves describing the duration of a towing task.

12.1 Schiphol CTMC Model

In this section it is aimed to build a CTMC that can be used to model aircraft arrivals and departures at AMS Schiphol Airport in order to estimate the expected demand for towing equipment. The final result, a CTMC with state space $S \subset \mathbb{Z}_+^8$ and transition rates as described by (12.2), is hard to grasp without sufficient understanding of the underlying processes. Therefore, the following subsections build the model step-by-step from scratch and these steps will subsequently be introduced. First, the modeling assumptions are summarized in section 12.1.1. Then, a simple model is initiated that only accounts for aircraft arrivals and departures. This model will then step-wise be extended with ground handling processes, gate capacity and towing tasks such that the desired model arises.

12.1.1 Modeling Assumptions

In order to model the movement of aircraft on Schiphol airport as a continuous time Markov chain, some assumptions have to be made. These assumptions include generalizations of handled aircraft in order to simplify the model as well as the required assumptions of exponential distributed interarrival and processing times. The following assumptions are made in this model:

1. Only wide body (WIBO) aircraft (Category 6 - 9) are considered.
2. All aircraft are treated equally, independent of the airline, aircraft type or destination. This means that no distinction is made in aircraft of airlines that have a contract with GS and aircraft that do not require any ground handling by GS. Also, every WIBO aircraft can be placed on every WIBO gate.
3. Assumption 2 has as consequence that every arrived aircraft can be used for each scheduled departing flight.
4. The arrivals of aircraft follow a Poisson process such that the expected number of arrivals of this process corresponds with the scheduled number of arrivals.
5. The departures of aircraft follow a Poisson process such that the expected number of departures of this process corresponds with the scheduled number of departures.
6. The duration of arrival handling, departure handling and aircraft towing are all exponentially distributed.
7. No aircraft are assigned to a gate. Only the total gate occupation is considered.
8. Every tow to or from a parking position is handled equally, i.e. no distinction is made between different parking locations.

12.1.2 Arrival and Departure Processes

Let's start with a Schiphol CTMC model in its most simple form: one that only considers arrivals and departures of aircraft. Assume that the arrivals of aircraft follow a Poisson processes with time-dependent rates $\lambda_a(t)$. The process of departing aircraft can be seen as a Poisson process of arriving departure requests. This means that there is a process that pushes aircraft into the system (arrivals) and a process that pulls aircraft out of the system (departures). Examples of such Push-Pull network can be found in literature [30]. Let the rate of the departure (i.e. pull) process be $\lambda_d(t)$.
With this reasoning, Schiphol can be compared with a shop where products (*aircraft*) are being processed and customers (*departure requests*) arrive in order to purchase a product. The number of aircraft at AMS can therefore be modeled by a time-inhomogeneous Birth-Death process (figure 12.1) with state space $S \subseteq \mathbb{Z}_+ = \{0, 1, 2, \dots\}$ and time dependent transition rates

$$q_t(n, n') = \begin{cases} \lambda_a(t) & (n' = n + 1) \\ \lambda_d(t)\mathbb{1}_{\{n \geq 1\}} & (n' = n - 1) \end{cases} \tag{12.1}$$

Here the time dependent rates are used to model the fluctuations of numbers of arrivals and departures over the day as a result of the bank system (see figure 2.2).
In the representation of transition classes (definition 11.5), the transition rates can be defined by two classes η_1 and η_2 corresponding to the events of an arrival and departure respectively

Whenever a departure request arrives while there is no aircraft available, the request will be discarded and ignored. However, it might be of interest to include pull requests that occur without any aircraft available to depart.

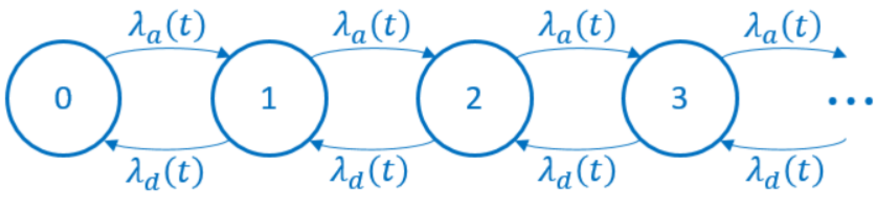


FIGURE 12.1: Birth-Death process of aircraft arrivals and departures at AMS.

Event	i	Guard G_i	Change Vector ν_i	Transition Rate $\alpha_i(n,t)$
arrival	1	$n \in S$	$+1$	$\lambda_a(t)$
departure	2	$n \geq 1$	-1	$\lambda_d(t)$

TABLE 12.1: Transition classes for airport Arrival-Departure (Birth-Death) process.

12.1.3 Arrival and Departure Processes with Departure Queue

An extra dimension can be added to the model in order to create a queue of departure requests that will be fulfilled as soon as a new aircraft arrives. At this point it is unrealistically assumed that an aircraft can depart directly after its arrival. Such a process, sketched in figure 12.2, has a 2 dimensional state space $S = \{\boldsymbol{n} \in \mathbb{Z}_+^2 : n_1 = 0 \vee n_2 = 0\}$ and transition classes described in table 12.2. Here the change vector $\boldsymbol{\nu}$ is represented as a sum of unit vectors e_i that have only 0 elements except for the i-th element that is equal to 1. The guard is given as a set of conditions that has to hold for state $\boldsymbol{n}$ to be contained in the guard of an event. From this point it is decided to describe the transitions as transition classes instead of the conventional transition description like in (12.1). The reason is that it is easier to present these transitions and to link them with the corresponding activity.

Event	i	Guard G_i	Change $\boldsymbol{\nu_i}$	Rate $\alpha_i(\boldsymbol{n},t)$
arrival, no dep. queue	1	$n_2 = 0$	e_1	$\lambda_a(t)$
arrival, dep. queue	2	$n_2 \geq 1$	$-e_2$	$\lambda_a(t)$
departure, aircraft available	3	$n_1 \geq 1$	$-e_1$	$\lambda_d(t)$
departure, no aircraft available	4	$n_1 = 0$	e_2	$\lambda_d(t)$

TABLE 12.2: Transition classes for airport Arrival-Departure (Birth-Death) process with infinite departure demand queue.

12.1.4 Adding a Ground Handling Process

Of course, aircraft require ground handling when they are at an airport. This ground handling consist of processes like passenger boarding and deboarding, baggage loading and unloading, and cleaning the interior of the aircraft. These processes can be grouped into arrival and departure related processes.
An aircraft will enter its arrival handling phase directly after arrival. When this process is finished, it will enter a possible waiting phase until it is triggered to enter its departure handling phase. After finishing the departure handling, the aircraft can leave the airport (figure 12.3).

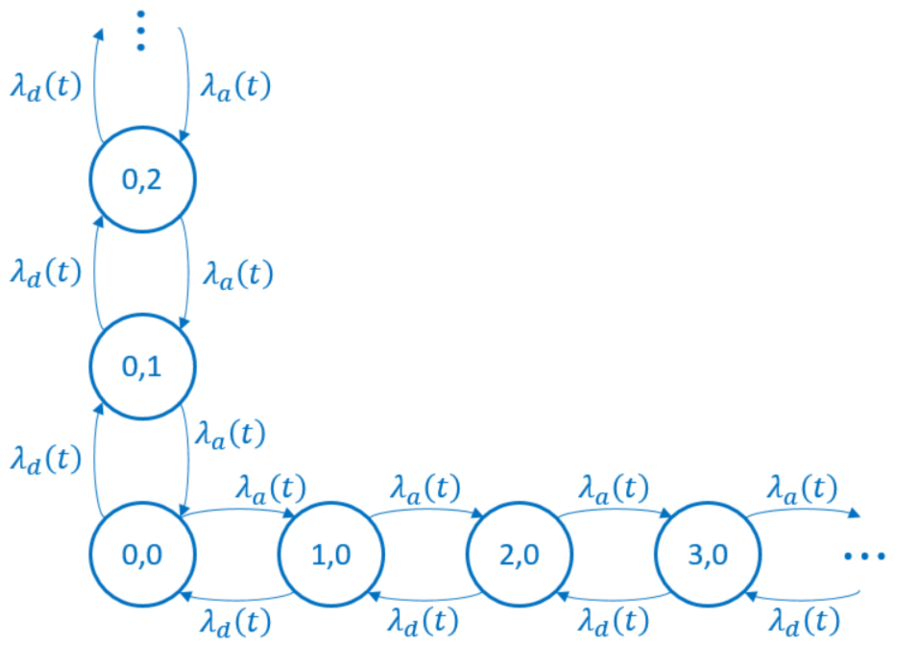


FIGURE 12.2: Birth-Death process of aircraft arrivals and departures at AMS including a queue for departure requests.

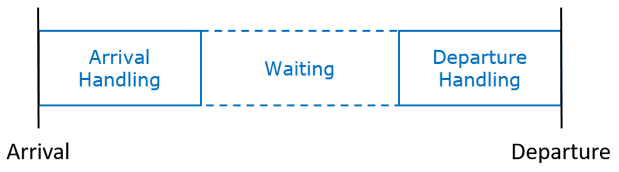


FIGURE 12.3: Phases of a single aircraft in the Arrival-Departure CTMC model including ground handling. The dashed waiting phase is optional.

A new problem that arises is that it has to be modeled when the departure process starts. This can be done by shifting the arrivals of departure requests with x time units. The idea is that it is known x minutes in advance that a departure request will come in and that the departure handling process can start.
Let $S = \{\boldsymbol{n} \in \mathbb{Z}_+^4 : n_2 = 0 \vee n_4 = 0\}$ be the state space of the CTMC that models this process. For state $\boldsymbol{n} \in S$, n_1 stands for the number of aircraft in the arrival handling phase, n_2 for the number of aircraft in the waiting phase, n_3 denotes the number of aircraft in the departure handling phase, and n_4 is used as the departure requests queue that grows when shifted departure requests come in while there are no aircraft available for departure handling.
Again we have Poisson arrival and departure processes with rates $\lambda_a(t)$ and $\lambda_d(t)$. If we assume that the arrival and departing handling processes follow an exponential distribution with rates μ_a and μ_d respectively, we can start describing the transition classes.

Event	i	Guard G_i	Change $\boldsymbol{\nu}_i$	Rate $\alpha_i(\boldsymbol{n}, t)$
arrival	1	$\boldsymbol{n} \in S$	e_1	$\lambda_a(t)$
finishing arrival handling, no queue	2	$n_1 \geq 1, n_4 = 0$	$e_2 - e_1$	$n_1\mu_a(t)$
finishing arrival handling, with queue	3	$n_1 \geq 1, n_4 \geq 1$	$e_3 - e_1 - e_4$	$n_1\mu_a$
dep handing request, aircraft available	4	$n_2 \geq 1$	$e_3 - e_2$	$\lambda_d(t+x)$
dep handing request, no aircraft available	5	$n_2 = 0$	e_4	$\lambda_d(t+x)$
dep handing end	6	$n_3 \geq 1$	$-e_3$	$n_3\mu_d$

TABLE 12.4: Transition classes for airport Arrival-Departure model including ground handling.

12.1.5 Including Gate Capacity and Towing Tasks

The model in the previous section does not take gate capacity into account. Realistically, there is a certain capacity of gates C_g. When the total number of aircraft that is modeled to be at a gate exceeds this number, an aircraft that is in its waiting phase has to be towed away to a parking location in order to create gate space for aircraft that have to be handled. An aircraft can only be towed away if it is not currently in its arrival or departure handling phase. Now there is a probability that an arriving aircraft has to wait before it can go into its arrival handling phase, simply because there are no gates available and no aircraft can be towed away. This results in the possible ground phases of an aircraft as in figure 12.4.

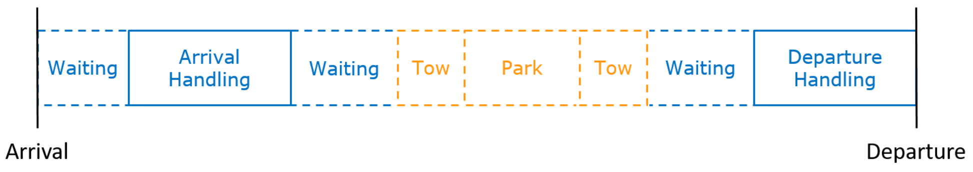


FIGURE 12.4: Phases of a single aircraft in the Arrival-Departure CTMC model including ground handling and towing possibilities. The dashed phases are optional.

Assuming that towing time is exponentially distributed with some parameter μ_{tow}, modeling this behaviour requires 4 extra state dimensions such that the total state space becomes $S \subseteq \mathbb{Z}_+^8$ with

n_1	Number of aircraft that is waiting to go into arrival handling phase.
n_2	Number of aircraft in arrival handling.
n_3	Number of aircraft waiting for a departure handling.
n_4	Number of aircraft that is being towed from a gate to a parking position.
n_5	Number of aircraft at a parking position.
n_6	Number of aircraft towed from a parking position to a gate.
n_7	Number of aircraft in departure handling.
n_8	Number of departure requests in queue waiting for an available aircraft to go into departure handling.

Here for every state either n_3 or n_8 is empty. If, by some event, the number of aircraft at a gate - i.e. $(n_2 + n_3 + n_7)$ - will exceed gate capacity C_g, an aircraft that is waiting for a departure handling will be towed away to a parking position. It is, however, harder to determine when an aircraft has to be towed back to the gate. There are two possible methods to model this decision making:

1. Start towing aircraft back to the gate when the total sum of aircraft that are currently at a gate or being towed back to a gate - i.e. $(n_2 + n_3 + n_6 + n_7)$ - drops below a certain capacity C_t.
2. Let the moments of starting a tow back to the gate follow a Poisson process with a parameter that depends on both a certain capacity C_t and the known arrival and departure rates $\lambda_a(t)$ and $\lambda_d(t)$ by some function $\Phi(t)$.

Here, $\Phi(t)$ can be used to model the "urge" of towing aircraft back to the gate since the incoming arrivals are likely not to satisfy the coming demand for departures. Option 1 is favoured since it does not add that much extra complexity to the model.
The following events provoke a state change.

Arrival

1. *No capacity problem*: Arriving aircraft goes into its arrival handling phase.
2. *Capacity problem, aircraft available for tow*: Arriving aircraft goes into its arrival handling phase, a waiting aircraft is being towed to parking location.
3. *Capacity problem, no aircraft available for tow*: Aircraft goes into the queue, waiting for an available gate.

Finished Arrival Handling

4. *No departure requests in queue, no aircraft waiting for arrival handling*: Aircraft will go into waiting phase.
5. *No departure requests in queue, some aircraft are waiting for arrival handling*: Aircraft that finishes arrival handling will be towed away immediately in order to create space for an aircraft that is currently waiting for arrival handling.
6. *Departure requests in queue*: Aircraft that finishes arrival handling goes straight into its departure handling phase and queue decreases by one.

(Shifted) Departure Request

7. *Aircraft available*: Aircraft goes from waiting phase into departure handling.
8. *No aircraft available*: Departure request is added to request queue.

Finished Tow to Park

9. Aircraft enters parking position.

Finished Tow to Gate

10. *No capacity problem, no demand request queue*: Aircraft enters waiting phase.
11. *No capacity problem, with demand request queue*: Aircraft will enter departure phase and queue decreases by one.
12. *Capacity problem*: At this point, the tow back to the gate shouldn't have started in the first place. However, an aircraft is towed to the gate while there is no gate available. The only way to solve this problem is to tow the aircraft back to a parking location.

Finished Departure Handling

13. *No aircraft waiting for arrival handling and no aircraft at park or the number of aircraft at gate remains above towing back capacity limit*: Aircraft that finishes departure handling leaves.
14. *Aircraft waiting for arrival handling*: Aircraft that finishes departure handling leaves, waiting aircraft goes into arrival handling.
15. *Aircraft available for tow and total number of aircraft at gate drops below the tow back capacity limit*: Start towing aircraft back to gate.

