Reducing peak stress on the Baggage Handling System

A research into the effect of buffering cold transfer baggage at Schiphol airport

Master thesis

Niek van der Grift

Delft University of Technology

TUDelft

Reducing peak stress on the Baggage Handling System

A research into the effect of buffering cold transfer baggage at Schiphol airport

By

F. N. S. van der Grift

Master Thesis

in partial fulfilment of the requirements for the degree of

Master of Science
in Mechanical Engineering

at the Department Maritime and Transport Technology of Faculty Mechanical, Maritime and Materials Engineering of Delft University of Technology
to be defended publicly on Tuesday February 20, 2023 at 12:45 AM

Student number:	4440676
MSc track:	Multi-Machine Engineering
Report number:	2023.MME.8760
Supervisor:	Ir. M.B. Duinkerken

Thesis committee:	Prof. dr. R.R. Negenborn,	TU Delft committee Chair, Faculty
	Ir. M.B. Duinkerken,	TU Delft committee member, Faculty
	C. Roos,	Company Supervisor, Royal Schiphol Group
Date:	February 03, 2023	

Preface

Growing up I always wanted to be a pilot. I doubted between airforce and airline pilot but because I continued growing the first option was not an option anymore. While in high school my interests shifted more to science but flying stayed in the back of my mind. My first year of studying I made an attempt at Aerospace engineering but this failed. After that I switched to mechanical engineering where I felt more at home. Together with a group of fellow students, especially Wouter, Rink and Job, the years of the bachelor flew by. After that is was the time to choose a masters degree. I chose the master Mechanical engineering with the track Multi Machine Engineering. Due to covid most of the lectures were online and all of the exams were made from home, this was a more difficult time but I managed to pull through and finish the first year with 61 points.

In order to get my degree I had to make a graduation thesis, I really wanted to do this at a company to get some working experience. The search for a good fit took a bit longer than I imagined but at the end I came into contact with Rosina Kotey. She works for the Royal Schiphol airport Group at the innovation hub. She introduced me to the problem of transfer baggage and the potential solution of cold buffering. I took this exciting opportunity to work at the airport because of my youth dream of becoming a pilot and to come back full circle. I would like to thank Chris Roos and Micha Dijkhuizen for supervising me on this Journey form the Schiphol side of the research. You guys made the job a lot of fun! I would also like to thank Mark Duinkerken and Rudy Negenborn from the TU Delft for their continues feedback on my progress.

Last but not least I would like to thank my girlfriend Mieke for her continues support during this time, especially during the more stressful times when you would bring me dinner.

Niek van der Grift
Delft, February 2023

Summary

Introduction
Billions of people fly every year and this number keeps increasing. Most of these people have checked baggage with them on their flights. In order to process all these people and bags airports need to keep up their capacity. Especially during holiday seasons there is a lot of pressure on the system. In order to cope with an increase in passengers and baggage there are two options, increase capacity or use the existing capacity more efficiently. As airports are most of the time conveniently located next to big cities increasing capacity might be a problem. A possible solution to use the existing capacity more efficiently is to buffer cold transfer baggage. In the current system all transfer baggage is fed into the system on a first come fist serve basis. Baggage with a lower transfer time does not get priority over bags with a longer transfer time. The baggage with a longer transfer time is called cold transfer baggage and this can be separated and brought to a buffer storage. This process is called cold bufffering.

Research Objective
The aim of this research is to answer the following research question:

What is the effect of cold buffering of transfer baggage and in what ways can this be done?

In order to answer the main question several subjects are researched such as the baggage journey at airports, the process of cold buffering and modelling this process. The research is a case study at Amsterdam Airport Schiphol (AAS), part of the research will be applied directly on the case.

Baggage journey
The baggage journey is the process a bag can go through at an airport. There are two ways for the journey to begin. It is brought in either by passenger at the check in area or by an arriving airplane. From the check in the baggage goes through the baggage handling system to make up. Baggage that arrives by airplane is unloaded and separated in to either transfer or reclaim baggage. Baggage can arrive arrive as loose pieces in bulk or inside a container. Reclaim baggage is transported to the infeed points of the reclaim area where passenger can pick it up. This is one ending it to the journey. The other ending is that the transfer baggage is transported to infeed points where it can enter the BHS and join the check in baggage at the make up stations. At the make up stations the baggage that belongs to one flight is collected and brought to the departing airplane. The key performance indicators of the baggage handling system are the throughput capacity, throughput time of a bag and the number of mishandled bags.

Cold buffer process
The cold buffer process is the process of separating cold transfer baggage from hot transfer baggage and to buffer to cold baggage in a storage until it can be brought to the infeed stations. There are two main categories of transfer baggage, hot and cold transfer baggage. The distinction between these two is the total transfer time, an airport can decide what the transition time is where cold baggage turns hot. In the current baggage journey there is no separation between hot and cold baggage, all the bags are collected from the airplane and brought to the transfer infeed points without any prioritisation. The goal cold buffering is to shave of the peak of the infeed of baggage. Baggage infeed is not constant during the day and especially in the morning a huge peak can arise. This peak can be lowered by feeding part of the baggage in at a later time.

A cold buffer process can be set up in many ways, for this research four main categories were made with different design options. The first category is the transition time that was just mentioned. A shorter transition time means more baggage is classified as cold but it could increase the amount of mishandled bags. The second category is the way of transportation. Baggage can be transported in several ways, three main options were looked at. Using drivers, autonomous tugs and autonomous carts. The third category is the buffer storage set up, the storage could be a simple parking space but it could also be a smaller baggage handling system on itself. The fourth category is the amount of buffer that are placed on an airport. This is depended on the size of the airport, the set up of the storage and the available space at an airport. If a set up with simple parking is chosen as the buffer storage and

there is enough space it is easier to build several storages. If the option is to build a buffer storage that is a small baggage handling system it would need a lot of space and won't be build multiple times at one airport. From all the categories a design option can be picked which are combined in a configuration.

Model and experiments
In order to test the different design options a model was build. A simulation model was chosen because this modelling technique allows to create a digital copy of a real world process and predicts performances. The process of hot and cold baggage was modelled from arrival of the airplane until the moment the baggage is fed in to the BHS. Steps in between like processing times, unloading times, driving times and infeed times were researched based on existing data at Schiphol and personal excursions to the airside with a stopwatch. The elements like airplanes, tugs, containers and infeed points were modelled as elements in the simulation. Input for the simulation was real world arrival data from a week of data in the summer of 2019. First a model was build without a buffer to be able to validate the outcome with the real world data. The output of the model were all the infeed times of bags. After the model was validated and verified a buffer was implemented, which showed the peak shaving that was desired.

From the research in to the design options two configurations were chosen, the first configuration is a buffer process that has a transition time of three hours, normal transport and one buffer location where carts and containers are parked. The second configuration has a transition time of three hours, autonomous tug transportation and three buffer locations where carts and containers can be parked. Next to the normal input three other possible future scenarios were devised that could be used as input. One scenario saw an increase in arriving flights during the morning peak hours, the second scenario saw an increase in bags to individual infeed locations because another infeed location would become unavailable. The last scenario saw the same amount of arriving flights but an increase in containers over bulk baggage. All scenarios were combined with the configurations as well as configuration without a buffer. This resulted in 12 different experiments.

Results
All the experiments were simulated. The outcomes of the new infeed times were grouped together in time slots of 15 minutes. Based on the configuration without a buffer a morning peak was found for each scenario. After that the new peak was found with the two configurations for each scenario and based on the old and the peak a peak reduction was calculated. The average peak reduction over all the experiments was 28.4%. The peak reduction on average for the experiments with configuration one were 29.7% and with configuration two 27.2%. In configuration one it could be seen that the South infeed hall had to work at maximum capacity for a couple of hours to process all the cold baggage that had to be fed in there after the buffer period. In configuration two, where three buffer locations were used, the individual infeed points were not working at full capacity if all were in use. The recommendation is to implement a cold buffer process that can still feed in baggage at all the infeed points. To transport all the cold baggage the recommendation is to use autonomous tugs, they drive a bit slower but that just means less time in the buffer storage for cold baggage. Another advantage of using autonomous tugs is less personnel, which there is a shortage of at the moment. The cold bags can be stored for a shorter time because a trough can be seen in the infeed distribution between the morning peak and the new buffer peak. This could also mean that the transition time could be lowered by the same amount to increase the number of bags that would be classified cold. But at a couple of experiments the buffer peak was already higher the highest point in the figure so this would not attribute to peak shaving.

Contents

Nomenclature

Abbreviations

Abbreviation	Definition
AAS	Amsterdam Airport Schiphol
AIBT	Actual In Block Time
BHS	Baggage Handling system
BSM	Baggage Source Message
CVT	Computer Vision Technology
DCV	Destination Coded Vehicle
GASP	Generic Assignment Scheduling Problem
GBS	Grid-Based Storage
IATA	International Air Transport Association
ICAO	InternationalCivil Aviation Organisation
KLM	Koninklijke Luchtvaart Maatschappij
KPI	Key Performance Indicator
LU	Loading Unit
MPC	Model Predictive Control
NABO	Narrow Body
PDL	Process Description Language
PN	Petri Net
RDS	Ramp Direct Service
RFID	Radio Frequency Identification
RSG	Royal Schiphol Group
SITA	Société Internationale de Télécommunications Aéronautiques
ShoCon	Short Connection
SysML	System Modelling Language
VOP	Vliegtuig Opstel Plaats
WIBO	Wide Body

1 Introduction

This chapter the research back background and the problem statement will be given. Based on that a research question and subquestions are derived. The approach to answering those questions will be told in the last part of this chapter.

1.1. Research Background

Amsterdam Schiphol Airport is a large hub airport used by 71,7 million passengers in 2019. Although the passenger numbers dropped in 2020 and 2021 due to COVID19, Amsterdam Schiphol Airport was Europe's busiest airport in 2021. Each year, Schiphol handles some 53 million pieces of baggage. This can vary from day to day from around 120,000 items on a slow day to 180,000 during extremely busy periods, such as the start of a holiday season. Bags have various journeys they can take at Schiphol airport. Baggage can come in by either airplane or it can be brought in by a passenger. If a passenger brings their bags in, it is checked in and transported with a baggage handling system (BHS) to an outfeed point. At the outfeed points there are make up areas where all the bags from one flight are collected and put in containers or carts. A ground handling company drives the carts or containers to an outgoing airplane. If a bag comes in by airplane it has 2 options. Option 1 is that is the passengers final destination is Schiphol airport. Then the bag is transported by a ground handling company to the BHS which moves it to the reclaim area. Option 2 is that the passenger is making a transfer at Schiphol airport, Then bag is transported by the ground handling company to a BHS infeed point where it is moved to an outfeed point again just like the baggage that is brought in by passengers. This process is visualised in figure 1.1 .