These events can be modeled by the transition classes in table 12.6 which correspond to the conventional representation of these transitions as presented in (12.2). Appendix C contains a description of these events and corresponding transition classes when the capacity of tow trucks is include. It shows that this capacity limit increases the complexity of the model.

i	Guard G_i	Change $\boldsymbol{\nu}_i$	Rate $\alpha_i(\boldsymbol{n}, t)$
1	$n_2 + n_3 + n_7 < C_g$	e_2	$\lambda_a(t)$
2	$n_2 + n_3 + n_7 = C_g,\ n_3 \geq 1$	$e_2 - e_3 + e_4$	$\lambda_a(t)$
3	$n_2 + n_7 = C_g,\ n_3 = 0$	e_1	$\lambda_a(t)$
4	$n_2 \geq 1,\ n_1 = 0,\ n_8 = 0$	$e_3 - e_2$	$n_2\mu_a$
5	$n_2 \geq 1,\ n_1 \geq 1,\ n_8 = 0$	$e_4 - e_1$	$n_2\mu_a$
6	$n_2 \geq 1,\ n_8 \geq 1$	$e_7 - e_2 - e_8$	$n_2\mu_a$
7	$n_3 \geq 1$	$e_7 - e_3$	$\lambda_d(t+x)$
8	$n_3 = 0$	e_8	$\lambda_d(t+x)$
9	$n_4 \geq 1$	$e_5 - e_4$	$n_4\mu_{tow}$
10	$n_2 + n_3 + n_7 < C_g,\ n_6 \geq 1,\ n_8 = 0$	$e_3 - e_6$	$n_6\mu_{tow}$
11	$n_2 + n_3 + n_7 < C_g,\ n_6 \geq 1,\ n_8 \geq 1$	$e_7 - e_6 - e_8$	$n_6\mu_{tow}$
12	$n_2 + n_3 + n_7 = C_g,\ n_6 \geq 1$	$e_4 - e_6$	$n_6\mu_{tow}$
13	$n_7 \geq 1,\ n_1 = 0,\ n_5 = 0 \vee n_2 + n_3 + n_6 + n_7 > C_t$	$-e_7$	$n_7\mu_d$
14	$n_7 \geq 1,\ n_1 \geq 1$	$e_2 - e_1 - e_7$	$n_7\mu_d$
15	$n_7 \geq 1,\ n_1 = 0,\ n_5 \geq 1,\ n_2 + n_3 + n_6 + n_7 \leq C_t$	$e_6 - e_5 - e_7$	$n_7\mu_d$

Table 12.6: Transition classes for airport Arrival-Departure model including ground handling and towing.

$$q_t(\boldsymbol{n},\boldsymbol{n}') = \begin{cases} \lambda_a(t)\mathbb{1}_{(n_2+n_3+n_7<C_g)} & (\boldsymbol{n}' = \boldsymbol{n} + e_2) \\ \lambda_a(t)\mathbb{1}_{(n_2+n_3+n_7=C_g,\ n3\geq 1)} & (\boldsymbol{n}' = \boldsymbol{n} + e_2 - e_3 + e_4) \\ \lambda_a(t)\mathbb{1}_{(n_2+n_7=C_g,\ n3=1)} & (\boldsymbol{n}' = \boldsymbol{n} + e_1) \\ n_2\mu_a\mathbb{1}_{(n_2\geq 1,\ n_1=0,\ n_8=0)} & (\boldsymbol{n}' = \boldsymbol{n} + e_3 - e_2) \\ n_2\mu_a\mathbb{1}_{(n_2\geq 1,\ n_1\geq 0,\ n_8=0)} & (\boldsymbol{n}' = \boldsymbol{n} + e_4 - e_1) \\ n_2\mu_a\mathbb{1}_{(n_2\geq 1,\ n_8\geq 1)} & (\boldsymbol{n}' = \boldsymbol{n} + e_7 - e_2) \\ \lambda_d(t)\mathbb{1}_{(n_3\geq 1)} & (\boldsymbol{n}' = \boldsymbol{n} + e_7 - e_3) \\ \lambda_d(t)\mathbb{1}_{(n_3=0)} & (\boldsymbol{n}' = \boldsymbol{n} + e_8) \\ n_4\mu_{tow}\mathbb{1}_{(n_4\geq 1)} & (\boldsymbol{n}' = \boldsymbol{n} + e_5 - e_4) \\ n_6\mu_{tow}\mathbb{1}_{(n_2+n_3+n_7<C_g,\ n_6\geq 1,\ n_8=0)} & (\boldsymbol{n}' = \boldsymbol{n} + e_3 - e_6) \\ n_6\mu_{tow}\mathbb{1}_{(n_2+n_3+n_7<C_g,\ n_6\geq 1,\ n_8\geq 1)} & (\boldsymbol{n}' = \boldsymbol{n} + e_7 - e_6 - e_8) \\ n_6\mu_{tow}\mathbb{1}_{(n_2+n_3+n_7=C_g,\ n_6\geq 1)} & (\boldsymbol{n}' = \boldsymbol{n} + e_4 - e_6) \\ n_7\mu_d\mathbb{1}_{(n_7\geq 1,\ n_1=0,\ n_5=0\wedge n_2+n_3+n_6+n_7>C_t)} & (\boldsymbol{n}' = \boldsymbol{n} - e_7) \\ n_7\mu_d\mathbb{1}_{(n_7\geq 1,\ n_1\geq 1)} & (\boldsymbol{n}' = \boldsymbol{n} + e_2 - e_1 - e_7) \\ n_7\mu_d\mathbb{1}_{(n_7\geq 1,\ n_1=0,\ n_5\geq 0,\ n_2+n_3+n_6+n_7\leq C_t)} & (\boldsymbol{n}' = \boldsymbol{n} + e_6 - e_5 - e_7) \end{cases} \tag{12.2}$$

12.1.6 Reward Structure

Reward vectors can be used to analyze the result of an instance of the Schiphol CTMC model as described in section 12.1. These rewards, that are introduced in section 11.4, can measure a specific metric over the course of a day that is evaluated by for example time discretization. In this study there is a specific interest in the number of towing equipment that is necessary in order to carry out a schedule. In order to evaluate this metric, the following reward vector can be used.

$$r_{num\_tow}(\mathbf{n}) = n_4 + n_6 \tag{12.3}$$

Here the value of the element of the reward vector that corresponds to state n is described. Taking the dot product of this reward vector and the state probability vector will result in the expected number of towing trucks that are necessary for towing aircraft to and from gates.

$$E[num\_tow]_t = \langle r_{num\_tow}, \pi_t \rangle \tag{12.4}$$

Other rewards that might be of interest are the expected number of aircraft in arrival handling, number of aircraft in departure handling, number of aircraft at a gate waiting for a departure handling, number of aircraft at a parking location, number of aircraft that have to queue up because there are no gates available and the number of departure requests in queue that are waiting for an aircraft to be available. These rewards are described by the following reward vectors.

$$\begin{aligned} r_{num\_arr\_hand} &= n_2 \\ r_{num\_dep\_hand} &= n_7 \\ r_{num\_wait\_gate} &= n_3 \\ r_{num\_park} &= n_5 \\ r_{len\_arr\_queue} &= n_1 \\ r_{len\_dep\_request\_queue} &= n_8 \end{aligned} \tag{12.5}$$

These rewards can all be evaluated at time t by taking the dot product with the current state distribution vector as shown in (12.4).

12.2 Retrieving Transition Rates From Data

In order to model the movement of aircraft on Schiphol for a given flight schedule as described in section 12.1, the transition rates for that instance have to be obtained. These transition rates can be grouped in arrival streams (aircraft arrival rate and departure request rate) and process rates (handling process rates and towing rate).

12.2.1 Arrival and Departure Rates

The rates for arriving and departing aircraft can be directly determined from a flight schedule. Figure 12.5 shows the scheduled arrival and departure pattern of wide body aircraft on an average day on Schiphol.

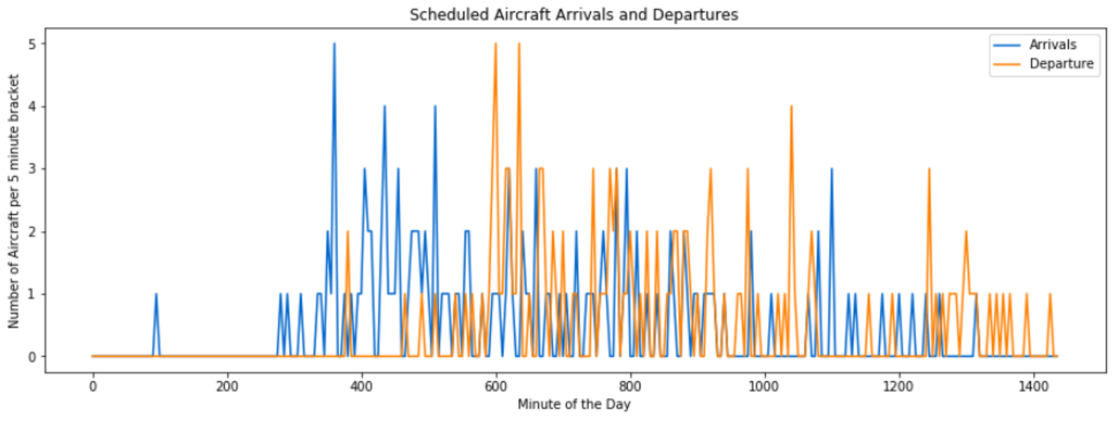


FIGURE 12.5: Example of scheduled arrivals and departures per 5 minute time brackets of wide body aircraft on one day on Schiphol airport.

The goal of this part of this thesis is not to model aircraft arrival and departure delays. Instead, there is more interest in the effect of the rate of arriving and departing aircraft on the usage of gate and tow truck capacity. Therefore, no flight time distributions are generated but the

scheduled arrivals and departures are translated to arrival and departure rates directly.
Let $n_a(t_i)$ be the number of scheduled arrivals in 5 minute interval i. The arrival rate per minute for that interval is given by

$$\lambda_a(t_i) = \frac{1}{5} \sum_{j=i-2}^{i+1} \frac{n_a(t_j)}{4} \tag{12.6}$$

In words: the number of arrivals in an interval are converted to an arrival rate that spans over the 5 minute interval in which the arrivals take place, the previous 5 minute interval and the following two 5 minute intervals. The arrival rate corresponding to these arrivals is distributed uniformly over these four 5 minute intervals (divide number of arrivals by the four intervals). To achieve the rate per minute per interval, the sum of all the rates that overlap this interval has to be multiplied by $\frac{1}{5}$.
The influence on the rates of each arrival and departure are spread out over multiple intervals in order to model the fact that an aircraft mostly arrives somewhere close to its scheduled time. It is not stated that this accounts for the expected delay of flights since that is not the goal of this study.
The departure rates can be obtained in a similar way. However, as stated in section 12.1.4, this rate has to be shifted with the expected duration of the departure handling phase such that it can be used as the rate at which aircraft have to enter departure handling. For this instance the expected duration of the departure handling is 85 minutes.

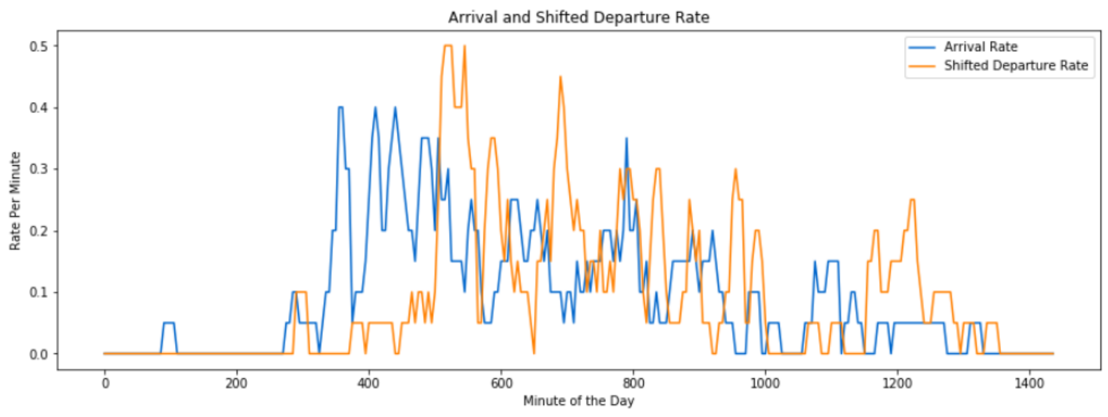


FIGURE 12.6: Arrival and departure rates corresponding to the arrivals and departure of figure 12.5 obtained by applying (12.6)

Figure 12.6 shows the arrival and departure rates as a result of applying (12.6) on the arrivals and departures in figure 12.5. The variance of these rates is pretty high and each transition rate change will result in updating a large transition matrix. Therefore the average rate per 15 minutes is used in order to smooth curve and decrease the number of transition rate changes. Figures 12.7 and 12.8 show the effects of such a smoothening.

12.2.2 Rates of Ground Processes

The rates of the arrival handling process, the departure handling process and the towing process can be deducted from operational data. This deduction is done regularly in order to generate input for OPiuM (section 7.2). Therefore the rates that will be used for this CTMC are based

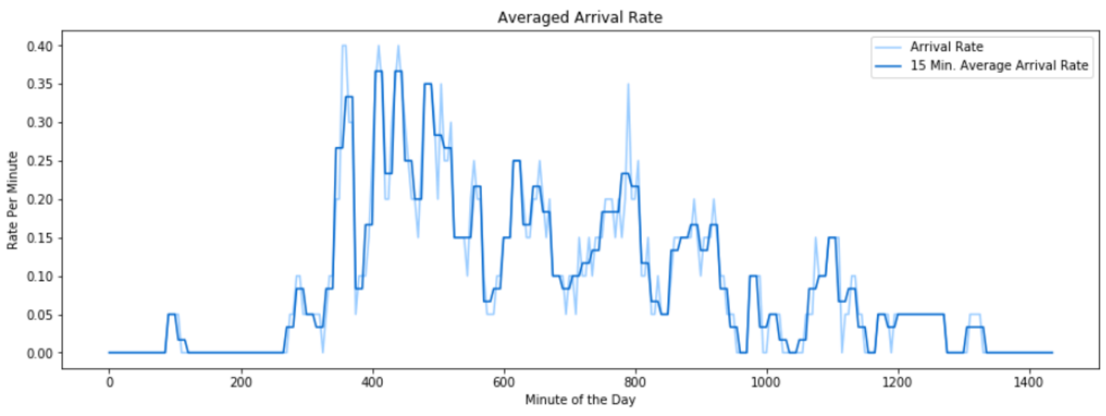


FIGURE 12.7: Arrival rate and its 15 minute average.

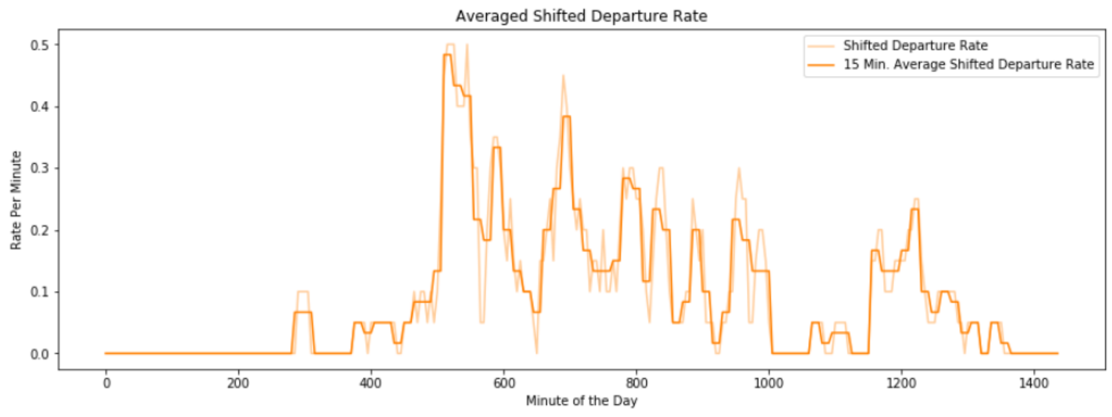


FIGURE 12.8: Shifted departure rate and its 15 minute average.

on the already known OPiuM curves.
In continuous-time Markov modeling, processing times are assumed to be exponentially distributed. It is clear that realistically the towing time of a wide body aircraft from a gate to a parking position does not follow an exponentially distribution. Figure 12.9 (a.) shows the fit of an exponential distribution with parameter 80. However, an Erlang distribution with shape parameter 30 and rate 2.45 shows a very good fit (figure 12.9 (b.)). This result can be achieved by modelling the towing process as a series of 30 consecutive Exponentially distributed towing phases with rate 2.45. This will make the model more realistic but will also increase the dimensionality of the state space with 29. Such an increment of the state space is disastrous when the Model is aimed to be analyzed using discretization. Therefore, the less realistic Exponential approximation is preferred over using the more extensive Erlang distribution in order to preserve the solvability of the CTMC instance.