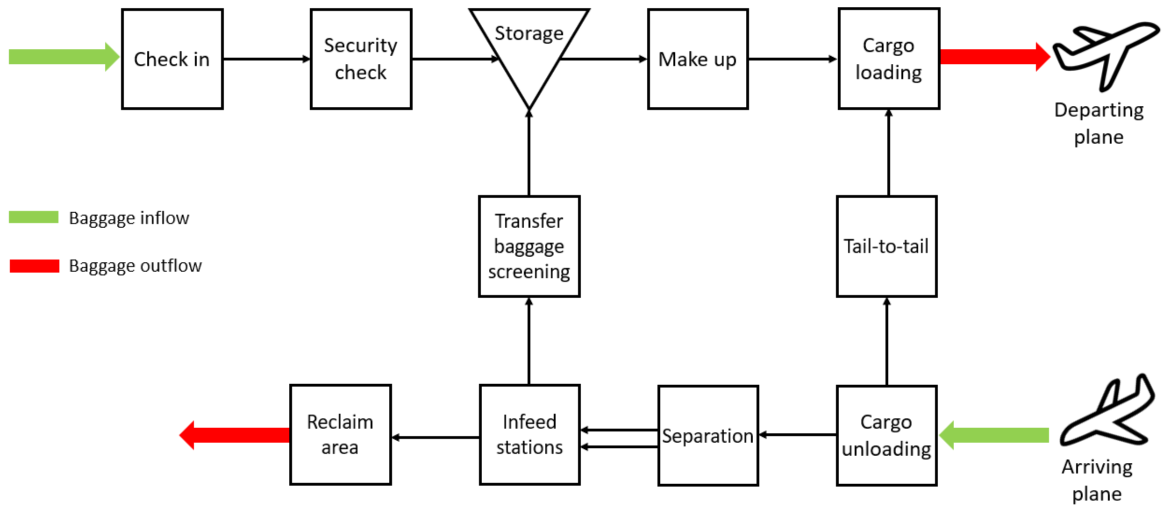


Figure 1.1: Visualization of the baggage journey

1.2. Problem statement

Due to an increase in demand of travel and an upward trend in flight movements over the last couple of years, the demand on the baggage handling system has increased especially at the infeed of the systems from airside. KLM is responsible for the biggest share of flight movements on Schiphol airport. Their business model uses Schiphol airport as an important transfer hub. To improve customers travel experiences, KLM tries to keep the transfer time a short as possible. To do this they first plan to let the intercontinental flights come in so that these passengers can transfer to the European flights that come in a bit later in a second wave. This all happens in the morning which creates a morning peak. This is shown in figure 1.2.

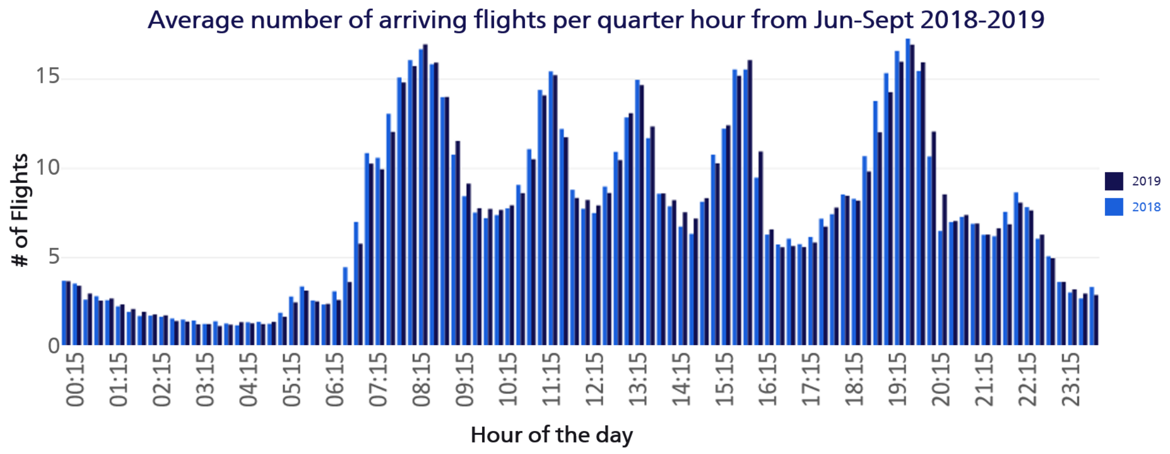


Figure 1.2: Distribution of flights during the day at Schiphol Airport (Royal-Schiphol-Group, 2019)

There are several peak moments during the day but there are a lot more transfer bags in the first morning peak. This can be seen by the distribution on infeed of transfer baggage in figure 1.3.

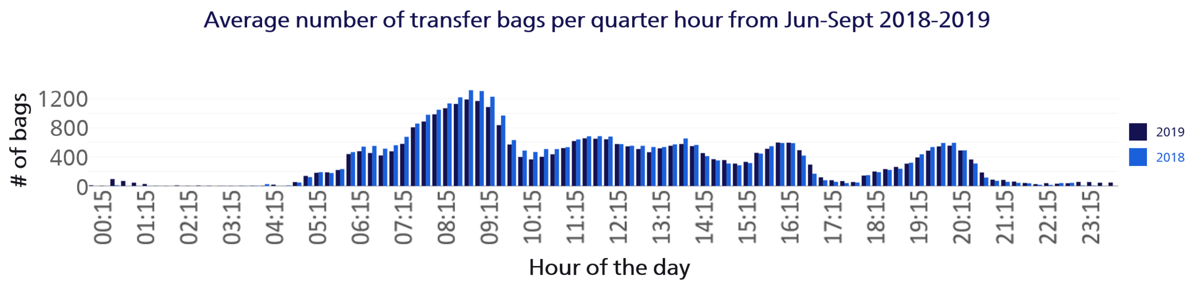


Figure 1.3: Distribution of transfer baggage during the day at Schiphol Airport (Royal-Schiphol-Group, 2019)

Due to this morning peak, problems could occur at the infeed points of the BHS which also affects the throughput of the whole system at the storage and the makeup areas. This can cause delays in flights or lost baggage which both cost a lot of money for the airlines and the airport.

1.3. Research objective

One of the solutions that is proposed by Schiphol airport to improve the performance of the system is cold buffering. Cold buffering is a form of peak shaving where certain bags are buffered before entering the system. Transfer baggage can be divided in to three categories: Short connection (ShoCon) bags, hot bags and cold bags. The division is made based on the transfer time between the connecting flights. If the transfer time is less than 45 minutes, the time it requires to go through the system, the bag is called ShoCon. If the bag has more transfer time than 45 minutes it will be categorized by a temperature. Hot means shorter transfer time and cold means longer. The division is currently set at three hours. Cold buffering, as the name implies, filters out the cold bags and buffers them outside the system to enter them at a later time with the aim to lower the peak at the infeed of baggage in the morning. The goal of this research is to prove it theoretically possible to achieve this, find out how this should be done at what practical implementations must be made to make this possible.

1.4. Research Question

The main research question would be:

What is the effect of cold buffering of transfer baggage and in what ways can this be done?

To guide this process the following sub questions have been derived:

1. *What does a baggage journey currently look like and what are the KPIs?*
2. *What is cold buffering and what are the design options to set up a cold buffering process?*
3. *What would cold buffering look like at Schiphol Airport?*
4. *What is potential of cold buffering on peak baggage load?*
5. *In what way could the cold buffering process be modelled?*
6. *What would be the effect of the different design options on the KPIs?*
7. *What would be the most desired implementation for Schiphol airport?*

1.5. Approach

The first step of this research would be to get familiar with the inner workings of the baggage journey at airports and to find out how this works. This is explained in chapter 2. After that it must be researched how cold buffering could be implemented at airports and how peak shaving works. This is written down in chapter 3. The next step is to dive deeper in to the case study at Schiphol airport and see how cold buffering could be implemented with different design options and what the potential of cold buffering might be. This will be explained in chapter 4. After the case study is done, a model must be made. Literature research must be done in order to find the correct modelling way. This research is shown in chapter 5 Then, after the modelling way is found the model can be set up with it's elements. The model must be verified and validated. All of this is done in chapter 6. Different experiments that will be tested in the model explained in chapter 7. The results of the experiments is shown in chapter 8. Based on the results a cost analysis is made in chapter 9 in order to give a better advice on the implementation for Schiphol airport. All the results will be discussed in chapter 10.

2 The Baggage Journey

In this chapter the journey of a bag is explained. An overview is seen in figure 2.1, the different points and areas will be further explained in the different sections. At the end of the chapter the key performance indicators of the system are discussed.

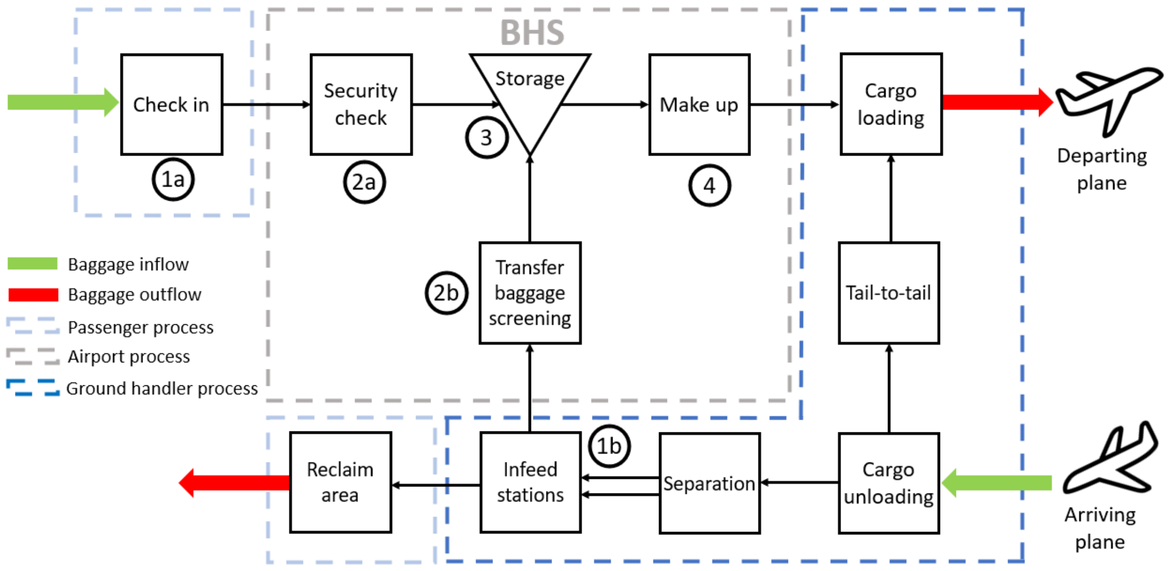


Figure 2.1: The baggage journey

2.1. Infeed of baggage

Baggage can enter or start it's journey at two points indicated by the green arrows in figure 2.1. The first entry point is checked baggage, this baggage is brought in by the passengers. The second way baggage can start it's journey is by arriving with an incoming flight.