Analysis of the duration of arrival and departure handling shows the same results as shown for the duration of towing an aircraft from a gate to a parking location (figure 12.9). Therefore it is assumed that the duration of all ground processes are exponentially distributed with an average as observed in operation, i.e. average of OPiuM curves. This results in the following rates

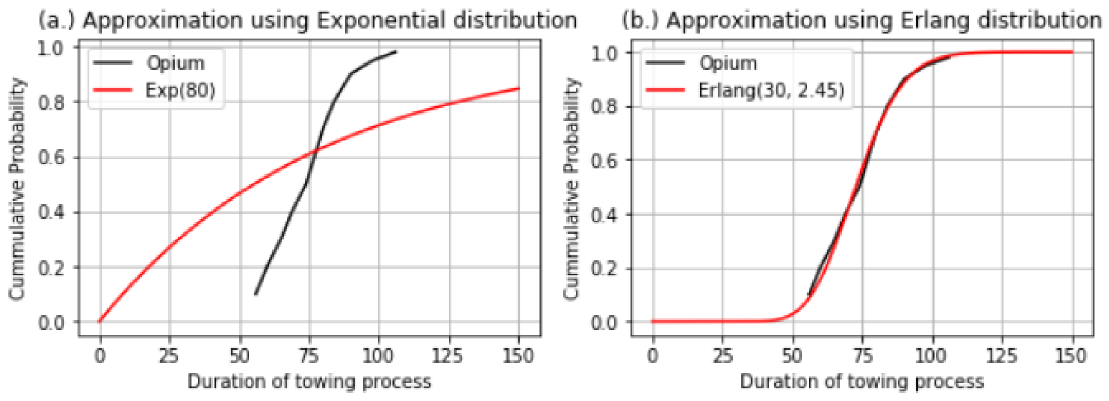


Figure 12.9: Approximation of the OPiuM cumulative probability curve of the duration of a tow task for wide body aircraft. Approximated using (a.) Exponential distribution and (b.) Erlang distribution.

Arrival Handling Rate	$\mu_a = 1/50$
Departure Handling Rate	$\mu_d = 1/85$
Towing Rate	$\mu_{tow} = 1/80$

Chapter 13

An Algorithm to Solve Large Time-Inhomogeneous CTMCs

The Schiphol CTMC, as described in chapter 12, is a Markov chain with a theoretical infinite state space and time-inhomogeneous transition rates. Because of these characteristics the behaviour of this model cannot be easily analyzed by, for example, a product form solution [27]. Therefore other solution approaches like discretization (section 11.2) or uniformization (section 11.3) can be used. Because of the high fluctuations in transition rates of this model, the static form of truncation as described in section 11.6 might cut of states that will be of high importance when transition rates change. Therefore a more dynamic approach of truncation is introduced.
The idea of discretization using iterative state exploration and dynamic state space truncation is proposed in section 13.1. A version of discretization using static state space truncation is sketched in section 13.2. The concepts of iterative state exploration and dynamic state space truncation are described in more detail in sections 13.3 and 13.4 respectively. Applying a similar approach on uniformization is briefly described in section 13.5. Finally, section 13.7 contains a description of the implementation of the proposed algorithm in programming language Python.

13.1 Iterative State Exploration and Dynamic State Space Truncation

Solving time-inhomogeneous CTMCs with a large or infinite state space, like the one described in section 12.1.5, can be hard because of memory issues. Despite the traditional static truncation, where states are eliminated from the analysis in advance, it can be an advantage to consider a more dynamical approach of truncation, where states are eliminated from the analysis when they are not of importance anymore.
This section describes methods of dynamical state space truncation while iteratively solving large CTMCs using time-discretization. The idea is to start with a small state space. As time progresses, new states can be explored that will be added to the analysis. After a while the total

found state space is evaluated and states with a small probability to be reached at that moment will be discarded from the state space, leaving a truncated state space small enough to further evaluate.
Idea's for such models have only been found in a 2010 Master Thesis [5]. Apart from that, there are no examples found of such dynamic truncation methods or its application on large time-inhomogeneous CTMCs.

13.2 Truncated Time-Discretization

Consider a time-inhomogeneous CTMC with large or infinite state space S and initial distribution vector π_0. Now let the truncated version of this model have initial state space

$$\bar{S}_0 = \{i \in S : \pi_0(i) > 0\} \tag{13.1}$$

with current distribution vector containing only positive elements of the original initial distribution vector

$$\bar{\pi}_{d,0} = [\pi_0(i) \in \pi : \pi_0(i) > 0] \tag{13.2}$$

With this truncated form we have not lost any information since all states with positive probability at time zero are included in the model.
Now, as described in section 11.2, this model can be analyzed at discrete time steps of size h in time window $[0, T]$. Let the truncated version of the time-dependent one-step transition matrix be

$$\bar{\mathbf{M}}_t(i,j) = \begin{cases} q_t(i,j)h & j \neq i, \quad i,j \in \bar{S}_t \\ 1 - \sum_{l \in \bar{S}_t \setminus i} q_t(i,l)h & j = i, \quad i,j \in \bar{S}_t \end{cases} \tag{13.3}$$

where $\bar{S}_t$ is the set of states at time t. This truncated transition matrix can be used to describe a truncated version of discretization iteration scheme (11.12)

$$\begin{aligned} \bar{\pi}_{d,t+h} &= \bar{\pi}_{d,t}\bar{\mathbf{M}}_t \\ \bar{S}_{t+h} &= \bar{S}_t \end{aligned} \tag{13.4}$$

where $\bar{\pi}_{d,0}$ and $\bar{S}_0$ are initialized as in (13.1) and (13.2) respectively. Note that the truncated state space remains the same during an iteration. This means that the usage of such an truncated state space will eventually lead to some loss of information since transitions from $i \in \bar{S}_t$ to $j \in S_t \setminus \bar{S}_t$ are not included in the analysis. Therefore the truncated state space $\bar{S}_t$ can be extended when transitions to truncated states are about to take place. This state space exploration is described in section 13.3. After a while the state space can become too large and some states might be not relevant anymore. Section 13.4 describes how the state space can be dynamically truncated in order to decrease the number of states and to exclude irrelevant states.

13.3 Iterative State Space Exploration

When the probability mass has reached the border of the truncated version of time-discretization it will loose information compared to its non-truncated version because transition might occur to states that are not considered in the analysis. In order to not lose that information, the current truncated state space can be extended by a state exploration that looks for states that are reachable from the current state space. This exploration can be done by searching for states that could possibly be reached after some transitions.
Let $\Omega_v(\bar{S}_t)$ be the operation that explores the state space for the next v iterations.

$$\Omega_v(\bar{S}_t) = \bar{S}_t \cup \{j \in S | \exists i \in \bar{S}_t,\ k = 1, \ldots, v : \mathbf{M}_t^k(i,j) > 0\} \tag{13.5}$$

where the same operation can be applied on the state probability vector such that the element of the result $\Omega_v(\bar{\pi}_{d,t})$ that corresponds with state i can be defined as

$$\Omega_v(\bar{\pi}_{d,t})(i) = \begin{cases} \bar{\pi}_{d,t}(i) & i \in \bar{S}_t \\ 0 & i \in \Omega_v(\bar{S}_t) \setminus \bar{S}_t \end{cases} \tag{13.6}$$

After exploring the state space for v additional steps, one is sure that the truncated discrete state probability distribution $\bar{\pi}_{d,t}$ can be calculated using iterative scheme (13.4) for the next v iterations without making any error compared to the non-truncated variant, i.e.

$$\begin{aligned} \pi_{d,t}(i) &= \bar{\pi}_{d,t}(i) \quad & i \in \bar{S}_t \\ \pi_{d,t}(i) &= 0 & i \in S \setminus \bar{S}_t \end{aligned} \tag{13.7}$$

Therefore this iterative state exploration can be used to shrink the size of the problem without losing any information via transitions to truncated states. This process is visualized for a two dimensional state space in figure 13.1.

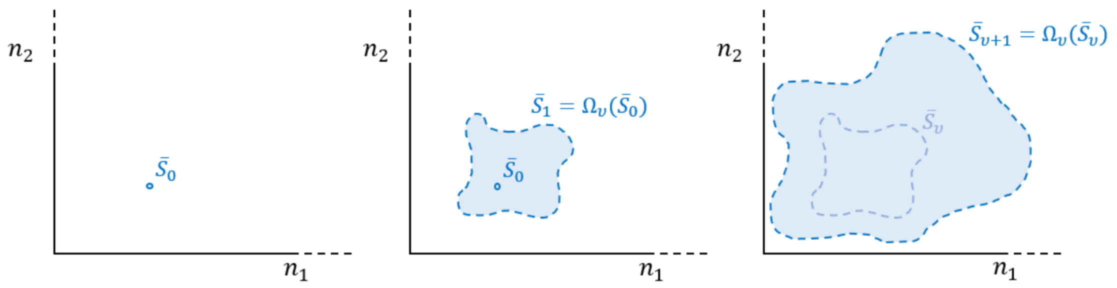

n_2
$\bar{S}_0$
n_1
n_2
$\bar{S}_1 = \Omega_v(\bar{S}_0)$
$\bar{S}_0$
n_1
n_2
$\bar{S}_{v+1} = \Omega_v(\bar{S}_v)$
$\bar{S}_v$
n_1

FIGURE 13.1: The process of state space exploration. The first figure shows the initial state space $\bar{S}_0$. New states are being explored in the second figure. After v iterations, the state space has to be expanded again (right figure).

Approximate State Space Exploration

Define states that have transitions to states that are not included in the truncated analysis as "border" states. This set of border states can be described as

$$\bar{S}_t^{Border} = \{i \in \bar{S}_t : \sum_{j \in S} \mathbf{M}_t(i,j) > 0\} \tag{13.8}$$

When the state space is being explored for the next υ transitions, one is sure that after υ iterations the border states will have a non-zero probability of being reached. Though we are sure that the border of the truncated state space will be reached with non-zero probability, this probability might not be significant enough to provoke further exploration. Therefore an approximated expansion can be used that only expands the state space when the states at the border of this state space have a probability mass higher than some probability threshold ς. Let $\widetilde{S}_t$ be the set of states that have a probability mass that is significantly higher than zero at time t.

$$\widetilde{S}_t = \{i \in S : \bar{\pi}_{d,t}(i) > \varsigma\} \tag{13.9}$$

Let the reached border of the explored state space then be described as

$$\widetilde{S}_t^{Border} = \{i \in \widetilde{S}_t : \sum_{j \in S \setminus \bar{S}} \mathbf{M}_t(i,j) > 0\} \tag{13.10}$$

In words: the reached border of the explored state space at time t is the set of states that at time t have a probability mass higher than ς and have a positive rate for some transitions to states that are not included in the current truncated state space.
This definition can be used to only provoke a state exploration whether $|\widetilde{S}_t^{Border}| > 0$, i.e. there are some reached border states. Furthermore, the state space only have to be explored from the states in this reached border set. This means that the approximated expansion becomes

$$\widetilde{\Omega}_\upsilon(\bar{S}_t) = \bar{S}_t \cup \{j \in S | \exists i \in \widetilde{S}_t^{Border},\ k = 1, \ldots, \upsilon : \mathbf{M}_t^k(i,j) > 0\} \tag{13.11}$$

with

$$\widetilde{\Omega}_\upsilon(\bar{\pi}_{d,t})(i) = \begin{cases} \bar{\pi}_{d,t}(i) & i \in \bar{S}_t \\ 0 & i \in \widetilde{\Omega}_\upsilon(\bar{S}_t) \setminus S_t \end{cases} \tag{13.12}$$

Figure 13.2 visualizes this approximated approach of state exploration.
Using this approximation will of course cause an error. Some states will have probability mass while having successors that are not included in the analysis. Therefore, equations (13.7) do not hold anymore. An indication of this error can be given by the total probability mass of border states in $\bar{S}_t^{Border}$ that are not included in the approximate border states $\widetilde{S}_t^{Border}$. A more elegant approach is to keep track of transitions from i to j with $i \in \bar{S}_t$ and $j \in S \setminus \bar{S}_t$. These transitions are omitted from the analysis and might take place with non-zero probability if $i \in \bar{S}_t^{Border} \setminus \widetilde{S}_t^{Border}$. This loss of probability flow can be tracked if all such transitions are modelled as transitions to some absorbing state ζ. This method requires more computations and if ς is chosen small enough, i.e. $\varsigma < 10^{-10}$ the error should be negligible. Algorithm 7 in appendix D describes iterative state exploration and includes the option of approximate expansion.

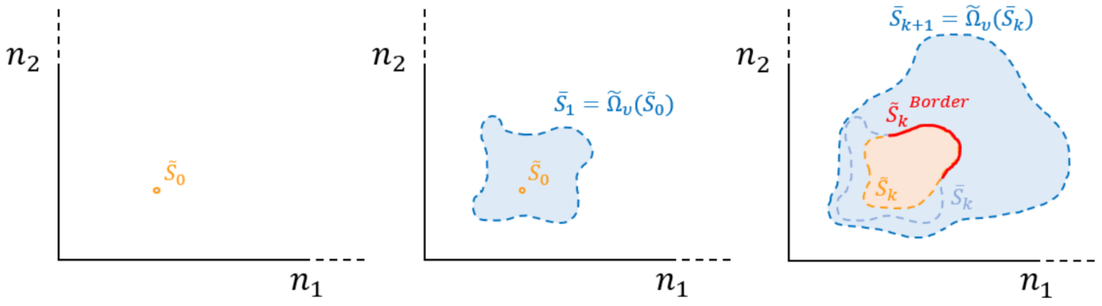

n_2
n_1
$\tilde{S}_0$
$\bar{S}_1 = \tilde{\Omega}_v(\bar{S}_0)$
$\bar{S}_{k+1} = \tilde{\Omega}_v(\bar{S}_k)$
$\tilde{S}_k$ Border
$\tilde{S}_k$
$\bar{S}_k$

FIGURE 13.2: The process of approximate state space exploration. The first figure shows the initial state space of relevant states $\widetilde{S}_0$. In this case it is exactly initial state space $\bar{S}_0$. New states are being explored in the second figure. After k iterations, $\widetilde{S}_k$ has reached the border of the current explored state space (right figure). This means that the border $\widetilde{S}_k^{Border}$ contains some states and a new state exploration is provoked.

Tuning State Space Exploration

Parameter υ, that indicates how many steps have to be iterated at every state exploration, can be tuned in order to decrease the runtime of the algorithm. When the parameter is too high, too many irrelevant states can be found. If the parameter is too low, the algorithm may perform poorly since the transition matrix has to be updated many times.
The number of steps to explore at every iteration might depend on some kind of exploration scheme. This can be compared with a cooling scheme for Simulated Annealing. After a state space truncation, it is probably best to slowly expand the state space in order for the algorithm to get some sense of relevant areas in the state space. After a while the algorithm might benefit from bigger state space explorations. Such a scheme can be defined as the number of expansion steps υ for the x-th expansion after the last truncation. Table 13.1 shows an example of such a state exploration scheme.

Number of explorations after last truncation	Exploration steps (υ)
1 - 10	1
11 - 20	2
21 - 30	3
31 +	5

TABLE 13.1: Example of a state exploration scheme.

13.4 Dynamic State Space Truncation

After a couple of state explorations the total explored state space might grow too large for analysis. It can also be the case that the state space contains states that are not relevant anymore. A state truncation can be considered when the number of states exceeds some state space threshold τ. At that point, all states that have a state probability that is lower than some truncation threshold ϵ can be omitted from the analysis.

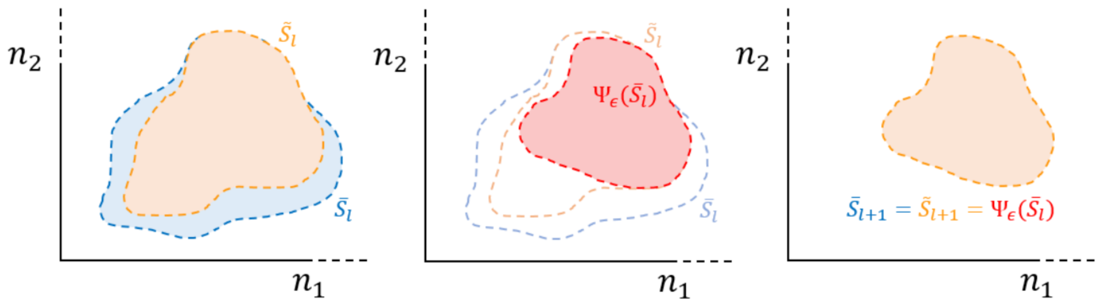
Let $\Psi_\epsilon(\bar{S}_t)$ be the operation that truncates state space $\bar{S}_t$ by

$$\Psi_\epsilon(\bar{S}_t) = \{i \in \bar{S}_t : \bar{\pi}_{d,t}(i) > \epsilon\} \tag{13.13}$$

Let the same operation be used to truncate the state probability vector

$$\Psi_\epsilon(\bar{\pi}_{d,t}) = [\bar{\pi}_{d,t}(i) \in \pi : \pi_{d,t}(i) > \epsilon] \tag{13.14}$$

This process is visualized for a two-dimensional state space in figure 13.3.



n_2
$\widetilde{S}_l$
$\bar{S}_l$
n_1
n_2
$\widetilde{S}_l$
$\Psi_\epsilon(\bar{S}_l)$
$\bar{S}_l$
n_1
n_2
$\bar{S}_{l+1} = \widetilde{S}_{l+1} = \Psi_\epsilon(\bar{S}_l)$
n_1

FIGURE 13.3: The process of dynamic state space truncation. The left figure shows that after l iterations the states with relevant probability mass $\widetilde{S}_l$ have reached the border of truncated state space $\bar{S}_l$. If the number of states has grown over threshold τ a truncation occurs. The middle figure shows all states that are left after this truncation. The remaining states form the new state space $\bar{S}_{t+1}$. From this state space all states are relevant (i.e. in $\widetilde{S}_{t+1}$).