2.1.1. Landside infeed

Baggage that the passengers want to take along on their travels that does not fit in cabin spaces needs to be checked in. This can be done at the check in desks at the airports. This is indicated with 1a in figure 2.1. This can usually be done from up to three hours before the departure of the flight and until half an hour before departure. Airports have several options for the check in desks, it can have a dynamic or static system and a manual or automated service. With the automated check-in service or self check-in the passenger must have checked in online and they place their baggage in a baggage drop off point where they need to attach a label themselves(Ma et al., 2021). This saves the need for personnel, which the airport would need in the manual system. If the manual or automated system is dynamic it means it will continuously open and close check in desks based on the desired demand and airlines can even use other airlines check in desks to handle a peak demand if that is necessary. If the system is static it means that all the desks are pre-allocated by a scheduler a certain time period in advance(Parlar and Sharafali, 2008, Bruno and Genovese, 2010, Hsu et al., 2012). For odd sized baggage most airports have a special infeed point which has a straight connection through the BHS(Bolijn, 2018).

All the baggage that is checked in needs to be able to be identified by the system of the airport where it is checked in, by the handler that puts the baggage in the airplane but also by the airport where the baggage will arrive. Currently this is done by attaching a label to the bag with a barcode on it. The barcode contains a lot of information including a identification number for the bag, the destination, the owner, the flight number and the date of the flight (Bardinas et al., 2019). One disadvantage of the barcode is that it must be completely visible for the scanner to recognize it. There are two currently identified potential alternatives, radio-frequency identification (RFID) and computer vision technology (CVT). The International Air Transport Association (IATA) is recommending airlines to use RFID technology. RFID does not require to be in the line of sight as the barcode needs to be and scanners can scan multiple bags at a time to receive live location updates of the bag in the system. The second option, CVT uses a multi camera set up to scan baggage going into the system and it uses image recognition software to keep track of the baggage. This would make the usage of labels with either barcodes or RFID chips redundant, which is better for the environment and because of the images, if there is any damage to the bag, it can be traced back to where it happened.

2.1.2. Airside infeed

The second green arrow in figure 2.1 comes from arriving airplanes. After the plane lands all the baggage needs to be unloaded. This is done by a ground handler who is either hired by the airline or by the airport itself, this is why this process is within the dark blue dotted lines. The baggage can either be bulk baggage or containerized baggage, this is shown in figure 2.2. The containers are specifically designed to fit inside the airplane (Brandt, 2017). The type of baggage has an effect on the unloading process of the airplane.

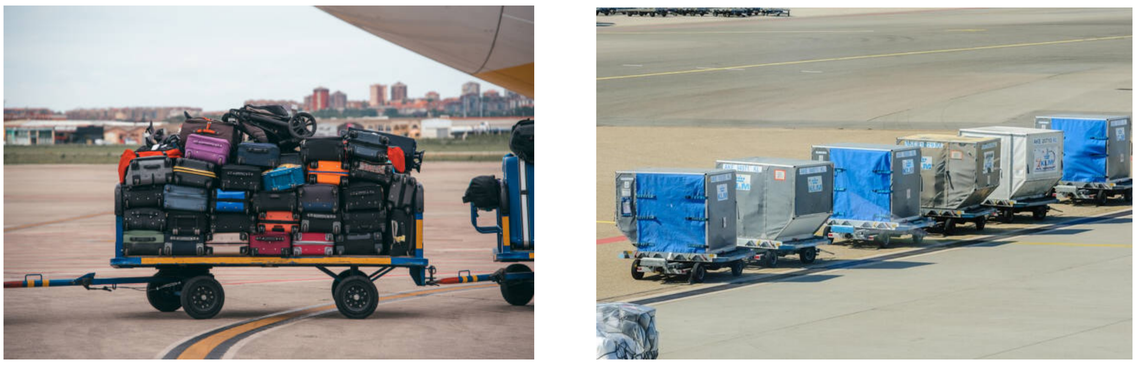

(a) Bulk baggage (Guti, 2018) **(b)** Containerized baggage (Barnes, 2018)

Figure 2.2: Different types of baggage

Within the bulk or containerized baggage another division can also be made between transfer baggage and original destination baggage. Smaller regional airports, such as Eindhoven airport, only have original destination baggage while larger international airports with a hub function have both. Transfer baggage accounts for around 40% of the baggage handled at Schiphol airport(Staat, 2022). In certain situations baggage needs to be transported from one airplane to the other because of short transfer times. The bag does not have enough time to go through the normal system. This type of baggage is called short connection (ShoCon) baggage and will be transported tail-to-tail by the ground handler. While unloading airplanes filled with bulk baggage, the original destination baggage must be separated from the transfer baggage. Currently this is done manually by checking the destination on the label of the bag. Containerized baggage is already separated at the originating airport when the containers are being filled. While unloading the containers the separation must be made between the containers with the two types of baggage. After unloading the bulk baggage or containers are placed on carts and the carts are transported by a tug to the infeed stations. The infeed stations are indicated with 1b in chapter 2. Original destination baggage and transfer baggage have different infeed locations. The infeed stations for original destination baggage are in direct connection with the reclaim belts and the infeed stations for transfer baggage are connected to the BHS where it can join the check in baggage.

2.2. Baggage handling system

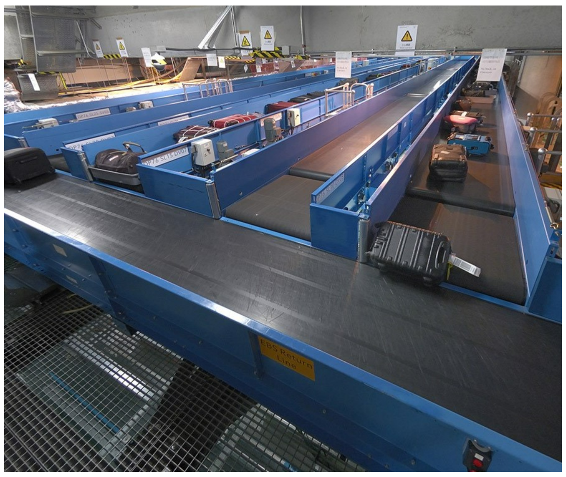
The BHS is enclosed by a gray dotted line in figure 2.1. This part of the process is owned by the airport. A BHS can range in complexity from a simple conveyor to a complicated system with different storage locations and interconnected transport methods. In general, baggage that is entered in to system will first be screened, then placed either into storage or immediately transported to the make up stations. These steps will be explained further in the following sections.

2.2.1. Screening

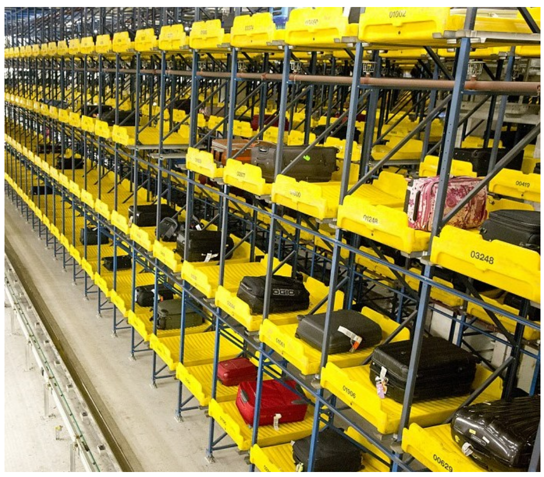
After the baggage is fed in to the system at an infeed station, it will go through a security check. The baggage must pass through a screening before it goes further in the BHS. Since the 9/11 attacks in 2001 this has become more thorough (Leone and Liu, 2005). The baggage goes through a CT scanner which X-rays the bag and generates a tomogram. The scanner then calculates the density and mass of individual objects inside the bag. If the density or mass falls in the range of a dangerous material the bag gets flagged. For checked baggage this happens at 2a in figure 2.1 and for transfer baggage at 2b. Not all transfer baggage has to checked. Depending on the land of origin, transfer baggage needs to be checked. After the screening the baggage needs to be transported further. There are several options for transportation in the BHS. The most common one is the usage of a conveyor belt. Baggage is placed on the belt and through a system of scanners, junctions and sorters all the bags are transported to the correct destination. Another option to sort baggage is to use tilt trays, in this system individual trays are loaded with one piece baggage and they can tilt at the correct place to get the bag to the correct destination. A third option is to use destination coded vehicles (DCV), a vehicle which carries one bag either on rails or autonomous to its next destination in the BHS(Markus and Frey, 2014, Silven, 2018).

2.2.2. Storage

After screening baggage has two options, it can go either to storage or directly to make up. This is based on the transfer time. If the baggage won't be immediately directed to the make up stations, it will be stored. This happens in the baggage storage indicated with the 3 in figure 2.1. As seen in this figure this storage can be used for check in baggage as well as transfer baggage. Airport baggage storage can have a centralized or a decentralized architecture. In a centralized architecture there is one central storage opposed to several storages in different locations in a decentralized architecture (Lin and Liou, 2015). Airports can have a simple storage with multiple parallel conveyor belts, as seen in figure 2.3a. Once the bags are on that belt it will be first in first out. Another option is to have a smart system, one of the examples of a smart system is to build a multilayered storage with individual trays which can be picked up at any moment. This can be seen in figure 2.3b. The advantage of having this system is that it can store more bags per square meter and it's more flexible if departure times change because it is not first in first out. All storages have a limited capacity which must be taken in to account by the airport (Hafilah et al., 2017).