One can keep track of the probability that the system will be in a state that is truncated to give an indication of the amount of information that is lost at a truncation step.

$$\textit{truncation loss} = \sum_{i \in \bar{S}_t} \bar{\pi}_{d,t}(i) - \sum_{j \in \Psi_\epsilon(\bar{S}_t)} \Psi_\epsilon(\bar{\pi}_{d,t})(j) \tag{13.15}$$

This truncation loss can be seen as the probability mass that is lost by the truncation. This mass is omitted from the analysis and it can be seen as if the CTMC contains an absorbing state where all this truncated probability mass is gathered. After truncation, the state probability vector might be normalized.
Algorithm 6 in appendix D describes this process of dynamic state truncation in full detail.

13.5 Uniformization and State Space Exploration and Truncation

The methods of state space exploration and state space truncation as described in this chapter apply directly on (time-inhomogeneous) discretization. In essence they can easily be translated to uniformization.
Assume that we have time interval (t_1, t_2) where transition rates do not change. Let $\bar{\pi}_{d,t_1}$ be the state probability vector at t_1 with current truncated state space $\bar{S}_{t_1}$. Now $\bar{\pi}_{d,t_2}$ can be calculated by the following steps

1. Determine the current truncated transition rate matrix $\bar{M}_t$ for state space $\bar{S}_t$ as in (11.19).
2. For this matrix, with corresponding uniformization rate B, determine value K to truncate the Poisson probability sum like (11.18).
3. Expand the state space for K iterations using $\Omega_K(\bar{S}_t)$ from equation (13.5).
4. Determine a new current truncated transition rate matrix $\bar{M}_t$ for state space $\Omega_K\bar{S}_t$ as in (11.19)
5. Calculate $P_{(t_1,t_2)}$ as in (11.17) using transition rate matrix $\bar{M}_t$ and Poisson probability truncation K.
6. $\bar{\pi}_{d,t_2} = \bar{\pi}_{d,t_1}P_{(t_1,t_2)}$

The only problem with this approach is that the Poisson probability sum is truncated at value K while not all transitions are known. It might be possible that the uniformization rate is higher after the state space expansion. This problem can be solved by either looping over step 2 to 4 until after some expansion the uniformization rate has not changed, or to increase K to prevent that too much Poisson probability is truncated.
The truncation of the state space is identical to the dynamic state space truncation for discretization as described in section 13.4
State space exploration and truncation are not further explored nor applied on a practical case in this thesis study. However, it might be an interesting case for further studies and implementations. A more general case of this idea for time-inhomogeneous CTMC with transition rates that change continuously is described in [5].

13.6 A Potential Analytic Bound on the Error of Dynamic Truncation

Using the approach of time-discretization in combination with dynamic state space truncation causes a certain error compared to the original continuous process. In order to get an indication of this error one has to take the following steps into account:

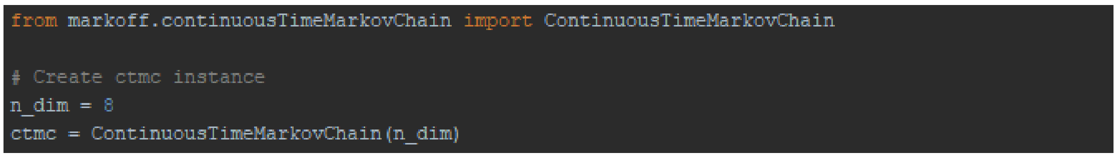
1. Time-discretization with step size h causes a first error. In fact, both cumulative rewards and the probability distribution have shown to be bounded by $O(h)$ [31].
2. The error on Markov rewards that is caused by static truncation can be bounded by a combination of state border probabilities and so called “mean first passage times” [4].
3. The error that is made by truncation - that can be bounded as stated in step 2 - occurs multiple times since the state space is truncated dynamically.
4. For this problem, truncation is applied on a time-inhomogeneous problem. The results described in step 2 are for time-homogeneous problems only. Since truncation is carried out sequentially, the effect of this truncation can be evaluated for intervals for which the transition rates do not change separately.

5. The number of steps until truncation depends on how fast the state space grows. It will probably be much easier to find a bound when the number of steps between truncating the state space is fixed.

By combining these five steps one may expect it to be provable that the error of reward r caused by time-discretization and dynamic truncation can be bounded by some function which is linear in h and ϵ and some constant that depends on T and B. Where B can be seen as a bound for reward r, i.e. $||r||_\infty < B$. This proof will be rather technical and theoretically complicated. Therefore this proof lies beyond the scope of this thesis.

13.7 Implementation

Programming language Python is used to implement the algorithm of discretization with iterative state exploration and dynamic truncation (algorithm 5 of appendix D) that is described in section 13.1. A library, named `markoff`, is built that consists of multiple classes that are used to describe Rates, Transitions, Rewards and States. These classes are used to form the Continuous Time Markov Chain class. Given an instance of this class, discretization algorithm (algorithm 5) can be run. Matrix computations are done by using the `numpy` and `scipy` libraries.
The library is constructed in such a way that an instance of a continuous-time Markov chain can easily be described. First a Continuous Time Markov Chain instance needs to be created with the desired number of dimensions (figure 13.4).

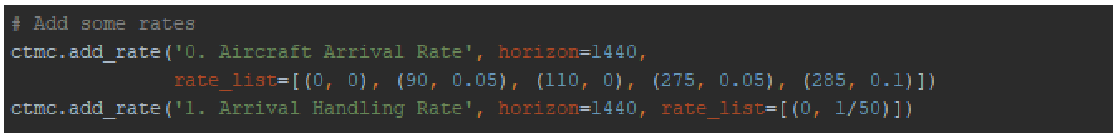


```
from markoff.continuousTimeMarkovChain import ContinuousTimeMarkovChain

# Create ctmc instance
n_dim = 8
ctmc = ContinuousTimeMarkovChain(n_dim)
```

FIGURE 13.4: Code snippet that shows how to create a CTMC instance.

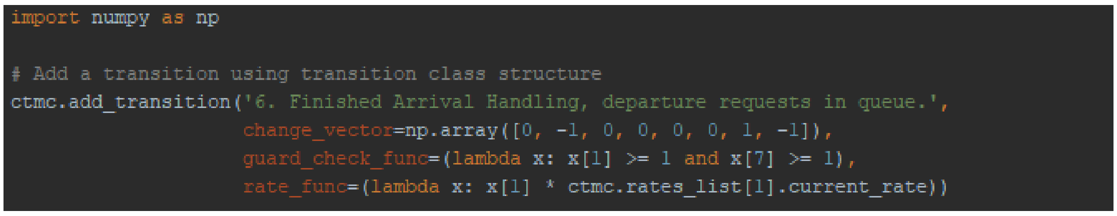
Rates can then be added to this CTMC instance. These rates require a name, a time horizon and a list of rate changes. The rate changes are described by a two-dimensional tuple `(time of rate change, new rate)`. In the example in figure 13.5 the Aircraft Arrival Rate is added that changes at some time points. The Arrival Handling Rate is constant and thus only has one "rate change" at $t = 0$.



```
# Add some rates
ctmc.add_rate('0. Aircraft Arrival Rate', horizon=1440,
              rate_list=[(0, 0), (90, 0.05), (110, 0), (275, 0.05), (285, 0.1)])
ctmc.add_rate('1. Arrival Handling Rate', horizon=1440, rate_list=[(0, 1/50)])
```

FIGURE 13.5: Code snippet that shows how to add rates to a CTMC instance.

The given rates can be used to construct transitions (see figure 13.6). These transitions are constructed using the transition class definitions (section 11.7) and require a name, a change vector as `numpy` array, a guard check function and a rate function.
The guard check function and rate function can be given by so called "anonymous functions"

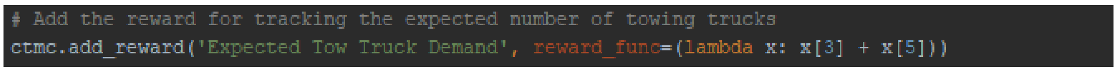


```
import numpy as np

# Add a transition using transition class structure
ctmc.add_transition('6. Finished Arrival Handling, departure requests in queue.',
                    change_vector=np.array([0, -1, 0, 0, 0, 0, 1, -1]),
                    guard_check_func=(lambda x: x[1] >= 1 and x[7] >= 1),
                    rate_func=(lambda x: x[1] * ctmc.rates_list[1].current_rate))
```

FIGURE 13.6: Code snippet that shows how to add transitions to a CTMC instance.

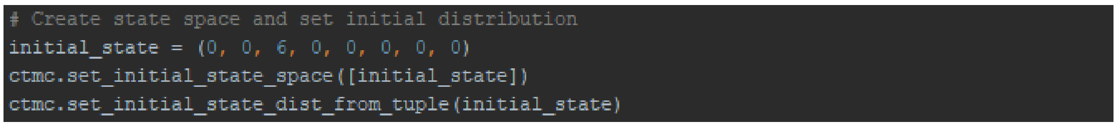
using the `lambda` keyword. Input `x` of these functions will always be a state in the form of a tuple describing the values for each dimension of this state, e.g. $(0, 8, 7, 0, 2, 0, 23, 0)$. The n-th element of this state tuple can then be used in the calculation by `x[n-1]`. When `guard_func` returns `True` for state `x`, the system will move to state `x + change_vector` with the rate that follows from `rate_func`. This rate function uses the defined Rates using `ctmc.rates_list[1].current_rate`, where 1 refers to the second rate that has been added since python starts indexing at 0.
Figure 13.7 shows that rewards can be added in a similar way. The added rewards will be calculated at every discrete time point that is evaluated. The reward of each state `x` is described by anonymous function `lambda` and the expected reward is calculated by the sum of the product of this reward and the state probability for all states.



```
# Add the reward for tracking the expected number of towing trucks
ctmc.add_reward('Expected Tow Truck Demand', reward_func=(lambda x: x[3] + x[5]))
```

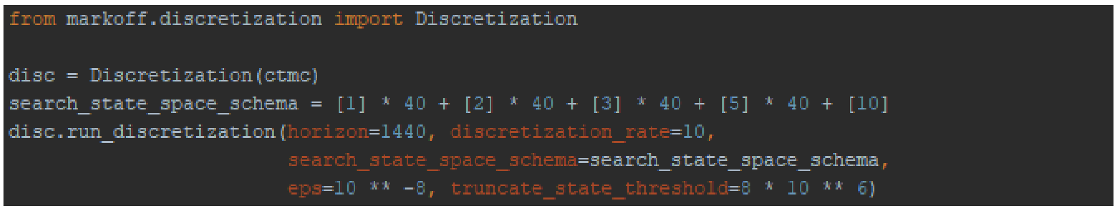
FIGURE 13.7: Code snippet that shows how to add rewards to a CTMC instance.

In order to evaluate the described CTMC instance, the initial distribution has to be defined. This can be done by giving a list of state tuples (figure 13.8). In this example the initial sate distribution is set from the same initial state tuple, which means that at $t = 0$ the probability of being in this state will be equal to 1.



```
# Create state space and set initial distribution
initial_state = (0, 0, 6, 0, 0, 0, 0, 0)
ctmc.set_initial_state_space([initial_state])
ctmc.set_initial_state_dist_from_tuple(initial_state)
```

FIGURE 13.8: Code snippet that shows how to add a initial sate space and define the initial distribution vector of a CTMC instance.



```
from markoff.discretization import Discretization

disc = Discretization(ctmc)
search_state_space_schema = [1] * 40 + [2] * 40 + [3] * 40 + [5] * 40 + [10]
disc.run_discretization(horizon=1440, discretization_rate=10,
                        search_state_space_schema=search_state_space_schema,
                        eps=10 ** -8, truncate_state_threshold=8 * 10 ** 6)
```

FIGURE 13.9: Code snippet that shows how to analyze a CTMC instance using discretization.

After the full description of a CTMC instance it can be given to the `Discretization` that is used to run the discretization algorithm. The `run_discretization` method requires a time horizon, a discretization factor and thresholds necessary to run algorithm 5. Here the schema for the

state space search is given as a list that describes the number of exploration steps as described in table 13.1.
Appendix E contains the full Python Code of this `markoff` library and can be received upon request.

Chapter 14

Results

This chapter contains the results of the Schiphol CTMC model as described in chapter 12 using time discretization with iterative state exploration and dynamic state space truncation as described in chapter 13. The model is run using different threshold settings which are described in table 14.1.

Run	State Threshold (τ)	Truncation Threshold (ϵ)	Exploration Scheme (υ)
1	2×10^6	10^{-7}	[1]*20 + [2]*20 + [3]*40 + [5]*40 + [10]
2	5×10^6	10^{-7}	[1]*10 + [2]*10 + [3]*20 + [5]*30 + [10]
3	8×10^6	10^{-8}	[1]*40 + [2]*40 + [3]*40 + [5]*40 + [10]

Table 14.1: Threshold settings for 3 different runs for calculating the rewards of the Schiphol CTMC model. For all runs a border probability threshold $\varsigma = 10^{11}$ is used.

The exploration scheme υ denotes the number of transitions that have to be explored at each state expansion. Here [1]*20 + [2]*10 + [5] means that the first 20 expansions after a truncation need to explore the state space for 1 transition, the next 10 expansion need to explore for 2 transitions, and all following expansions - until the next truncation - need to explore for 5 transitions (see the last paragraphs of section 13.3). These schemes are chosen based on a combination of runtime performance and the truncation loss that a scheme causes. However, it is not stated that these schemes are optimal (see appendix G for a short analysis of the effect of an exploration scheme). The Schiphol CTMC is evaluated for a single day according to the transition rates as described in section 12.2 using algorithm 5. Discretization factor $\beta = 10$ is used to let the step size of the discretization be 10 times smaller than the expected time between events. The most interesting reward, the expected number of tow trucks, is shown in figure 14.1. Other rewards are included in appendix F.

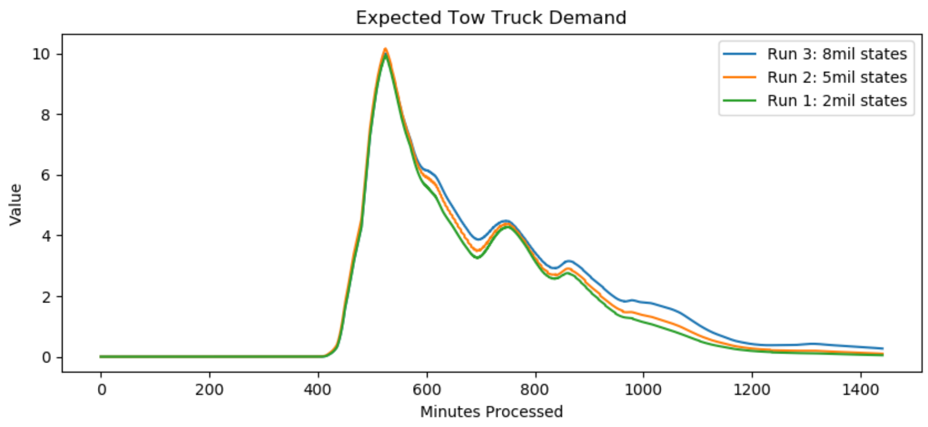


FIGURE 14.1: The expected demand for tow trucks according to the runs with threshold setting as in table 14.1.

This reward is an approximation since the model loses information by truncating the state space. Figure 14.2 shows the cumulative probability loss for the Schiphol CTMC. This graph shows the probability that, after a certain number of minutes of the day are processed, the system was in a state that has been truncated at some point. Figure 14.3 shows the number of states during processing a single day for run 3. The runtime progress for processing a full day is shown in figure 14.4

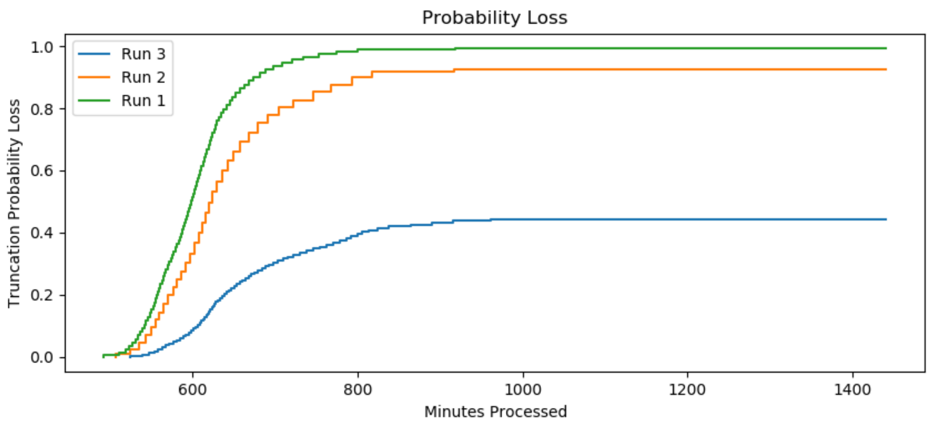


FIGURE 14.2: The cumulative probability loss of the runs with threshold setting as in table 14.1.

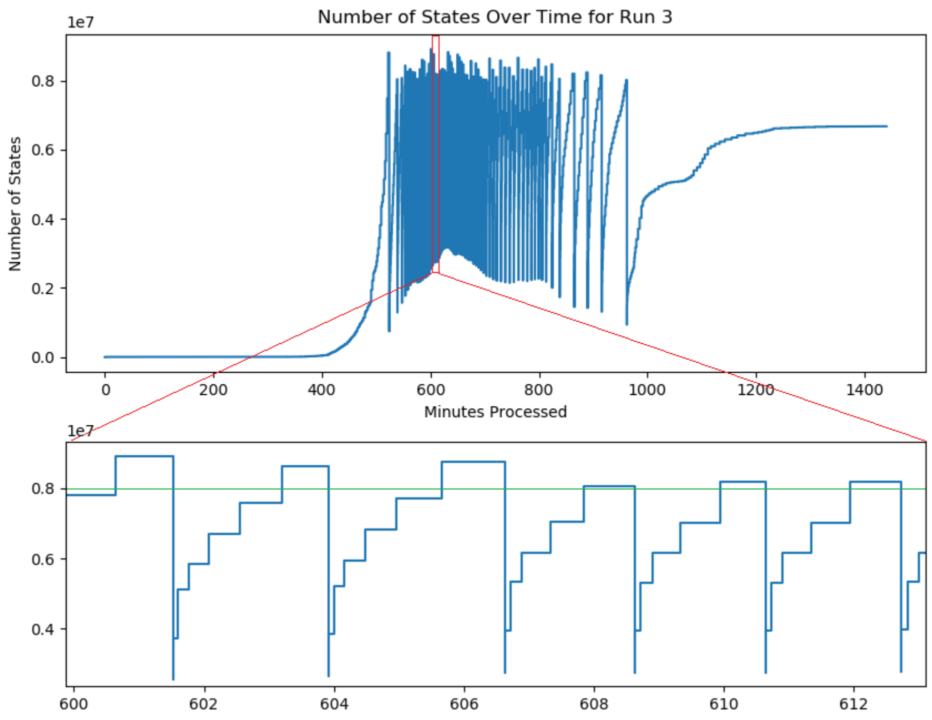


FIGURE 14.3: The number of states during processing a single day using threshold settings for run 3 in table 14.1.

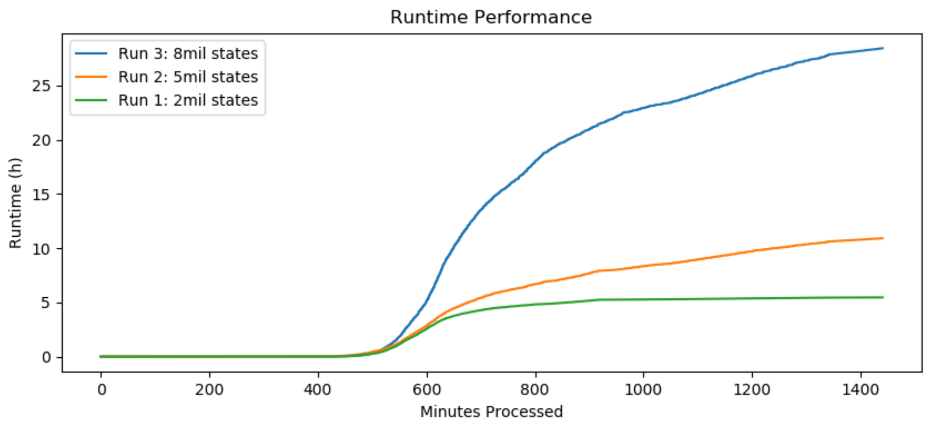


FIGURE 14.4: The runtime of each run described in table 14.1. Performed on a BTO Notebook with an Intel i7-6700HQ 2.6 GHz processor and 32GB RAM.

Chapter 15

Discussion and Conclusions

This chapter discusses the results of the Schiphol CTMC model (chapter 12) that is evaluated by discretization using iterative state space exploration and dynamic truncation (chapter 13). These results were presented in chapter 14. The discussion is contained in section 15.1 after which a conclusion is drawn in section 15.2. Furthermore, recommendations for the usage of these results for KLM and future studies are given in section 15.3.

15.1 Discussion

At first glance it seems to be possible to evaluate large time-inhomogeneous continuous time Markov chains as the Schiphol CTMC model using the proposed discretization algorithm. However, due to the size of the problem, using a state space maximum of 8 million states and a state probability threshold of $\epsilon = 10^{-8}$ is not enough to lower the truncation loss to a satisfying amount. With these threshold settings, the program ran 28.4 hours and the probability that the system was in a state that has been truncated still cumulates up to 44%.
The question is whether this total probability loss of 44% is that bad. There will always be some probability loss caused by truncation and this loss will only cause problems when the calculated rewards are influenced such that wrong conclusions are drawn. Figure 14.1 shows that there is not a significant difference in the expected tow truck demand between a state space threshold of 2, 5, and 8 million. Even though the truncation caused cumulated losses of 99%, 92% and 44% respectively.
For the run with a maximum of 2 million states, the program took 5.5 hour to complete (figure 14.4). This is a long time for evaluating a CTMC model knowing that there is a probability of 99% that the system was in a state that has been truncated. Though the rewards of the towing truck demand were close to the run with a maximum of 8 million states, which took 5 times longer.
KLM is most interested in the peaks of demand. With a state space threshold of 8 million states, the first 600 minutes of a day can be run in 4.6 hours with a cumulated probability loss of 8.3%. This might already be useful to indicate the expected peak demand. However, one is not sure

whether another demand peak might follow.
The result of this model is hard to validate. Assumptions are made that generalize all *widebody* aircraft on Schiphol. Since GS does not have contracts with all airlines, it does not provide towing trucks when some aircraft have to be towed away. Therefore comparison with actual or planned data cannot be made.

15.2 Conclusions

The time-inhomogeneous continuous time Markov chain model of the movements of wide body aircraft on Schiphol airport is presented in chapter 12. Because of the time-inhomogeneity and state dependency of transitions, this CTMC could not be evaluated by elegant analytical approaches. Therefore, a method of iterative state exploration and dynamic truncation has been developed to be able to estimate the characteristics of such large state time-inhomogeneous CTMCs.
It is shown that using the described approach for analyzing the Schiphol CTMC results in a program that can estimate the expected tow truck demand for a single day in 5.5 hours. This model assumes exponentially distributed processing and inter-arrival times and does not distinguish between airline, aircraft type, special cases of aircraft handling and distance to the parking position to which an aircraft has to be towed. These assumptions make it very hard to validate the models since comparisons with actual of planned demand cannot be made.
Appendix C shows that it it possible to include a towing truck capacity. This makes it possible to analyze what would happen if there current capacity changes. It becomes clear that adding this feature added extra complexity to the model. Including other resources and the medium and small body fleet will make the model even more complex and therefore harder to solve.
This approach can probably not be used by KLM since the model's assumptions make it impossible to validate and a more complex model will increase the runtime even more. Nevertheless, an algorithm has been described that can be used to analyze large time-inhomogeneous CTMC instances. This model can by used for other research.

15.3 Recommendations

The model that is developed in *Part III* of this thesis allows KLM to evaluate the expected number of towing trucks that have to be used to tow wide body aircraft on the day of operation. These models are hard to validate and therefore it is not sure whether they perform as expected. It is therefore not recommended for KLM's Network department to use these models to support any decision making.

15.3.1 Future Studies

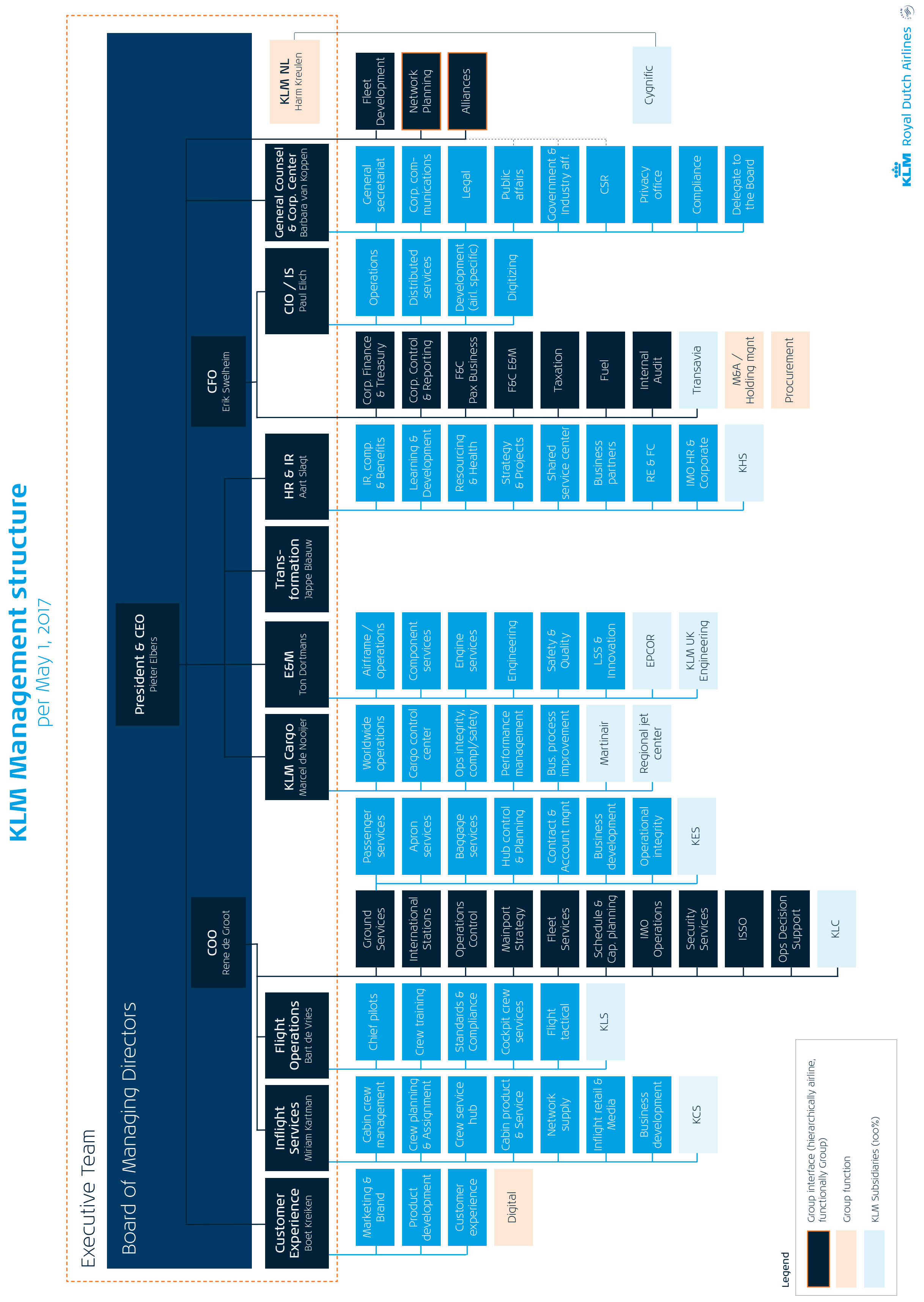
The solution approach of time discretization using iterative state exploration and dynamic state space truncation may open some new doors for the evaluation of large state time-inhomogeneous

continuous-time Markov chains. Future studies therefore include the application of this model on other CTMC instances.
Furthermore, section 13.6 sketches a way to proof that the error that is made by dynamic truncation can be analytically bounded by the truncation and discretization parameters. This proof will be technical and theoretically challenging. It would be interesting to further study the concepts of this proof.

Appendix A

KLM Organizational Chart

The next page shows the KLM organizational chart that visualizes the structure and dependencies of departments within KLM.


KLM Management structure
per May 1, 2017
Executive Team
Board of Managing Directors
President & CEO Pieter Elbers
COO Rene de Groot
CFO Erik Swelheim
KLM NL Harm Kreulen
General Counsel & Corp. Center Barbara van Koppen
CIO / IS Paul Elich
HR & IR Aart Slagt
Trans-formation Jappe Blaauw
E&M Ton Dortmans
KLM Cargo Marcel de Nooijer
Flight Operations Bart de Vries
Inflight Services Miriam Kartman
Customer Experience Boet Kreiken
Fleet Development
Network Planning
Alliances
Cygnific
General secretariat
Corp. communications
Legal
Public affairs
Government & Industry aff.
CSR
Privacy office
Compliance
Delegate to the Board
Operations
Distributed services
Development (airl. specific)
Digitizing
Corp. Finance & Treasury
Corp. Control & Reporting
F&C Pax Business
F&C E&M
Taxation
Fuel
Internal Audit
Transavia
M&A / Holding mgnt
Procurement
IR, comp. & Benefits
Learning & Development
Resourcing & Health
Strategy & Projects
Shared service center
Business partners
RE & FC
IMO HR & Corporate
KHS
Airframe / operations
Component services
Engine services
Engineering
Safety & Quality
LSS & Innovation
EPCOR
KLM UK Engineering
Worldwide operations
Cargo control center
Ops integrity, compl/safety
Performance management
Bus. process improvement
Martinair
Regional jet center
Passenger services
Apron services
Baggage services
Hub control & Planning
Contract & Account mgnt
Business development
Operational integrity
KES
Ground Services
International Stations
Operations Control
Mainport Strategy
Fleet Services
Schedule & Cap. planning
IMO Operations
Security Services
ISSO
Ops Decision Support
KLC
Chief pilots
Crew training
Standards & Compliance
Cockpit crew services
Flight tactical
KLS
Cabin crew management
Crew planning & Assignment
Crew service hub
Cabin product & Service
Network supply
Inflight retail & Media
Business development
KCS
Marketing & Brand
Product development
Customer experience
Digital
Legend
Group interface (hierarchically airline, functionally Group)
Group function
KLM Subsidiaries (100%)
KLM Royal Dutch Airlines

Appendix B

Schedule Characteristics

This appendix contains an overview of all schedule characteristics used to quantify a flight schedule and the GS-TP planning rules (PUGs).
The metrics can be applied on multiple (combinations of) subgroups which are combined by underscores. For example `AC_AMS_KL_456` denotes all KLM aircraft of category 4-6 on Schiphol in a certain time bracket.

Metrics	
`AC_AMS`	Maximum number of aircraft on AMS in time bracket
`ARR`	Number of arrivals in time bracket
`DEP`	Number of departures in time bracket
`AC_GATE`	Number of aircraft that demand a gate.
`AC_GATE_SEP`	Number of aircraft that demand a gate including a separation time of 20 minutes. 10 minutes before and 10 minutes after each gate occupation.
`RAMP`	Number of rampsnake tasks per time interval.
`POWER`	Number of powerstow tasks per time interval.
`PB`	Number of push backs in time interval.
`TOW_MAIN`	Number of maintenance tow tasks in time interval.
`TOW_NORM`	Number of normal (length) tow tasks in time interval. Note that these tasks do not necessarily have to be carried out.
`TOW_EAST`	Number of tow tasks to Schiphol East or far parking locations in time interval. Note that these tasks do not necessarily have to be carried out.
`FUEL`	Number of fuel tasks in time bracket.
`FUEL_Hydrant`	Number of freighter fuel tasks in time bracket in case that aircraft is located at a Hydrant position.
`FUEL_NonHydrant`	Number of freighter fuel tasks in time bracket in case that aircraft is not located at a Hydrant position.
`DISP_STATIC`	Number of dispensers that are reserved for a static task.

`KTW_STATIC`	Number of KTW's that are reserved for a static task.

Subgroups

`KL`	KLM aircraft
`AF`	Air France aircraft
`DL`	Delta aircraft
`F`	Freighters (cargo aircraft)
`LC`	Low Cost carriers
`1`	Category 1 aircraft
`2`	Category 2 aircraft
$\vdots$	$\vdots$
`9`	Category 9 aircraft
`AM500`	Task planned for AM500 towing truck. Subgroup only applies on `PB` and `TOW` schedule characteristics.
`AM210`	Task planned for AM210 towing truck. Subgroup only applies on `PB` and `TOW` schedule characteristics.
`AM110`	Task planned for AM110 towing truck. Subgroup only applies on `PB` schedule characteristic.

Time Shifts

`_Px`	Suffix to denote the sum of a schedule characteristic of the previous x minutes
`_Nx`	Suffix to denote the sum of a schedule characteristic of the next x minutes
`_UNCSHIFT`	Suffix of an uncertain schedule characteristic that sums up the value of this characteristic on complementary time brackets of the uncertain parts of task probability distributions that cover the current time bracket.