(a) Storage (JEC, 2017)



(b) Smart Storage (LHR, 2017)

Figure 2.3: Different types of storage

2.2.3. Make up

The last section of the BHS, before the ground handlers bring the baggage to the airplanes, is the make up station. This is were all the baggage that has to go in to one flight is brought together and made up, this happens at the 4 in figure 2.1. From the BHS the baggage arrives either on a carousel or a lateral. A carousel can hold several flights while laterals are allocated per flight. Because there are several flights loaded onto one carousel this can increase the amount of mishandled baggage but carousels take up less space relatively to laterals. Because the laterals are only used for one flight either the container or the baggage cart will be loaded correctly but the downside of having a lateral is that if a flight is delayed the chute cannot be used until that flight departs.

2.3. Outfeed of baggage

Baggage can leave or end it's journey at two points indicated by red arrows in figure 2.1. This is either at landside through reclaim baggage or airside with departing flights.

2.3.1. Landside outfeed

As explained in the airside infeed section of this chapter, original destination baggage is fed in to the system and transported to the reclaim area. At Schiphol airport there is a direct connection between these infeed points and reclaim carousels(Boute, 2016). In that case those infeed points can only be used for reclaim baggage. It is also possible for an airport to have general infeed points where the reclaim baggage is separated from the transfer baggage in the system and that the reclaim baggage is then transported to the reclaim area. At the reclaim area the passengers can pick up their baggage and leave the airport.

2.3.2. Airside outfeed

All the baggage that is supposed to go one one flight is collected at the make-up area. From there, a ground handler will load the bags either in a container or on a cart. This process can also be automated with the help of a robot (Bryndin, 2021). From the make-up area the carts or containers are driven by a tug to apron where the baggage will be loaded into the airplane.

2.4. Conclusion

Different design options have an effect on the overall performance of the whole baggage journey The choice of transportation, the amount of screening machines and the amount of infeed stations are all important to determine the number of bags than can be handled by the system. There are several key performance indicators (KPIs) in the system. The three most important ones are

1. System capacity (bags/hour)
2. Throughput time (hours)
3. Mishandled baggage (total number of bags mishandled)

3 Peak Shaving

In the previous chapter the journey of a piece of baggage at an airport was explained, in the conclusion some key performance indicators of the baggage handling system were given. One of these KPIs was the infeed capacity. The infeed capacity is determined by the number of infeed points and their individual throughput of bags per hour. Infeed of baggage will rarely be a constant number due to the fact that airplanes arrive and depart periodically is sort of waves. The system must be able to handle the highest peak which mostly occurs in the morning. There are several solutions to lower this peak and thereby either lowering the capacity needed to handle all the baggage or to offer the opportunity to handle more airplanes while maintaining the same peak. This process is called peak shaving. The peak can either be shaved to the front or to the back. The peak can be shaved to the left by effecting the checked baggage or

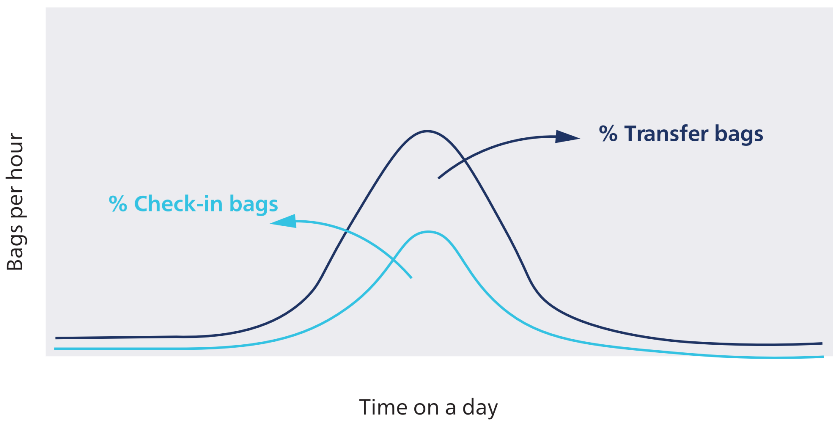


Figure 3.1: Peak shaving effect (van Wingerden, 2022)

it can shaved to the right by effecting the transfer baggage, this is visualized in figure 3.1. In this chapter the latter option will be further explored to see how it can be achieved and what the potential effect is.

3.1. Cold buffering

One of ways to pull or shave the peak to the right is to prioritize the transfer baggage in a way. The less important baggage will be fed in to the system later in order to flatten the curve. The importance of the baggage is directly linked to the transfer time. If the transfer time is long, the baggage is called cold and if the transfer time is short, the baggage is called hot. Cold baggage can be fed in to the system later than hot baggage and still make it through the system in time to catch the next flight. In order to do this the hot and the cold baggage must be separated from each other. This is done differently for bulk and containerized baggage. In order to be able to separate hot and cold baggage from containers, without the need to unload containers at the apron, the containers must be made up as either a hot or cold container at the make-up stations of the original departure airport. When unloading bulk baggage a separation is already being made between reclaim and transfer baggage on the apron. An extra separation could be made while unloading between hot and cold bags by placing them on different carts. After the baggage is separated the reclaim baggage will go to the reclaim infeed points, the hot baggage will go to the transfer infeed points, this could be the same location for certain airports. The cold baggage can be brought to a storage where it will wait before being brought to the infeed points for transfer baggage. This is where the term cold buffering derives from. This changes the baggage journey slightly and is shown in figure 3.2. Compared to figure 1.1 an extra line is added after separation which leads to another storage, the buffer location.

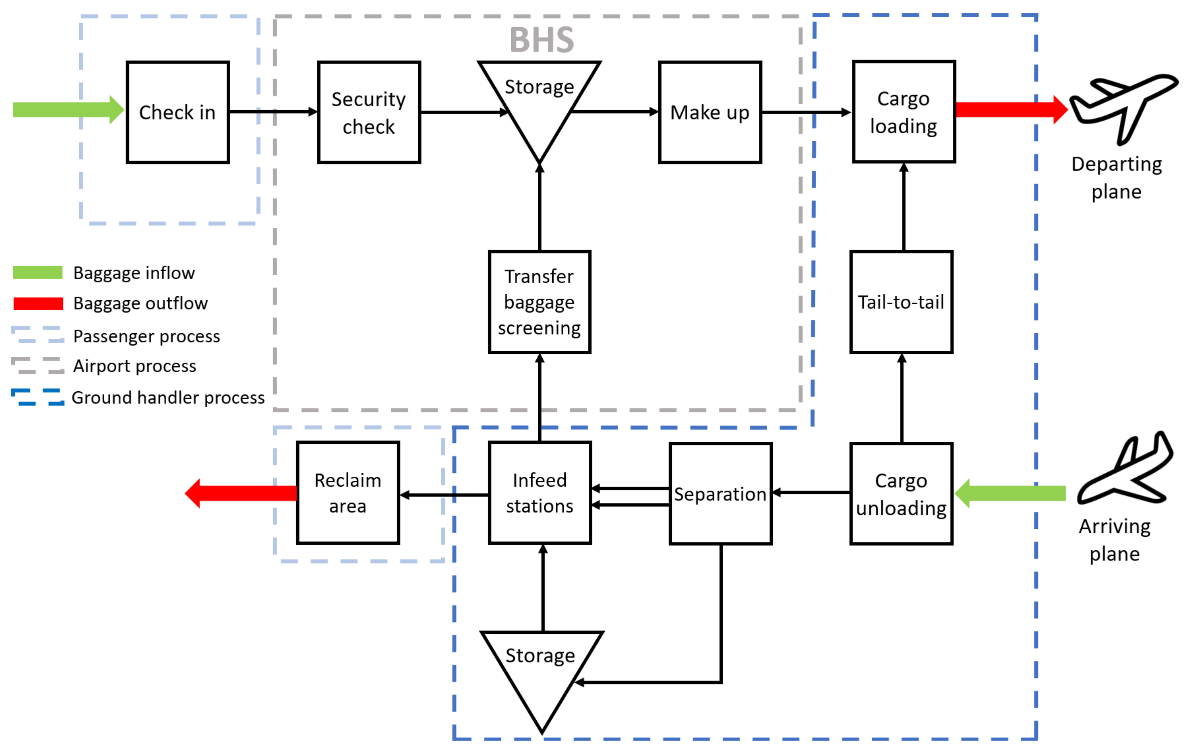


Figure 3.2: The baggage journey with cold buffering

3.2. Design options

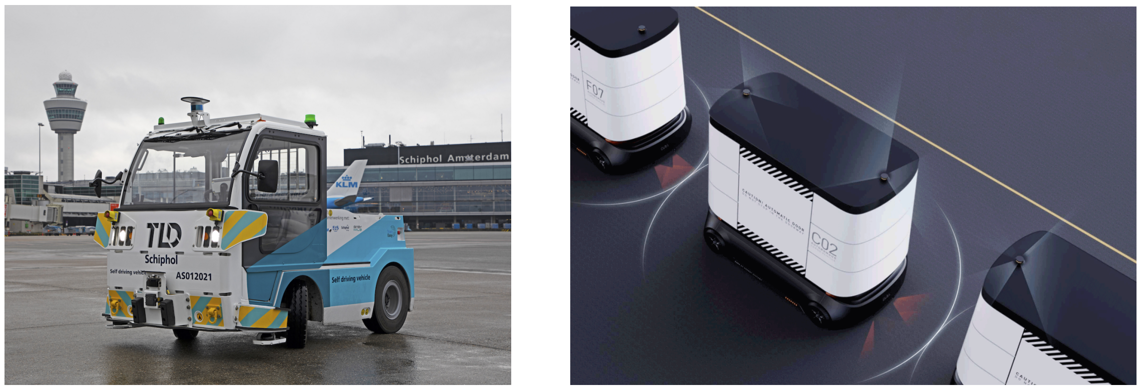
Airports have several options in the way cold transfer baggage could be buffered. An important question to ask is from what transfer time does cold baggage become hot. Another question would be how the cold baggage is moved from the apron to the buffer and there are several options for where a buffer storage could be located. The last question that is looked at in this section is the configuration of the buffer storage it self, how will be the baggage be entered, how long should the baggage be buffered and how is transported and entered in to the actual BHS. This section aims to offer different options for these questions on which airports could make a design.