Appendix C

Schiphol CTMC With Tow Truck Capacity Limit

Chapter 12 describes Schiphol as a continuous-time Markov chain (CTMC) where the capacity of towing trucks is unconstrained. This appendix shows the changes that have to be made to the model in order to include towing truck capacity C_{tow}.
The state description remains unchanged:

n_1	Number of aircraft that is waiting to go into arrival handling phase.
n_2	Number of aircraft in arrival handling.
n_3	Number of aircraft waiting for a departure handling.
n_4	Number of aircraft that is being towed from a gate to a parking position.
n_5	Number of aircraft at a parking position.
n_6	Number of aircraft towed from a parking position to a gate.
n_7	Number of aircraft in departure handling.
n_8	Number of departure requests in queue waiting for an available aircraft to go into departure handling.

Including towing truck capacity in the model requires additional transitions. Some transitions has to be changed in order to account for this new capacity limit. The additional and changed transitions are underlined.

Arrival

1. *No capacity problem*: Arriving aircraft goes into its arrival handling phase.
2. *Capacity problem, aircraft available for tow*: Arriving aircraft goes into its arrival handling phase, a waiting aircraft is being towed to parking location.
3. *Capacity problem, no aircraft available for tow <u>or no towing truck available</u>*: Aircraft goes into the queue, waiting for an available gate.

Finished Arrival Handling

4. *No departure requests in queue, no aircraft waiting for arrival handling <u>or no towing truck available</u>*: Aircraft will go into waiting phase.

5. *No departure requests in queue, some aircraft are waiting for arrival handling, tow truck available*: Aircraft that finishes arrival handling will be towed away immediately in order to create space for an aircraft that is currently waiting for arrival handling.
6. *Departure requests in queue*: Aircraft that finishes arrival handling goes straight into its departure handling phase and queue decreases by one.

(Shifted) Departure Request

7. *Aircraft available*: Aircraft goes from waiting phase into departure handling.
8. *No aircraft available*: Departure request is added to request queue.

Finished Tow to Park

9. *Not all towing trucks in use or ({no gate capacity problem or (no aircraft waiting for arrival handling or no aircraft waiting for departure handling)} and {number of aircraft at gate above the tow back limit})*: Aircraft enters parking position.
10. *All towing trucks in use, gate capacity problem, aircraft waiting for arrival handling and aircraft waiting for departure handling*: Towed aircraft enters parking position and towing truck starts towing another aircraft from gate to parking. Waiting aircraft goes into arrival handling.
11. *All towing trucks in use, number of aircraft at gate below the tow back capacity limit.* While an aircraft is being towed to the parking location, the number of aircraft at a gate dropped and aircraft that has been towed to parking can be towed back to a gate immediately.

Finished Tow to Gate

12. *Capacity problem*: At this point, the tow back to the gate shouldn't have started in the first place. However, an aircraft is towed to the gate while there is no gate available. The only way to solve this problem is to tow the aircraft back to a parking location.
13. *No capacity problem, not all tow trucks used or {all tow trucks used and (no aircraft at park or number of aircraft at gate not under tow back limit)}, no demand request queue*: Aircraft enters waiting phase.
14. *No capacity problem, not all tow trucks used or {all tow trucks used and (no aircraft at park or number of aircraft at gate not under tow back limit)}, with demand request queue*: Aircraft will enter departure phase and queue decreases by one.
15. *No gate capacity problem, all towing trucks in use, some aircraft at parking, number of aircraft at gate under tow back limit, demand queue*: Aircraft that arrives at gate goes into departure handling phase. Start immediately with towing back a new aircraft from park to gate.
16. *No gate capacity problem, all towing trucks in use, some aircraft at parking, number of aircraft at gate under tow back limit, no demand queue*: Aircraft that arrives at gate goes into waiting phase. Start immediately with towing back a new aircraft from park to gate.

Finished Departure Handling

17. *Aircraft waiting for arrival handling*: Aircraft that finishes departure handling leaves, waiting aircraft goes into arrival handling.

18. *No aircraft waiting for arrival handling and no aircraft at parking or there are no towing trucks available or the number of aircraft at gate remains above towing back capacity limit*: Aircraft that finishes departure handling leaves.
19. *No aircraft waiting for arrival handling, some aircraft at parking, towing truck available and total number of aircraft at gate drops below the tow back capacity limit*: Aircraft that finishes departure handling leaves and start towing aircraft back to gate.

These events can be modeled by the transition classes in table C.2. Because of its complexity a conventional notation is not given.

i	Guard G_i	Change $\boldsymbol{\nu}_i$	Rate $\alpha_i(\boldsymbol{n}, t)$
1	$n_2 + n_3 + n_7 < C_g$	e_2	$\lambda_a(t)$
2	$n_2 + n_3 + n_7 = C_g,\ n_3 \geq 1$	$e_2 - e_3 + e_4$	$\lambda_a(t)$
3	$n_2 + n_7 = C_g,\ n_3 = 0 \vee n_4 + n_6 = C_{tow}$	e_1	$\lambda_a(t)$
4	$n_2 \geq 1,\ n_1 = 0 \vee n_4 + n_6 < C_{tow},\ n_8 = 0$	$e_3 - e_2$	$n_2\mu_a$
5	$n_2 \geq 1,\ n_1 \geq 1, n_4 + n_6 = C_{tow},\ n_8 = 0$	$e_4 - e_1$	$n_2\mu_a$
6	$n_2 \geq 1,\ n_8 \geq 1$	$e_7 - e_2$	$n_2\mu_a$
7	$n_3 \geq 1$	$e_7 - e_3$	$\lambda_d(t + x)$
8	$n_3 = 0$	e_8	$\lambda_d(t + x)$
9	$n_4 \geq 1, n_4 + n_6 < C_{tow} \vee \{(n_1 = 0 \vee n_2 + n_3 + n_7 < C_g)$ $\wedge(n_5 = 0 \vee n_2 + n_3 + n_6 + n_7 > C_t)\}$	$e_5 - e_4$	$n_4\mu_{tow}$
10	$n_4 \geq 1, n_4 + n_6 = C_{tow}, n_2 + n_3 + n_7 = C_g, n_1 \geq 1$	$e_5 - e_3 + e_2 - e_1$	$n_4\mu_{tow}$
11	$n_4 \geq 1, n_4 + n_6 = C_{tow}, n_2 + n_3 + n_6 + n_7 \leq C_t$	$n_6 - n_4$	$n_4\mu_{tow}$
12	$n_6 \geq 1,\ n_2 + n_3 + n_7 = C_g$	$e_4 - e_6$	$n_6\mu_{tow}$
13	$n_6 \geq 1,\ n_2 + n_3 + n_7 < C_g,\ (n_4 + n_6 < C_{tow} \vee$ $(n_4 + n_6 = C_{tow} \wedge (n_5 = 0 \wedge n_2 + n_3 + n_6 + n_7 > C_t))),$ $n_8 = 0$	$e_3 - e_6$	$n_6\mu_{tow}$
14	$n_6 \geq 1,\ n_2 + n_3 + n_7 < C_g,\ (n_4 + n_6 < C_{tow} \vee$ $(n_4 + n_6 = C_{tow} \wedge (n_5 = 0 \wedge n_2 + n_3 + n_6 + n_7 > C_t))),$ $n_8 >= 1$	$e_7 - e_6 - e_8$	$n_6\mu_{tow}$
15	$n_6 \geq 1,\ n_2 + n_3 + n_7 < C_g,\ n_4 + n_6 = C_{tow},\ n_5 >= 1$ $n_2 + n_3 + n_6 + n_7 <= C_t,\ n_8 >= 1$	$e_7 - e_8 - e_5$	$n_6\mu_{tow}$
16	$n_6 \geq 1,\ n_2 + n_3 + n_7 < C_g,\ n_4 + n_6 = C_{tow},\ n_5 >= 1$ $n_2 + n_3 + n_6 + n_7 <= C_t\ n_8 = 0$	$e_3 - e_5$	$n_6\mu_{tow}$
17	$n_7 \geq 1,\ n_1 \geq 1$	$e_2 - e_1 - e_7$	$n_7\mu_d$
18	$n_7 \geq 1,\ n_1 = 0, (n_5 = 0 \vee n_4 + n_6 = C_{tow} \vee$ $n_2 + n_3 + n_6 + n_7 > C_t)$	$-e_7$	$n_7\mu_d$
19	$n_7 \geq 1,\ n_1 = 0,\ n_5 >= 0,\ n_4 + n_6 < C_{tow},$ $n_2 + n_3 + n_6 + n_7 <= C_t$	$e_6 - e_5 - e_7$	$n_7\mu_d$

Table C.2: Transition classes for airport Arrival-Departure model including ground handling and towing.

Appendix D

Algorithms

Algorithm 1: GenerateTranstitionMatrix

```
Input  : Markov State Set $S$
         Current Time $t$
         Transition Classes $\eta = (\eta_1, \eta_2, \dots, \eta_k)$ with $\eta_i = (G_i, \nu_i, \alpha_i(s,t))$
             Guard $G_i$
             Change Vector $\nu_i$
             State and Time dependent Rate Function $\alpha(s,t)$
Output: Transition Rate Matrix $Q$
        Uniformization Rate $B$

/* Initialize transition matrix                                          */
1  $n \leftarrow |S|$
2  $Q \leftarrow \mathbf{0}_{n \times n}$
3  for $s \in S$ do
4      for $i = 1, \dots, k$ do
5          if $s \in G_i$ then
6              $s' \leftarrow s + \nu_i$
7              if $s' \in S$ then
                   /* Only add transitions to states that are included in the state
                      space.  It might happen that some states are not found yet.   */
8                  $Q_{s,s'} \leftarrow \alpha_i(s,t)$
9              end
10         end
11     end
12 end
13 row_sum$_i \leftarrow \sum_{j \neq i} Q_{i,j}$          $\forall i = 1, \dots, n$
14 $Q_{i,i} \leftarrow$ row_sum$_i$                     $\forall i = 1, \dots, n$
15 $B \leftarrow \max_i$ row_sum$_i$
16 return $Q$, $B$
```

Algorithm 2: Discretization

```
Input  : Continuous Time Markov Chain instance $\mathcal{C} = (S, \eta, R, p)$
             Markov State Set $S$
             Transition Classes $\eta = (\eta_1, \eta_2, \dots, \eta_k), \eta_i = (G_i, \nu_i, \alpha_i(s))$
             Reward Vectors $\rho = (\rho_1, \rho_2, \dots, \rho_l)$
             Current (Initial) State Distribution Vector $p$
         Discretization Factor $\beta \geq 1$
         Time Horizon $T$
Output: Reward Results $r_i = (r_{i,t_1}, r_{i,t_2}, \dots r_{i,T}) \qquad \forall i = 1, \dots, l$
$t \leftarrow 0$
$Q, B \leftarrow$ GenerateTransitionMatrix$(S, t, \eta)$
$h \leftarrow \frac{1}{\beta B}$
$P \leftarrow I + hQ$
for $i = 1, \dots, l$ do
    $r_{i,t} \leftarrow \langle \rho_i, p \rangle$
end
while $t < T$ do
    $t \leftarrow t + h$
    $p \leftarrow pP$
    for $i = 1, \dots, l$ do
        $r_{i,t} \leftarrow \langle \rho_i, p \rangle$
    end
end
return $r_i, \quad i = 1, 2, \dots l$
```

Algorithm 3: Time-Inhomogeneous Discretization

```
Input  : Continuous Time Markov Chain instance $\mathcal{C} = (S, \eta, \rho, p)$
             Markov State Set $S$
             Transition Classes $\eta = (\eta_1, \eta_2, \ldots, \eta_k)$, with $\eta_i = (G_i, \nu_i, \alpha_i(s,t))$
             Reward Vectors $\rho = (\rho_1, \rho_2, \ldots, \rho_l)$
             Current (Initial) State Distribution Vector $p$
         Discretization Factor $\beta \geq 1$
         Time Horizon $T$
Output: Reward Results $r_i = (r_{i,t_1}, r_{i,t_2}, \ldots r_{i,T})$    $\forall i = 1, \ldots, l$
1  $t \leftarrow 0$
2  $t_{update} \leftarrow$ GetNextRateUpdateTime$(S, \eta, t)$
3  $Q, B \leftarrow$ GenerateTransitionMatrix$(S, t, \eta)$
4  $h \leftarrow \frac{1}{\beta B}$
5  $P \leftarrow I + hQ$
6  for $i = 1, \ldots, l$ do
7      $r_{i,t} \leftarrow \langle \rho_i, p \rangle$
8  end
9  while $t < T$ do
10     $t \leftarrow t + h$
11     $p \leftarrow pP$
12     for $i = 1, \ldots, l$ do
13         $r_{i,t} \leftarrow \langle \rho_i, p \rangle$
14     end
15     if $t = t_{update}$ then
16         $t_{update} \leftarrow$ GetNextRateUpdateTime$(S, \eta, t)$
17         $Q, B \leftarrow$ GenerateTransitionMatrix$(S, t, \eta)$
18         $h \leftarrow \frac{1}{\beta B}$
19         $P \leftarrow I + hQ$
20     end
21 end
22 return $r_i$,   $i = 1, 2, \ldots l$
```

Algorithm 4: GetNextUpdateTime

```
Input      : State Space $S$
             Transition Classes $\eta = (\eta_1, \eta_2, \ldots, \eta_k)$, with $\eta_i = (G_i, \nu_i, \alpha_i(s,t))$
             Time $t$
Output     : Next Transition Rate Update Time $t_{update}$
Parameter: Arbitrarily small $\Delta$
/* Find the earliest time for which some transition rate changes.    */
1  $t_{update} = \min\{t' : t' < t,\ |\alpha_i(s,t') - \alpha_i(s,t'-\Delta)| > 0\}$    $\forall i = 1, \ldots, k \quad s \in S$
2  return $t_{update}$
```

Algorithm 5: Time-Inhomogeneous Discretization with Iterative State Exploration and Truncation.

```
Input  : Continuous Time Markov Chain instance $\mathcal{C} = (S, \eta, \rho, p)$
             Current (Initial) Markov State Set $S$
             Transition Classes $\eta = (\eta_1, \eta_2, \ldots, \eta_k)$, with $\eta_i = (G_i, \nu_i, \alpha_i(s,t))$
             Current (Initial) Reward Vectors $\rho = (\rho_1, \rho_2, \ldots, \rho_l)$
             Current (Initial) State Distribution Vector $p$
         Discretization Factor $\beta \geq 1$
         Time Horizon $T$
         Truncation State Space Size Threshold $\tau$
         Truncation State Probability Threshold $\epsilon$
         Number of Steps per State Exploration $\upsilon$
Output: Reward Results $r_i = (r_{i,t_1}, r_{i,t_2}, \ldots r_{i,T})$    $\forall i = 1, \ldots, l$
/* Initialize transition matrix */
1  $t \leftarrow 0$
2  $t_{update} \leftarrow$ GetNextRateUpdateTime$(S, \eta, t)$
3  $Q, B \leftarrow$ GenerateTransitionMatrix$(S, \eta, t)$
4  $h \leftarrow \frac{1}{\beta B}$
5  $P \leftarrow I + hQ$
/* Calculate Rewards */
6  for $i = 1, \ldots, l$ do
7      $r_{i,t} \leftarrow \langle \rho_i, p \rangle$
8  end
9  while $t < T$ do
10     $t \leftarrow t + h$
11     $p \leftarrow pP$
       /* Calculate Rewards */
12     for $i = 1, \ldots, l$ do
13         $r_{i,t} \leftarrow \langle \rho_i, p \rangle$
14     end
       /* Update transition matrix if necessary */
15     if $t = t_{update}$ then
16         $t_{update} \leftarrow$ GetNextRateUpdateTime$(S, \eta, t)$
17         $Q, B \leftarrow$ GenerateTransitionMatrix$(S, \eta, t)$
18         $h \leftarrow \frac{1}{\beta B}$
19         $P \leftarrow I + hQ$
20     end
       /* Truncate and/or Explore state space if border is reached */
21     if $\max_{s \in S_{border}} p_s > 0$ then
22         if $|S| > \tau$ then
23             $\mathcal{C} \leftarrow$ TruncateStateSpace$(\mathcal{C}, \epsilon)$
24         end
25         $\mathcal{C}, S_{border}, Q, B \leftarrow$ ExploreStateSpace$(\mathcal{C}, Q, t, S_{border}, \upsilon)$
26     end
27 end
28 return $r_i$,   $i = 1, 2, \ldots l$
```

Algorithm 6: TruncateStateSpace

```
Input  : Continuous Time Markov Chain instance $\mathcal{C} = (S, \eta, \rho, p)$
             Current (Initial) Markov State Set $S$
             Transition Classes $\eta = (\eta_1, \eta_2, \ldots, \eta_k)$, with $\eta_i = (G_i, \nu_i, \alpha_i(s,t))$
             Current Reward Vectors $\rho = (\rho_1, \rho_2, \ldots, \rho_l)$
             Current State Distribution Vector $p$
         Truncation State Probability Threshold $\epsilon$
Output: New Continuous Time Markov Chain instance $\mathcal{C} = (S, \eta, \rho, p)$
1  $S^* \leftarrow S$
   /* Truncate all states with a probability lower than epsilon                    */
2  for $s \in S^*$ do
3      if $p_s < \epsilon$ then
4          $p \leftarrow \big(p_1, p_2, \ldots, p_{s-1}, p_{s+1}, \ldots p_{|S|}\big)$
5          for $i = 1, \ldots, l$ do
6              $\rho_i \leftarrow \big(\rho_{i,1}, \rho_{i,2}, \ldots, \rho_{i,s-1}, \rho_{i,s+1}, \ldots \rho_{i,|S|}\big)$
7          end
8          $S \leftarrow S \setminus s$
9      end
10 end
   /* Normalize State Probability Vector                                             */
11 $p_i \leftarrow \frac{p_i}{||p||}$
12 return $\mathcal{C}$
```

Algorithm 7: ExploreStateSpace

```
Input  : Continuous Time Markov Chain instance $\mathcal{C} = (S, \eta, \rho, p)$
             Current (Initial) Markov State Set $S$
             Transition Classes $\eta = (\eta_1, \eta_2, \ldots, \eta_k)$, with $\eta_i = (G_i, \nu_i, \alpha_i(s,t))$
             Current Reward Vectors $\rho = (\rho_1, \rho_2, \ldots, \rho_l)$
             Current State Distribution Vector $p$
         Current Transition Matrix $Q$
         Current Time $t$
         Border States $S_{border} \subseteq S$
         Number of Steps per State Exploration $\upsilon$
         Reached Border Threshold $\varsigma$
Output: New Continuous Time Markov Chain instance $\mathcal{C} = (S, \eta, \rho, p)$
         New Border States $S_{border}$
         Transition Rate Matrix $Q$
         Uniformization Rate $B$
1  $S_{explore} \leftarrow \{s \in S_{border} : p_s > \varsigma\}$
2  for step $= 1, \ldots, \upsilon$ do
3      $S_{newborn} \leftarrow \emptyset$
4      $S_{reached\_border} \leftarrow \emptyset$
5      for $s \in S_{explore}$ do
6          for $i = 1, \ldots, k$ do
7              if $s \in G_i$ then
8                  $s' \leftarrow s + \nu_i$
9                  if $s' \in S$ then
                       /* Transition to existing state, add to trans mat.            */
10                     $Q_{s,s'} \leftarrow \alpha_i(s,t)$
11                     if $s' \in S_{border}$ then
                           /* Include border state in next exploration step.         */
12                         $S_{reached\_border} \leftarrow S_{reached\_border} \cup \{s'\}$
13                         $S_{border} \leftarrow S_{border} \setminus s'$
14                     end
15                 else
                       /* Transition to a new state.                                 */
16                     $S_{newborn} \leftarrow S_{newborn} \cup \{s'\}$
17                     $S \leftarrow S \cup \{s'\}$
18                     $Q \leftarrow \left[Q;\ \mathbf{0}_{|S|-1 \times 1}\right]$                  // add zero column
19                     $Q \leftarrow \left[Q^T;\ \mathbf{0}_{|S| \times 1}\right]^T$                     // add zero row
20                     $Q_{s,s'} \leftarrow \alpha_i(s,t)$
21                     $p \leftarrow [p; 0]$              // add zero probability for new state
22                 end
23             end
24         end
25     end
26     $S_{explore} \leftarrow S_{newborn} \cup S_{reached\_border}$
27 end
28 $\text{row\_sum}_s \leftarrow \sum_{s' \in S \setminus s} Q_{s,s'}$        $\forall s \in S$
29 $Q_{s,s} \leftarrow \text{row\_sum}_s$                  $\forall s \in S$
30 $B \leftarrow \max_s \text{row\_sum}_s$
31 return $\mathcal{C}, S_{border}, Q, B$
```

Appendix E

CTMC Python Code

The Python code of the `markoff` library as defined in section 13.7 including a script to use this library for the airport instance as described in chapter 12 can be received upon request by contacting the author at `jordrood@gmail.com`.
Though there are some code snippets of this library included in section 13.7, the full code is not attached because it would drastically increase the size of this report.

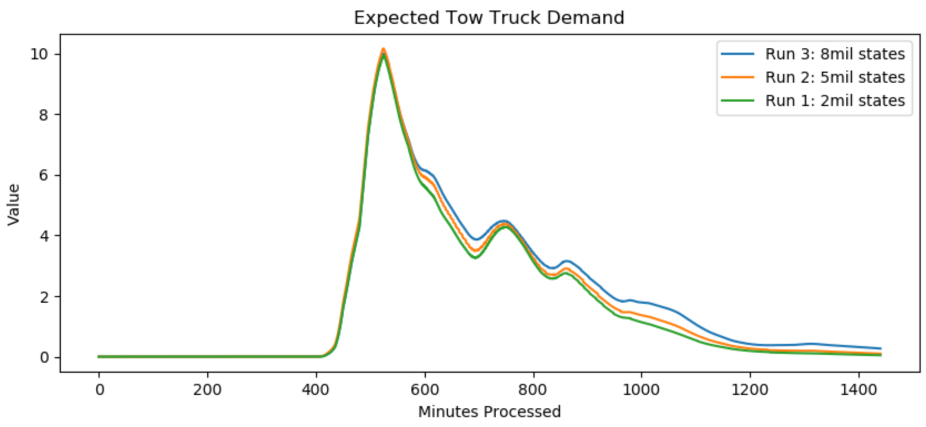
Appendix F

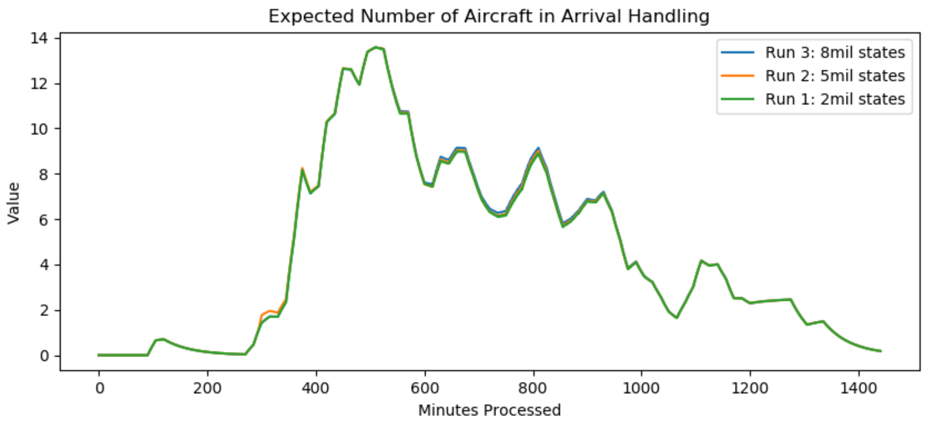
Rewards of Schiphol CTMC Runs

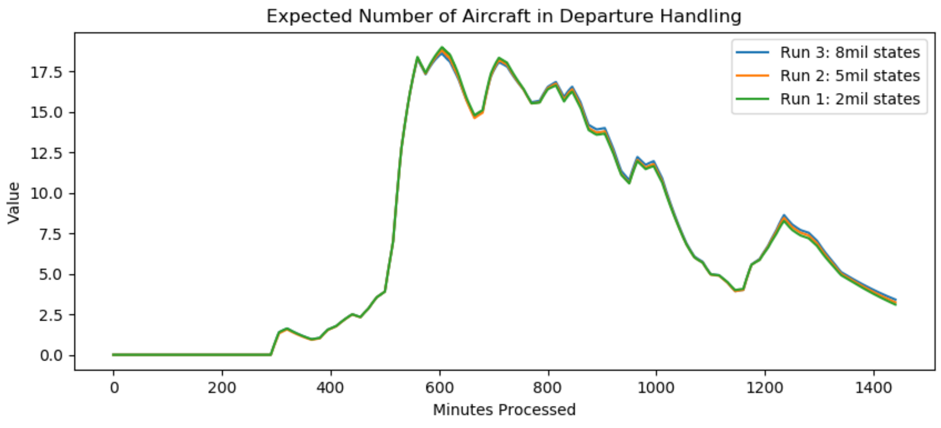
This appendix contains the results of all rewards of the Schiphol CTMC runs with discretization factor 10 and threshold settings as in table F.1. The most interested reward has already been presented in chapter 14. Furthermore, the last figure of this appendix shows the demand in the case when the probability vector is not normalized after truncation.

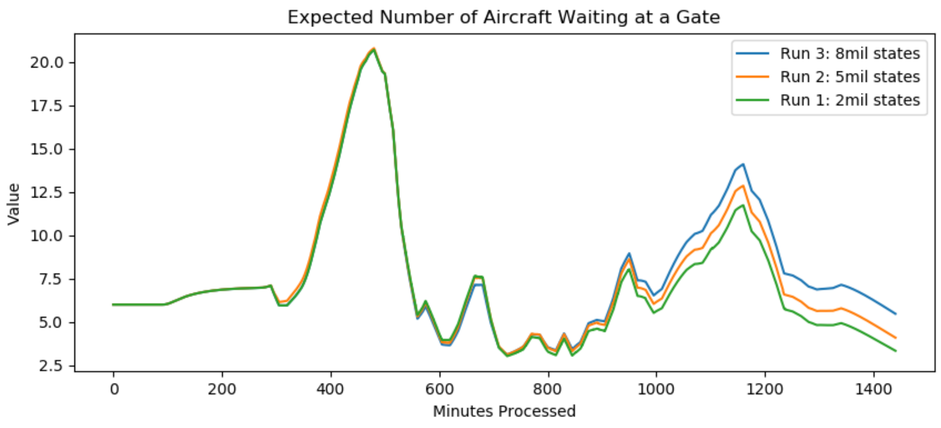
Run	State Threshold (τ)	Truncation Threshold (ϵ)	Exploration Scheme (υ)
1	2×10^6	10^{-7}	[1]*20 + [2]*20 + [3]*40 + [5]*40 + [10]
2	5×10^6	10^{-7}	[1]*10 + [2]*10 + [3]*20 + [5]*30 + [10]
3	8×10^6	10^{-8}	[1]*40 + [2]*40 + [3]*40 + [5]*40 + [10]

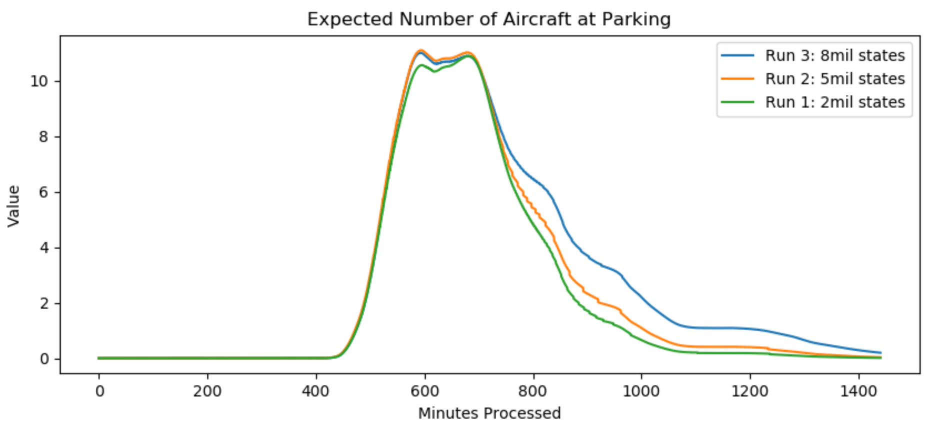
TABLE F.1: Threshold settings for 3 different runs for calculating the rewards of the Schiphol CTMC model. For all runs a border probability $\varsigma = 10^{11}$ is used.

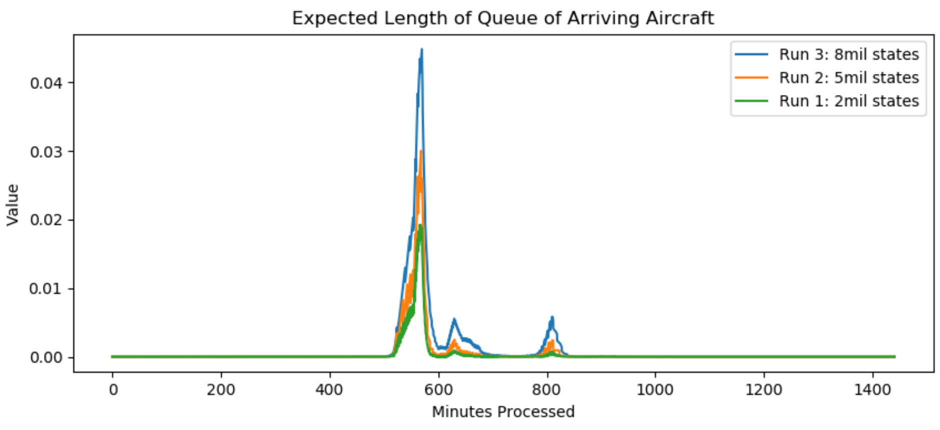


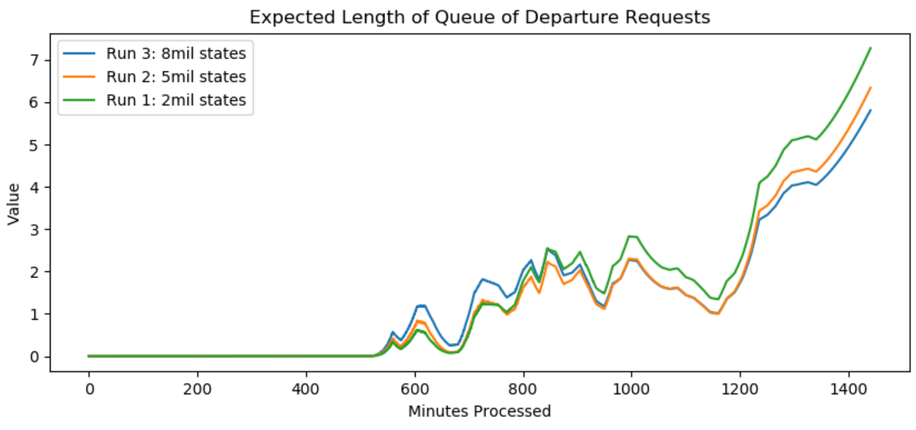

Expected Number of Aircraft in Arrival Handling
Value
Minutes Processed
Run 3: 8mil states
Run 2: 5mil states
Run 1: 2mil states
0
2
4
6
8
10
12
14
0
200
400
600
800
1000
1200
1400


Expected Number of Aircraft in Departure Handling
Value
Minutes Processed
Run 3: 8mil states
Run 2: 5mil states
Run 1: 2mil states
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0
200
400
600
800
1000
1200
1400


Expected Number of Aircraft Waiting at a Gate
Value
Minutes Processed
Run 3: 8mil states
Run 2: 5mil states
Run 1: 2mil states
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
0
200
400
600
800
1000
1200
1400


Expected Number of Aircraft at Parking
Run 3: 8mil states
Run 2: 5mil states
Run 1: 2mil states
Value
Minutes Processed
0
2
4
6
8
10
0
200
400
600
800
1000
1200
1400


Expected Length of Queue of Arriving Aircraft
Run 3: 8mil states
Run 2: 5mil states
Run 1: 2mil states
Value
Minutes Processed
0.00
0.01
0.02
0.03
0.04
0
200
400
600
800
1000
1200
1400


Expected Length of Queue of Departure Requests
Run 3: 8mil states
Run 2: 5mil states
Run 1: 2mil states
Value
Minutes Processed
0
1
2
3
4
5
6
7
0
200
400
600
800
1000
1200
1400

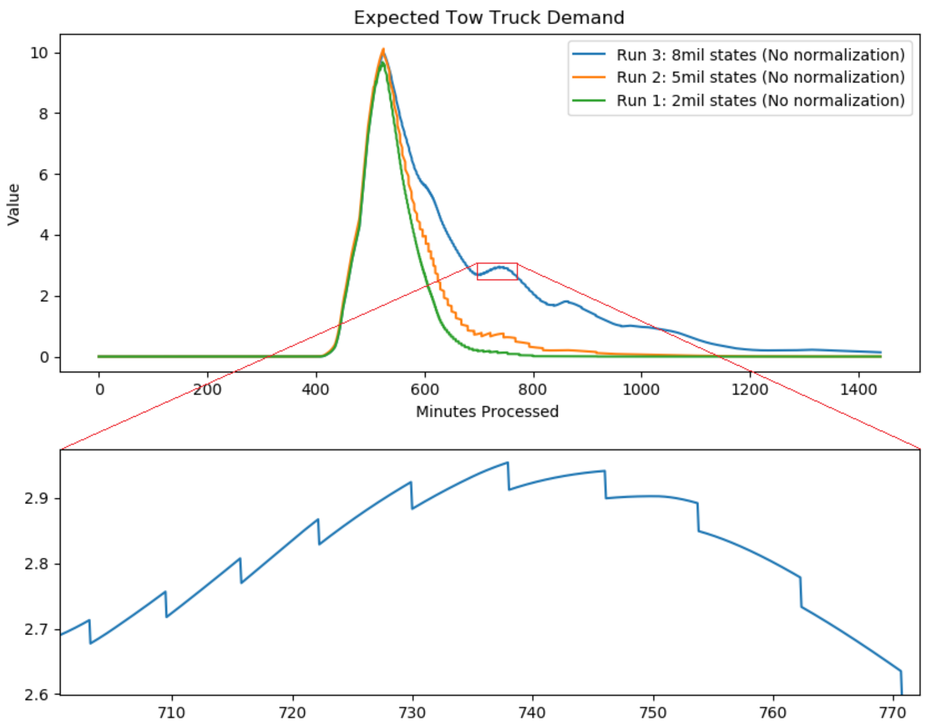


FIGURE F.1: The expected demand for tow trucks according to the runs with threshold setting as in table 14.1.