3.2.1. Transition time

A distinction is made between super hot, hot and cold baggage. Super hot is the ShoCon baggage that does not have enough transfer time to go through the BHS. This time is set by the airport operators, for Schiphol airport this is 45 minutes. If the transfer time is shorter than 45 minutes it must be transport tail to tail. The choice of time from where cold baggage transitions into hot baggage is something that can be chosen by the airport operators as well. This choice depends on how long baggage should be buffered, how long it will take to bring the baggage through the system and how many pieces of baggage should be buffered. If the airport were to take a longer transfer time less baggage would be classified cold but the cold baggage can be buffered longer without the problem of not reaching their next flight. Looking at figure 1.3, the morning peaks seems to take around two hours. In order to feed the cold baggage in after this peak the baggage must be buffered for two hours as well. With a process time of 45 minutes through the system and accounting for extra travel time for the baggage the transition time should be set around 3 hours. If the baggage is buffered shorter, for instance one hour, the transition time can become two hours. It is also possible set the transition time at four hours to decrease the chance of mishandled baggage by arriving late but this might decrease the effect of peak shaving.

3.2.2. Transport options

In the current operations there is not a lot of automation yet in the transport of baggage from the apron to the infeed points. All the carts are loaded manually and driven in a train, pulled by a tug, to the infeed points. If the baggage is separated between cold and hot baggage and extra tug would be needed to drive the cold baggage to the buffer. An extra tug also needs an extra driver. As personnel is in high demand this might not be the feasible to do. Another option for transport is to use self-driving tugs (Gosling and Barton, n.d.). The airport operators can choose to implement the self-driving tugs just for the transport of cold baggage but also for the whole process in general. A third option is to make use of self-driving baggage carts(Lee and Seo, 2022). Currently there are several companies working on self-driving baggage carts. This would also make the use of driving personnel obsolete and could also be first used for the transport of cold baggage and later the for the transport of all baggage.



(a) Self-driving tug **(b)** Self-driving baggage carts (Cura, 2020)

Figure 3.3: Autonomous transport solutions

3.2.3. Buffer configuration

The buffer configuration is the set up of the buffer, how will the cold baggage be fed in, stored and fed out. This can be as simple as just a cart storage where the cold carts are parked and picked up after the peak to be brought to the infeed points. The configuration can also become more complicated by creating a system where each piece of baggage can move around individually, just like in the actual BHS. The choice depends on how space and money is available and on which transportation method is chosen. If the transport is done with regular carts, tugged by either a self-driving or normal tug, the carts just be placed in the buffer. If the transportation method that is chosen makes use of self-driving carts than these must be unloaded because it would be too costly to park these carts and let them wait for two hours. So in this case a system must be set up where the baggage is unloaded from the self-driving carts and either loaded in to normal carts or into a storage system. Another part of the configuration is how long the cold baggage should be buffered. With the simpler system of parking the carts without unloading it is only possible to buffer the whole batch for a certain amount of time. If the baggage is unloaded the buffer time can be based on individual departure times of the baggage and stored that way. This does require the more complicated system that the self-driving carts also need. This would look like the storage seen in figure 2.3b. It is also possible to work with bins. Certain areas or carts are allocated a time slot based on the departure time of the baggage. When the baggage is transported to the buffer the baggage is put in the right bin and will be buffered in a group but based on their departure time.

3.2.4. Buffer placement

The last design option an airport operator needs to think about is the placement of the buffer or buffers. Buffers can be placed close to infeed areas to reduce transport time after buffering but it might also be efficient to use a more central location to reduce transport time to the buffer itself. It is also depended on the space an airport has available. At Schiphol airport there are 3 main infeed areas for transfer baggage. Next to the infeed areas there is space to park carts and it will be interesting to see the effects of creating a buffer space at one, tow or even all three at the infeed areas.

3.3. Conclusion

Cold buffering is a form of peak shaving where transfer baggage is separated based on transfer time and fed in to the system on different moment based on separation instead of all at once. The goal is to spread out the morning peak. Cold buffering can be done in several ways with different design options, these are shown in table 3.1.

Transition time	Transport options	Buffer configuration	Buffer placement
x hours	Current transportation	Buffering with carts	1 location
y hours	Self-driving tug	Buffering with bins	2 locations
z hours	Self-driving baggage cart	Buffering individual bags	3 locations

Table 3.1: Design alternatives

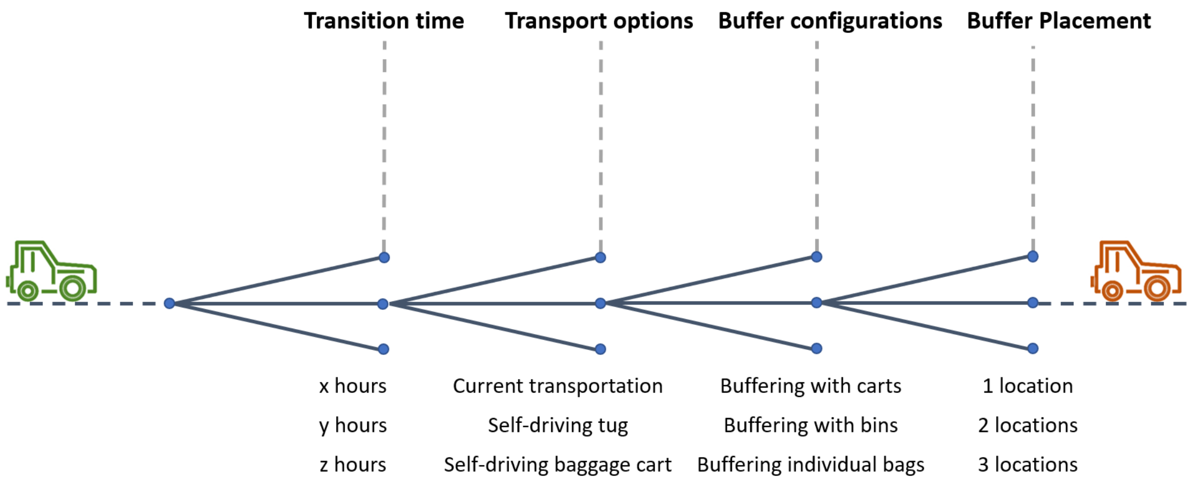


Figure 3.4: Design alternatives decision tree

From all the design alternatives one option could be chosen and combined with all the other options. This will give a decision tree with 81 outcomes. this is shown in figure 3.4. Not all the options fit logically together. If the transport option of self-driving baggage carts is chosen then it's smart to use the buffering configuration of buffering with carts because then an airport would need a lot more expensive self-driving carts. So if the option self-driving carts is chosen the buffer configuration will be buffering with individual bags where the bags are unloaded form the carts at a buffer. If the transport option of normal transportation is chosen it is more logical to use the buffer configuration where the carts are parked. This would not require extra personnel. From all the design options 3 configurations are chosen:

- 3 hours - Current transportation - Buffering with carts - one location.
- 3 hours - Self-driving tug - Buffering with carts - three locations.
- 2 hours - Self-driving baggage carts- Buffering individual bags - one location.

The configurations are further elaborated in chapter 7

4
Cold buffering at Schiphol airport

In the previous chapter cold buffering was explained and how this could be done. In this chapter a closer look will be taken in to cold buffering at Schiphol airport. The different design options will be specified, the data will be looked at and the first results of cold buffering are shown.

4.1. Design options

The different choices for transition time, transport options, buffer configuration and buffer placement will be more specified for what is possible at Schiphol airport.

4.1.1. Transition time

The transition time from cold baggage becomes hot baggage is based on the process time of the system, the width of the peak and the buffer period. At Schiphol airport the peak of incoming flights last from 7:00 till 9:00 in the morning as seen in figure 4.1. The peak of infeed baggage start around half an hour later at around 7:30 and lasts till around 9:30 as seen in figure 4.2. So the width of the peak at Schiphol airports last around two hours. Part of the throughput time of the transfer baggage is time it takes

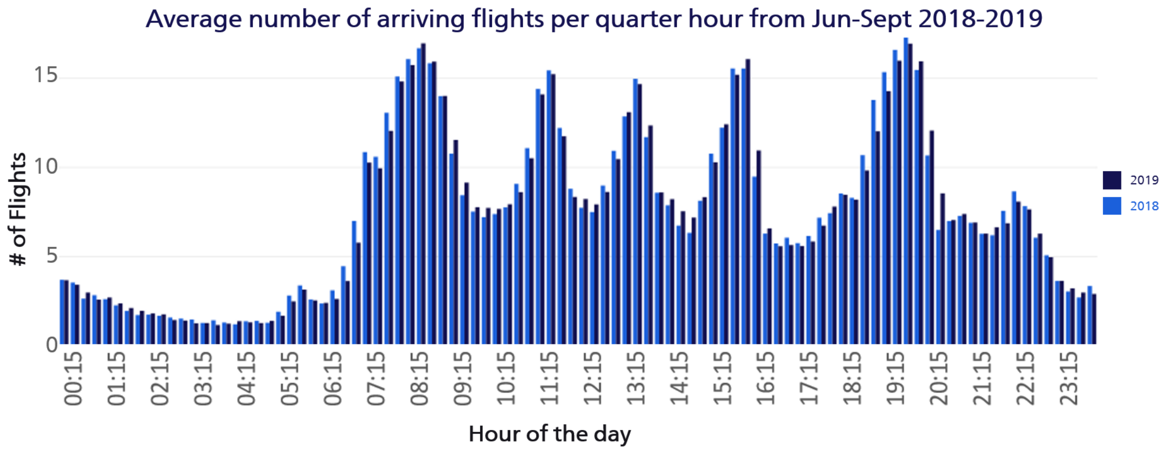


Figure 4.1: Distribution of flights during the day at Schiphol Airport (Royal-Schiphol-Group, 2019)

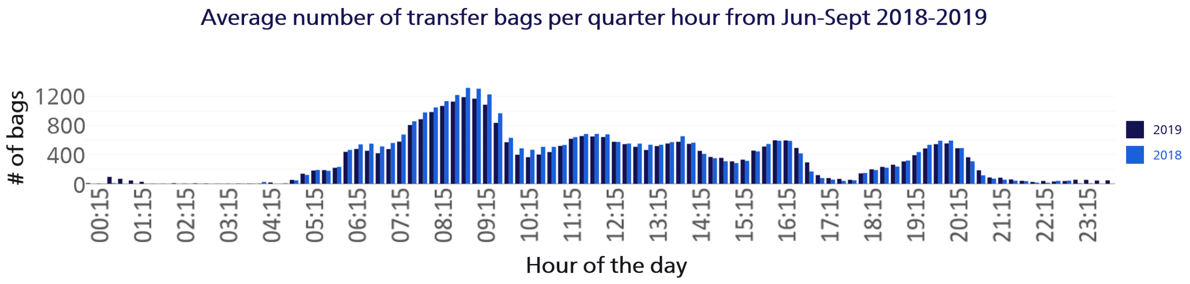


Figure 4.2: Distribution of transfer baggage during the day at Schiphol Airport (Royal-Schiphol-Group, 2019)

between plane being in blocks and the bag being entered in to the BHS. Based on data this takes 43 minutes on average during the peak hours. This is shown in figure 4.3. The average travel time is almost a constant during the day and goes up during the night because there is less personnel working at those moments. The travel time is measured from the point the plane is in blocks, this is the actual in block time (AIBT). The time the next plane, where the transfer baggage is going to, is taking off is called the actual off block time (AOBT) After the plane wheels are locked by the blocks the turn around services start. This includes the loading and unloading of passengers, catering service and baggage but also refuelling. After all the baggage is unloaded the carts or containers are linked up and picked up by a tug which brings them to the correct location. The unloading, coupling and driving time make up the 43 minutes. After the baggage is fed in to the system it has a process time depending on the in and