Appendix G

Effect of Exploration Scheme

This appendix contains a brief analysis of the effect of an exploration scheme on the the run-time performance and the truncation loss of the Schiphol CTMC model. The idea of such an exploration scheme is to control parameter υ that indicates how many transitions have to be explored when the state space is expanded (section 13.3). For this analysis 3 runs are carried out using state space threshold $\tau = 8 \times 10^6$, state probability threshold $\epsilon = 10^{-8}$, border probability threshold $\varsigma = 10^{11}$ and discretization factor 10. The runs all use a different fixed state space exploration parameter $\upsilon = 1$, $\upsilon = 2$ and $\upsilon = 5$. Figure G.1 shows the effect of this exploration parameter on the truncation probability loss for the first 600 minutes of a processed day. Figure G.2 shows the corresponding runtime performance. From these figures it can be observed that

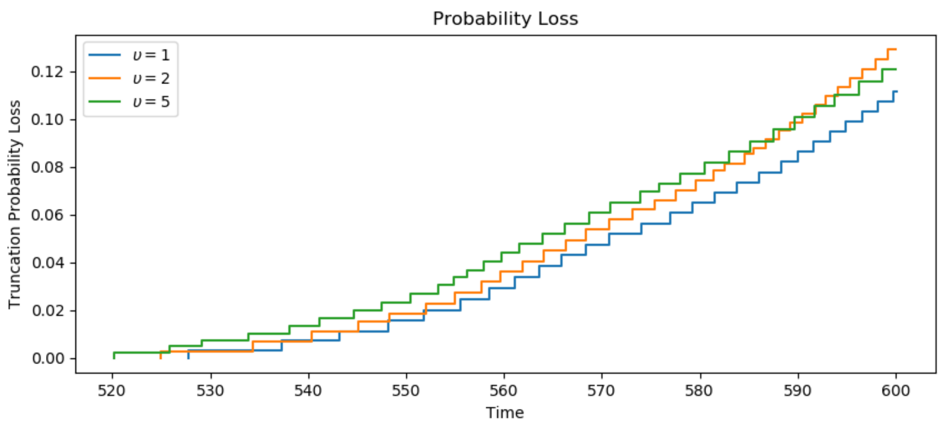


FIGURE G.1: Effect of exploration scheme on truncation loss.

a higher state space exploration parameter υ leads to a faster running program ($\upsilon = 5$ ran 16 % faster than $\upsilon = 1$, 3.7 against 4.4 hours) with the trade-off of a higher probability loss. Though this relation is not always consistent: the run using $\upsilon = 2$ shows a higher probability loss after 600 minutes than the $\upsilon = 5$ run.

In general it makes sense that a higher υ will cause the state space to expand more rapidly, which leads to finding more irrelevant states in terms of probability mass. This rapid expansion causes more state truncations, leading to a higher truncation loss. However, expanding the state space

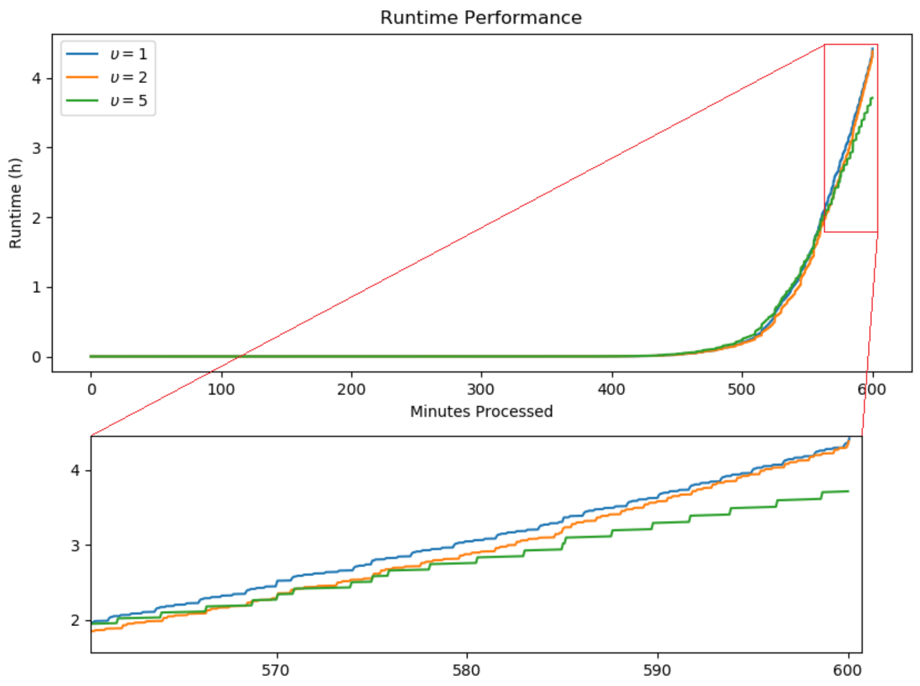


FIGURE G.2: Effect of exploration scheme on runtime performance.

for multiple transitions at a time is computationally less expensive. This follows from the fact that the transition matrix can be used for more iterations before it needs to be updated. This update consist of finding new states and stacking large sparse matrices, which can be expensive. A state exploration scheme uses the advantages of of high and low exploration rates υ. After a truncation, the algorithm takes advantage of small exploration rates since searching the state space for a small amount of transitions does not add an unnecessarily amount of irrelevant states. If, after a while, the state space truncation threshold τ has not been reached. A larger υ can speed up the search process and improve running time.

Bibliography

[1] C.M. Bischop. *Pattern Recognition and Machine Learning*. Springer US, 2006.

[2] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM.

[3] S.P.J. van Brummelen N.M. van Dijk and R.J. Boucherie. Uniformization: Basics, extensions and applications. *Performance Evaluation*, 2017.

[4] R.J. Boucherie and N.M. van Dijk. *Queueing Networks: A Fundamental Approach (CH9)*. International Series in Operations Research & Management Science. Springer US, 2010.

[5] A. Andreychenko. Uniformization of time-inhomogeneous markov population models. *Saarland University Master Thesis*, 2010.

[6] klm.com. Website KLM Royal Dutch Airways. `https://www.klm.com/corporate/en/about-klm/profile/index.html`, October 2017.

[7] NOS. Schiphol nu derde luchthaven van Europa. `https://nos.nl/artikel/2158672-schiphol-nu-derde-luchthaven-van-europa.html`, February 2017.

[8] NRC. Schiphol ziet ruimte voor groei. `https://www.nrc.nl/nieuws/2017/09/27/schiphol-ziet-ruimte-voor-groei-a1575058`, September 2017.

[9] NRC. Gemeenten protesteren tegen uitbreiding Schiphol. `https://www.nrc.nl/nieuws/2017/08/28/gemeenten-protesteren-tegen-uitbreiding-schiphol-a1571351`, August 2017.

[10] Financieel Dagblad. Groei Schiphol en uitbreiding Lelystad strijdpunten in beladen luchtvaartdebat. `https://fd.nl/economie-politiek/1217276/groei-schiphol-en-uitbreiding-lelystad-strijdpunten-in-beladen-luchtvaartdebat`, September 2017.

[11] schipholcargoworld.com. Slot scarcity at Amsterdam Airport Schiphol explained. `https://schipholcargoworld.com/supply-chain/slot-scarcity-amsterdam-airport-schiphol-explained/`, 2016.

[12] reuters.com. Dutch authorities try to avoid Russian ban over Schiphol landing slots. `http://www.reuters.com/article/us-netherlands-russia-airport/dutch-authorities-try-to-avoid-russian-air-ban-over-schiphol-landing-slots-idUSKBN1D02DB`, October 2017.

[13] Clay D. Whybark Thomas E. Vollmann, William L. Berry. *Manufacturing Planning and Control Systems.* March 1997.

[14] Leo Breiman. Prediction games and arcing algorithms. *Neural Comput.*, 11(7):1493–1517, October 1999. ISSN 0899-7667.

[15] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

[16] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the Second European Conference on Computational Learning Theory*, EuroCOLT '95, pages 23–37, London, UK, UK, 1995. Springer-Verlag.

[17] P. Prettenhofer and G. Louppe. Gradient Boosted Regression Trees. `http://orbi.ulg.be/bitstream/2268/163521/1/slides.pdf`, 2014.

[18] I. Reinstein. XGBoost, a Top Machine Learning Method on Kaggle, Explained. `https://www.kdnuggets.com/2017/10/xgboost-top-machine-learning-method-kaggle-explained.html`, 2017.

[19] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[20] Steinkraus D. Platt J. C. Simard, P. Y. Best practices for convolutional neural networks applied to visual document analysis. *ICDAR*, 3:958–962, 2003.

[21] J.R. Norris. *Markov Chains.* Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1998.

[22] S. Karlin. *A First Course in Stochastic Processes.* Academic Press, Inc., 1975.

[23] R. Nelson. *Probability, Stochastic Processes, and Queueing Theory: The Mathematics of Computer Performance Modeling.* Springer, 1995.

[24] V.S. Frost and B. Melamed. Traffic modeling for telecommunications networks. *Communications Magazine*, 32(3):70–81, 1994.

[25] TuÇğrul Dayar, Holger Hermanns, David Spieler, and Verena Wolf. Bounding the equilibrium distribution of markov population models. *Numerical Linear Algebra with Applications*, 18(6):931–946, 2011.

[26] E. Gelenbe and G. Pujolle. *Introduction to queueing networks.* Wiley, 1998.

[27] James R. Jackson. Networks of waiting lines. *Oper. Res.*, 5(4):518–521, August 1957.

[28] A. Jensen. Markoff chains as an aid in the study of markoff processes. *Scandinavian Actuarial Journal*, 1953(sup1):87–91, 1953.

[29] W.L. de Kort S.P.J. van Brummelen and N.M. van Dijk. Queue length computation of time dependent queueing networks and its application to blood collection. *ORHS*, 2017.

[30] Nazarathy Y. Kopzon, A. and G. Weiss. A push pull network with infinite supply of work. *Queueing Systems*, 62(1):75–111, 2009.

[31] N.M. van Dijk. On the finite horizon bellman equation for controlled markov jump models with unbounded characteristics: existence and approximation. *Stochastic Processes and Their Applications*, 28:141–157, 1988.

Glossary

AM110	Small towing truck.
AM210	Medium towing truck.
AM500	Large towing truck.
Bank System	System of inbound and outbound peaks on an airport.
Building Blocks	Blocks that correspond to an activity that can be simulated by OPiuM.
Critical Resources	Resources that often cause a flight schedule to be infeasible.
CHIP	System that the KLM uses to track the usage of resources on the day of operation.
Dispenser	Fuel equipment that is used to tank an aircraft from the airport fuel pipelines.
Fleet Lines	Lines on a Gannt chart that represent a single aircraft. Flights and reservations on this line are planned to be carried out by one aircraft.
Freeze	The (temporary) result of the flight schedule phase. The scheduling process is literally frozen.
Freighter	Cargo Aircraft.
Gateplanner	Tool used by GS-TP to create a gate planning.
GTW	Grote TankWagen, Dutch for big fuel truck.
Hub and Spoke	Airline network strategy to mainly provide flights to and from a single or multiple hub airports.
Infeasible Schedule	A flight schedule that can not be operated with the current resources.
KTW	Kleine TankWagen, Dutch for small fuel truck.
Medium Body Aircraft	Medium size aircraft, for KLM the EUR Boeing 737 fleet is considered to be medium body.
Operational Feasibility	A schedule is operational feasible if it is expected that it will not be a problem to operate the schedule with the current resources.
Operational Schedule	A schedule that somehow included aspects of the day of operation. In this report, an ‘operational’ schedule is the result of an OPiuM simulation.
OPiuM	KLM’s simulation software that is used to simulate flight and ground process delays in a flight schedule.

OPiuM Curve	Cummulative probability curve that describes the probability of the length of a process (building block) in OPiuM simulation.
Powerstow	Small sized aircraft baggage conveyor loader.
Ramsnake	Medium sized aircraft baggage conveyor loader.
Slot	A right to take off or land an aircraft in a certain time interval.
Small Body Aircraft	Small size aircraft, for KLM the Cityhopper fleet is considered to be small body.
Turn(around)	The process between an arrival and departure of an aircraft.
Wide Body Aircraft	Big size aircraft, for KLM the ICA-fleet is considered to be wide body.

Abbreviations

AF	Air France (airline).
AMS	Amsterdam Airport Schiphol (IATA airport code).
ARR	Arrival (of an aircraft).
BRP	Base Rolling Planning. Four-weekly planning check by GS-TP.
BTS	GS sub department responsible for loading and unloading baggage and cargo.
CRP	Capacity Requirements Planning
CTMC	Continuous Time Markov Chain.
DEP	Departure (of an aircraft).
DISP	Dispenser.
DL	Delta (airline).
EUR	Europe, used to indicate short haul flights within Europa.
FIFO	First In First Out scheduling technique.
GB	Gradient Boosting
GS	Ground Services (department).
GS-TP	Ground Services - Tactical Planning (sub department).
GTW	Grote TankWagen, Dutch for big fuel truck.
IATA	International Air Transport Association.
ICA	Intercontinental, used to indicate long haul flights to other continents.
KL-AF	KLM - Air France (airline group).
KLC	KLM Cityhopper - Regional Subsidiary of KLM.
KLM	Koninklijke Luchtvaart Maatschappij - KLM Royal Dutch Airways.
KPI	Key Performance Indicator
KTW	Kleine TankWagen, Dutch for small fuel truck.
LETO	Lr Estimation TOol - working title of thes thesis at KLM
LIFO	Last In First Out scheduling technique.
MAPE	Mean Absolute Percentage Error
ML	Machine Learning
MSE	Mean Squared Error
OPC	Operational Planning Cycle. Process to verify whether a schedule is operationally feasible.
RMSE	Rooted Mean Squared Error
SPT	Standard Processing Time.

VOP	Vliegtuig OpstelPlaats, Dutch for aircraft position; A location on an airport where an aircraft can be placed/parked. This might be at a gate.

Management Summary

This thesis study has been commissioned by KLM's Network department in order to find answers to the following questions.

1. What is the relation between a flight schedule and the resource demand for the most critical ground resources as calculated by Ground Services - Tactical Planning (GS-TP) and is it possible to quickly verify whether a flight schedule will be accepted after an operational check?
2. Given that a schedule will not be accepted by Ground Services - Tactical Planning, how can the Network capacity planners be advised to alter the schedule such that it increases the probability of acceptation?
3. Given a flight schedule, what can be said about the demand for ground resources on the day of operation?

Questions 1 and 2 are answered by implementing the plannings rules (i.e. PUGs) that GS-TP uses in order to determine the resource demand of a planned schedule. It has been shown that using these PUGs to generate tasks can give a sufficient indication of the planned demand for some resources without the realization of a gate planning. For other resources, machine learning models can be used in order to quickly estimate their demand.
The demand for ground resources on the day of operation (question 3) is estimated by applying PUGs and the machine learning models on schedules for which flight delays are simulated by OPiuM. It is shown that this approach provided a better estimator for operational demand than the current planned demand as calculated by GS-TP.
Furthermore, a mathematical model is developed to evaluate the expected demand for tow trucks that carry out widebody tows. Unfortunately, this model could not be validated because of assumptions and generalizations that had to be made in the modeling phase.

Recommendations

It is recommended to use the implementation of PUGs and the machine learning models to give a quick estimation of the resource demand as calculated by GS-TP during the network scheduling phase. Both the resource demand and the lists of tasks that can be produced using these methods can provide sufficient information to adequately implement schedule changes in order to prevent operational infeasibility.
Comparison with actual data showed that for some resources it seems like the resource demand

planning according to GS-TP differs consistently from the demand observed in operations. It is recommended to study these cases in more detail in order to improve the planning of ground resources.