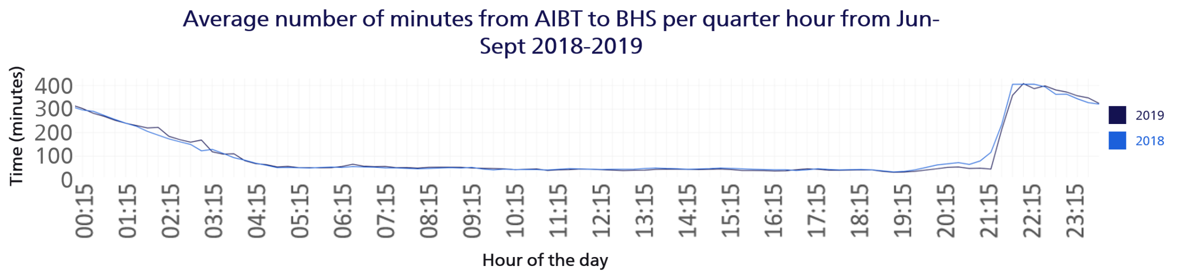


Figure 4.3: Average time between landing and being entered in to system during the day at Schiphol Airport (Royal-Schiphol-Group, 2019)

outfeed locations. This ranges from 6 to 24 minutes (Royal-Schiphol-Group, 2019). So the throughput time from AIBT to AOBT would take about one hour on average. With the two hour duration of the peak the most logical transition time would be three hours. But as design options it will be interesting to see what happens if the transition time is set either an hour less or more. If the transition time is set at two hours, the buffer time must only be one otherwise the baggage will not be on time for their next flight. Buffering for one hour might already be enough to shave the peak and more baggage will be classified as cold. If the transition time is set at four hours, less baggage will be classified as cold but the extra throughput time added by driving to and from a buffer will have less of an effect on the on time performance of the baggage.

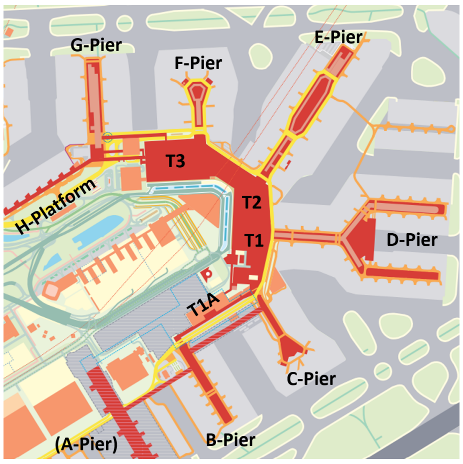
4.1.2. Transport options

In the previous chapter three transport options were explained. These three were the current way of transportation, so completely manual, transportation by a self driving tug and transportation with self driving carts. All of these three options could be applicable to Schiphol airport. The manual transportation is of course already in place. Manual driven carts have a maximum speed of 25km/h. eDuring the Covid period in 2021 trials took place with a self driving tug (Jacobs, 2021). The tug drove around using lidar and GPS was able to complete tasks of driving carts from loading and unloading areas to aprons and back. The tug had a maximum speed of 15 km/h so it was slower and it had to stop more often making it around 3 times slower. Currently there has not been any testing with self driving carts at Schiphol yet but successful experiments have taken place at other airports. These experiments also limited the driving speed of the self driving carts to 15km/h. So the carts will also go slower than manual driving but the carts can already start driving after they are loaded. This will save time as it does not have to wait before the whole plane is unloaded.

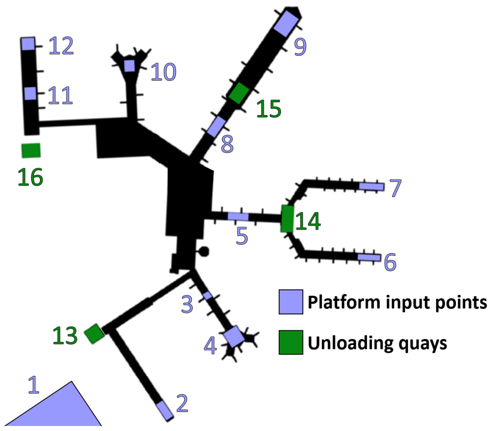
4.1.3. Buffer configuration

In chapter 3 the different options of buffer configurations were explained. The three possibilities are to either have a simpler buffer were carts and containers can be parked for a certain amount of time until they are driven to the infeed locations. The second option is to have a buffer where bags can be changed between carts and containers based on the scheduled departure time of their transfer flight. In this case this would be a manual process. The third option is to have an automated buffer storage where bags are unloaded and then stored individually until it becomes time to be brought to the infeed location either in batches or also individually. All three options are a possible to be implemented at Schiphol Airport and the first option has already been tried out successfully.

4.1.4. Buffer Placement



(a) Piers and terminals at Schiphol airport



(b) Map of platforms and unloading locations

Figure 4.4: overview of Schiphol airport

At Schiphol airport the placement of the buffer storage would have an effect on the configuration. In figure 4.4 the map of Schiphol is shown twice. In figure 4.4a the piers and terminals are indicated and in figure 4.4b all the unloading and platform areas. What the numbers specify is shown in table 4.1. For an airport like Schiphol it is difficult to find available space around the airport to place a buffer storage. Most available space is already in use. At the unloading areas at South, D-hall and E-hall there are shunting areas where unused carts and containers can be parked. These areas could be used as buffer storage, at either one location, two locations and even all three locations. If these shunting areas are used as buffer storage the configuration of the buffer will be the simple one as explained in the previous chapter. The carts or containers are parked and driven to the unloading quays after a certain amount of time without looking at individual bags. If it is possible to build a central buffer location then the other configurations with a smarter system might be implemented.

Nr.	Name	Description
1	A gebied + B-platform	A00 t/m A9 and B51 t/m B91
2	B-pier	B13 t/m B36
3	C-steel	C04 t/m C09
4	C-kop	C10 t/m C18
5	D-steel	D02 t/m D08
6	D-vorkzuid	D10 t/m D31
7	D-vorknoord	D41 t/m D99
8	E-steel	E02 t/m E07
9	E-kop	E08 t/m E99
10	F-pier	F01 t/m F99
11	G_steel	G02 t/m G06
12	G_kop	G07 t/m G80, H1 t/m H7 and M1 t/m M7
13	Unloading area South	4 unloading quays
14	Unloading area D	5 unloading quays
15	Unloading area E	2 unloading quays
16	Unloading area West	1 unloading quay

Table 4.1: Description of the numbers shown in figure 4.4b

4.1.5. Design options table

All the different design options are put together in table 4.2 and visualised in figure 4.5. All possible combinations could be used resulting in 81 posibilities.

Transition time	Transport options	Buffer configuration	Buffer placement
2 hours	Current transportation	Buffering with carts	1 buffer
3 hours	Self-driving tug	Buffering with bins	2 buffers
4 hours	Self-driving baggage cart	Buffering individual bags	3 buffers

Table 4.2: Design options for Schiphol Airport

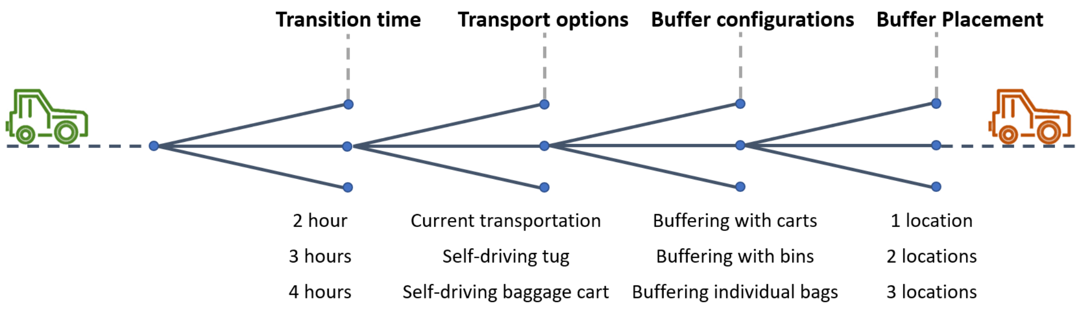


Figure 4.5: Design options for cold buffering at Schiphol airport

4.2. Data analysis

The IT&Data department of the Schiphol Group has provided a way to obtain data on transfer bags that arrive and depart form Schiphol. The data that was provided gave all the known information on individual bags. It contained the scheduled arrival time, the actual arrival time, the arrival gate, the airline, the flight number, the ground handler, the infeed time of the BHS, the entry station of the BHS, the exit station of the BHS, the exit time of the BHS, the flight number of the transfer flight, the departure gate, the scheduled departure time , the actual departure time and aircraft type of both the flights. To download all the correct data of one day the search net needed to be 3 days, also including the day before and after the specific day you want. Otherwise there might be information missing on the either the arriving flight if it has landed a day before or if the transfer flight is leaving a day later. This was done 7 times to obtain one week of data. For every day of this week the infeedtimes of the bags of the system were grouped together to see how many bags enter the system in in time interval of 15 minutes. The distribution over the day is shown for the whole week in figure 4.6.

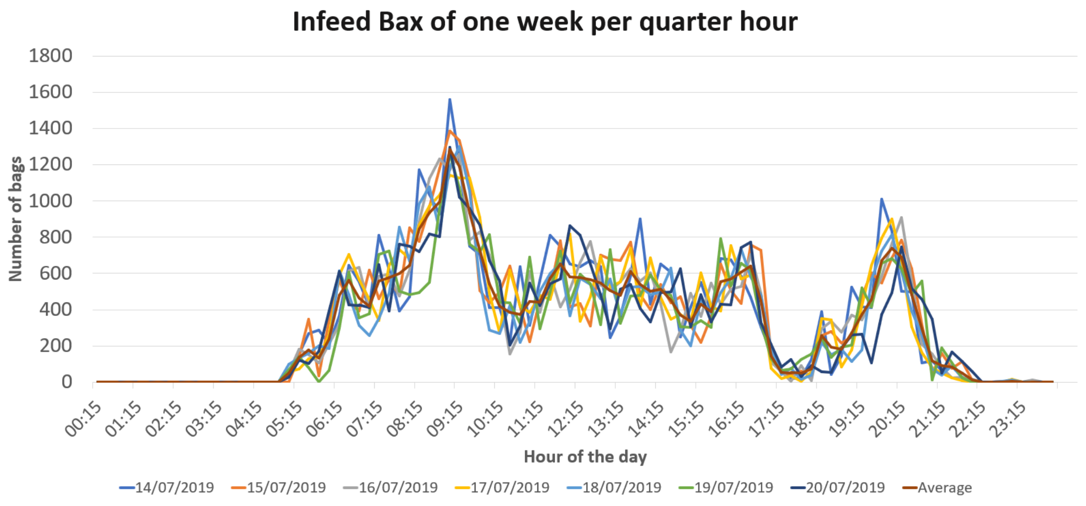


Figure 4.6: Distribution of baggage entering the BHS over the day

In this figure it can be seen that it follows the same trend as figure 4.2 and that 14/07/2019 has the highest morning peak. So it was decided to further investigate this day because the highest potential of peak shaving could be achieved here. In figure 4.7 the distribution of transfer baggage of only the 14th is shown. The peak is set at 1549 bags that were entered in fifteen minutes.

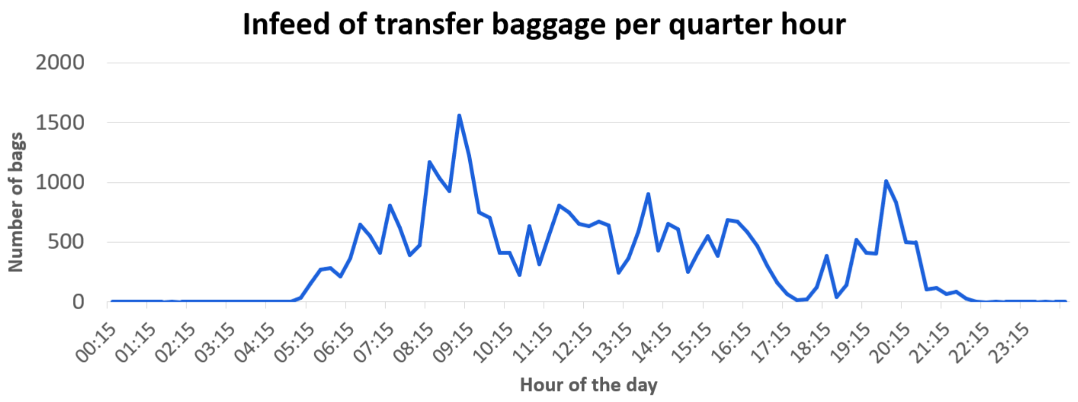
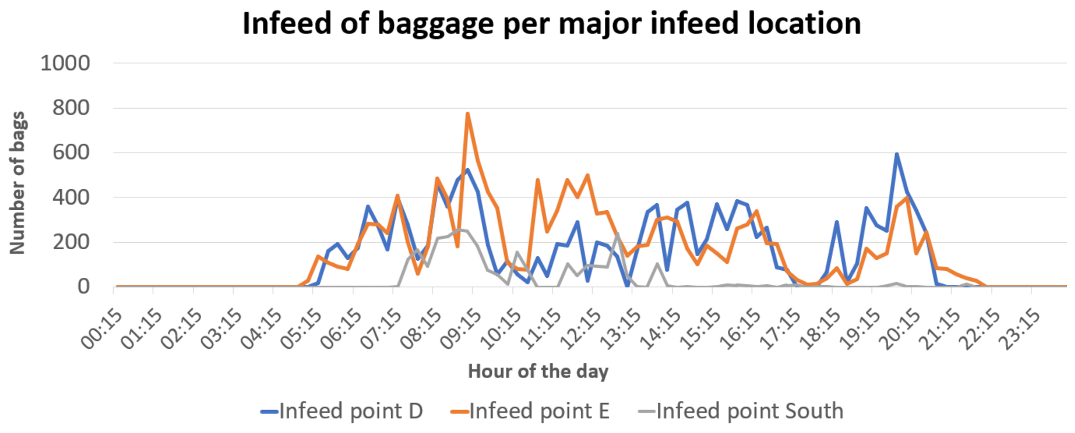
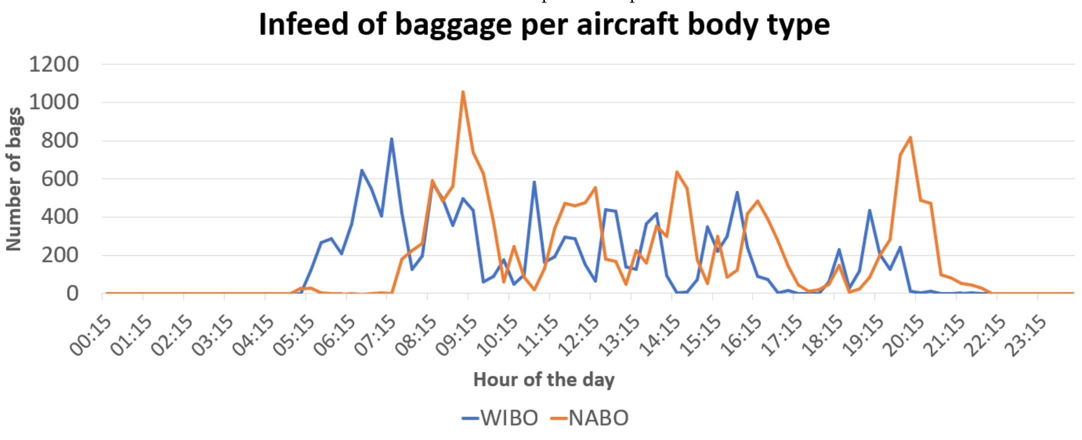


Figure 4.7: Distribution of baggage entering the BHS over the day on 14-07-2019

From the data it was known at which entry points the baggage was fed in to the system. figure 4.7 shows the sum of all these entry points. In figure 4.8a a distinction is made between the three major entry points, D, E and South. From the aircraft type given in the data it was derived if this was either a narrow body (NABO) airplane or a wide body (WIBO) airplane. The body type on which the bag came in with was added as an extra data entry and based on the body type the same graph of the distribution of infeed at the BHS was made for both aircraft types. This is shown in figure 4.8b. What can be observed here is the most WIBO baggage arrives earlier than NABO baggage, this is due to the fact that KLM let's their intercontinental flights arrive first and than the European flights so that the intercontinental passengers can transfer to those European flights. The peak of infeed point D is at



(a) Distribution per infeed point



(b) Distribution per body type

Figure 4.8: Distribution of baggage entering the BHS over the day on 14-07-2019

524 bags per quarter hour, the peak of infeed E is at 774 bags per quarter hour and the peak of infeed point south is at 257 bags per hour. All the infeed points have a specific capacity. Infeed point D can handle 4500 bags per hour using 5 unloading quays, infeed point E can handle 3600 bags per hour using 4 unloading quays and infeed point south can handle 200 bags per hour using 2 unloading quays. This brings the total capacity of these three unloading points to 9100 bags per hour.

4.3. First set-up

In 2019 KLM and Schiphol Group for saw problems for the summer holiday of that year in the morning hours at the infeed capacity of transfer baggage. They decided to do a trial with cold buffering on WIBO and NABO airplanes. This meant that they would be buffering containers as well as bulk baggage. The way this was done for the containers is that the containers were either loaded with cold baggage or hot baggage at the departing airport. A minimum of 25 cold bags was needed was needed to fill a container and label it cold. A containers can fit up to 42 bags. If their would be less than 25 bags it would just be combined with hot bags and labeled as hot to make efficient use of the space available inside the airplane. Bags would be classified cold if the transfer time was above 3 hours. They way this was done for bulk baggage is that the labels were printed with a C on it at the departing airport and added to the baggage. With this the ground handling service at Schiphol airport could more easily detect cold baggage at put it on a different cart. This meant the detection and separation was done completely manually. All the cold bags were transported manually to a buffer at the South unloading area where they were stored for two hours in the same carts and containers. After the two hours the baggage was fed in to the system. Looking at table 4.2 this results in the following decision tree as seen in figure 4.9.

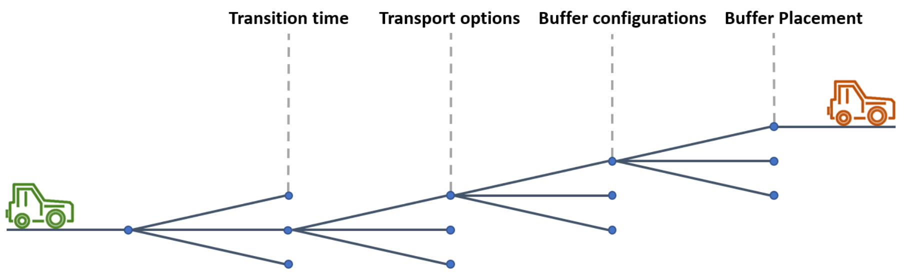


Figure 4.9: Chosen Design options in the KLM trial

The results of the first trial were interesting. During the trial the cold buffering of bulk baggage was stopped early because too many bags got lost. The cold buffering of containers showed successful and the target peak shaving was achieved. The goal was to be able to take out 1400 bags during the two peak hours so 700 bags per hour.

4.4. First results

Using the SIBT and the SOBT the transfer time of a bag can be calculated. If this transfer time is longer than the transition time that is set, than a data entry will be added with the temperature cold. If the transfer time is lower the temperature will be set to hot. With the baggage data from the 14th of July it is possible to replicate the trial KLM ran. All the bags that were entered between 8 and 10 am in to the infeed and that had a cold temperature were 'buffered' for two hours. This meant adding two hours to the actual infeed time to see the difference on the distribution of infeed baggage during the day. This result is shown in figure 4.10. The original peak of 1549 is reduced to 1189 bags that are fed in per

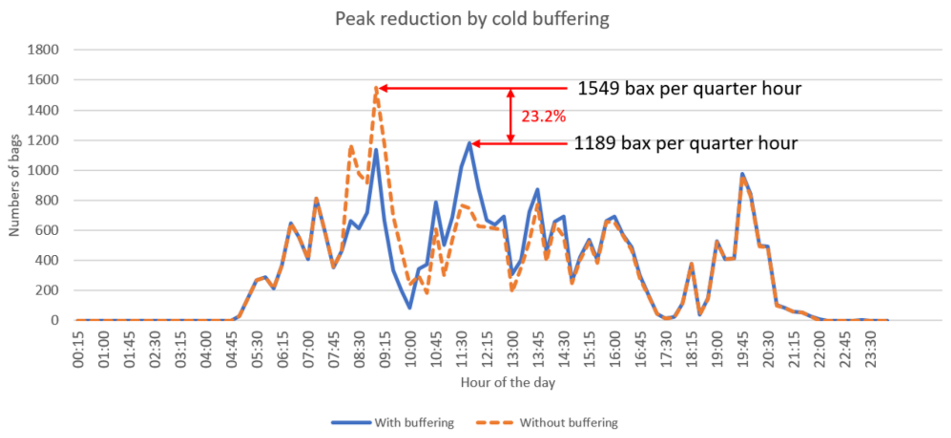


Figure 4.10: Distribution of infeed of baggage with and without cold buffering

quarter hour. This peak reduction of 360 bags is a reduction of 23.2%. In total 7209 bags were fed in to the system between 8 and 10 am. Of these 7209 bags, 3093 were cold and 4116 were hot. Of the 3093 cold bags, 1402 came from containers and 1691 from bulk baggage. With buffering all the bags a new peak arises as well, this happens around 11:30, as seen in figure 4.10. The key performance indicators of a process using buffering would be the newly created peak in the overall system, the amount of tugs it would take to process all the baggage and the pressure on the individual infeed points.

4.5. Conclusion

Schiphol airport sees a high peak of infeed baggage in the morning. The entire baggage handling system is designed to handle this peak but it is possible to prioritize baggage and to spread out this peak. The data shows a large reduction is possible, even more than is needed. There are several ways of obtaining peak reduction. This ways differentiate themselves in the design options that are made. The design options are split up in to four categories: Transition time, transportation options, buffer configurations and buffer placement. All these four categories have three options. These are were given in table 4.2 and shown in figure 4.5. In chapter 7 the chosen configurations based on these design options will be given. The KPIs that will be derived from the modelling of the cold buffering process must be able to determine the peak of bags that are fed in, in a specified time window. The second KPI is the amount of tugs it takes in order to bring the hot bags to the infeed locations and the cold bags to the buffer storage. The third indicator is the pressure on the individual infeed locations E, D and South.

5 Modelling methods

In order to test out different design options a model is needed. Several ways of modelling can be found in literature such as predictive modelling, simulation, optimization and queue modelling. Those ways of modelling will be discussed in this chapter. And a comparison is made in order to select the best method for this research.

5.1. Predictive modelling

Predictive modelling is a technique that uses machine learning, historical data and existing data to predict and forecast the most likely future outcome. It analyses the historical data to generate patterns. The existing data and the patterns are then used to predict the future events. The model is not fixed but can be revised and validated regularly to stay up to date. For the baggage process this means processing historical data of where baggage has been and which route the baggage takes. (Ma et al., 2021) has done research to predicting check in baggage based on a time series analysis. They made use of a seasonal autoregressive integrated moving average (SARIMA) model. In this research they concluded that based on the historical data they could make an accurate prediction of the amount of checked in baggage per time slot per day. By doing this it is possible to allocate more personnel on the peak moments to decrease the waiting time for passengers. There were also some shortcomings according to the researchers, with this model it was harder to make long term predictions especially during the holiday periods.

Another part of the baggage handling process at an airport is the transport of the bags,(Tarau et al., 2009) takes a look model predictive control (MPC) for route choices in an automated BHS. In their research the transportation of baggage is handed by destination coded vehicles (DCVs). The bag is dropped onto the DCV, this is visualized in figure 5.1, and transported along high speed network of tracks. The MPC was done in three ways, centralized, decentralized and distributed. The results of the research show that a centralized control has the best performance but this method requires too much computational effort in a system with several junctions. A trade off must be made between time required to compute the routes for the DCVs and the performance of the system. Both the decentralized and the distributed method offers a balanced trade-off between these to.

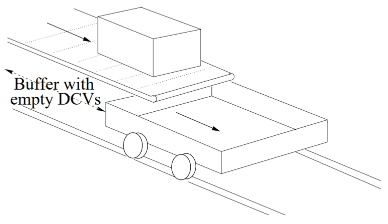


Figure 5.1: Loading process of a DCV (Tarau et al., 2009)

(Zeinaly et al., 2012) did a follow up research on DCVs, they made predictive control model to determine the optimal amount of DCVs while minimizing the overall baggage waiting time and the overall energy consumption of the system.

5.2. Simulation modelling

Simulation modelling is a technique where a digital copy of a real world process is created to analyse and predict it's performance. This has been done several times in literature such as by (Cavada et al., 2017). They made a simulation of Santiago Airport in Chile. The amount of passengers has outgrown the BHS's capacity. The goal of the simulation is to accurately predict the outcome of different scenarios and to see how the system reacts. Their simulation is model of the whole baggage journey divided in to three parts, check in, behaviour (transportation) and baggage loading area. For each part the sub process are simulated but the interactions between the processes are also investigated. As a basis for the transportation part of the model, vehicle traffic microsimulation software was extended to simulate all the conveyor belts.

(Le et al., 2012) also made simulation model but a more generalized one where there is less interaction between the sub systems, their goal was to take a look at the most important key performance indicators such as peak throughput and in-system time of bags. Based on the KPIs, estimations are made on the number of replications that are needed in the simulation, warm-up and cool down times and system recovery times.

(Wuisman, 2016) did research on simulation at KLM ground services and specifically on transfer baggage. The goal was to simulate different scenarios of handling transfer baggage from unloading the plane to the infeed areas. There were several different alternative standards to assess the process in order to optimize the overall performance of the system. These alternatives standards were the corrected number of mishandled bags, required tug driver capacity and required unloading quay capacity. The simulation model was made in Python using Pandas and Simpy. The reason for using Python was to keep the simulation available inside KLM without anyone having to learn other coding languages or KLM having to buy licenses for existing software packages.

There are also existing software packages for simulation modelling at airports,(Hafilah et al., 2017) used Petri Net (PN). PN makes it possible to model and visualise behaviour comprising resource sharing, synchronisation and concurrency. (Lin and Liou, 2015) proposes the use of the modelling framework SysML, SysML stands for system modelling language. The advantage of using this framework was that SysML supports data and information sharing between domain experts and because the language is object oriented a wide range of systems is reusable. Another software than has been used is Arena(AlKheder et al., 2019). (Malandri et al., 2018)used this to simulate the inbound baggage at bologna airport in Italy. In the model a distinction was made between Schengen an non-Schengen flights which is an important detail to take into account while simulating other European airports. A delay of 30 minutes was added to the non-Schengen baggage to compensate for the time it goes through security. In another research of (Lin et al., 2015) they use the simulation model Flexim to simulate the baggage flows at Taiwan Airport. This software also produces 3D models of the system which is shown in figure 5.2 Using this software they tested different interval release times of bags going through the scanner and in to the buffer. With the optimal interval they were able to be 23% more efficient at peak capacity.

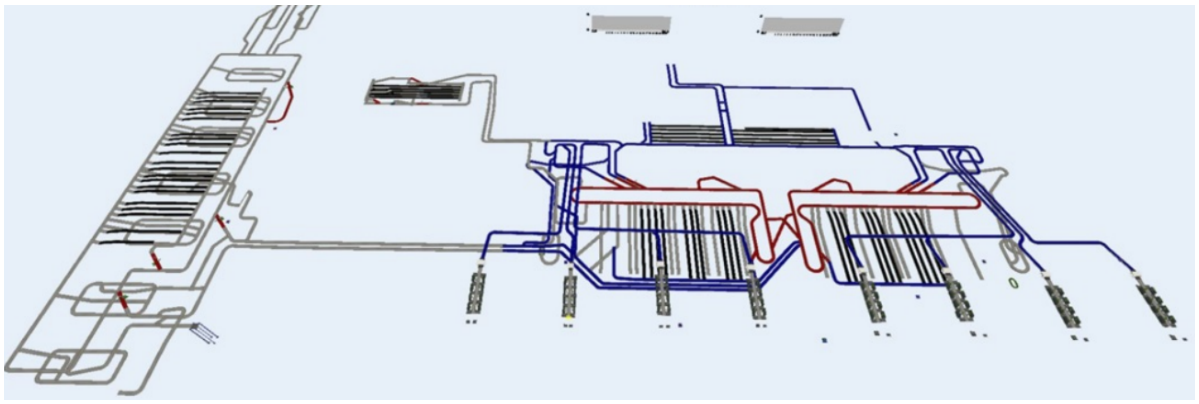


Figure 5.2: Flexsim simulation 3D model

5.3. Optimization modelling

Optimization modelling attempts to find the optimal solution to a single or multiple complex equations using a set of constraints. There are four programming techniques that can be used. Linear Programming (LP), Mixed Integer Programming (MIP), Nonlinear Programming (NLP) and constraint programming (CP). Most literature found on optimization of baggage handling at airports makes use of MIP, such as (Barth et al., 2021) and (Holm et al., 2013). (Huang et al., 2016, Huang et al., 2018) uses robust optimization to assign make-up areas to outgoing flights. The goal is to minimize the amount of unassigned flights. The BHS makes us of the assigning of flight by sorting the bags and transporting it to the right place. This decreases the manual labour that needs to be done and thus will increase the efficiency of the overall system. Sets, indices, variables and parameters are needed for optimization. (Loenhout, 2019) also did research on the subject of outbound baggage planning, this research was done for KLM at AAS. In this research the amount of personnel is also taken into account and one of the objectives is to minimize this amount. This includes drivers and personnel in the baggage hall. (Bruno and Genovese, 2010) made an optimization model to decide the optimal amount of check in gates to balance the operating cost and the waiting time for passengers. This was mixed integer programming. (Markus and Frey, 2014) also worked on optimization of baggage at an airport, they looked at the entire baggage process explained in chapter 2. All the flows were mathematically formulated by a generic assignment scheduling problem (GASP).

5.4. Queue modelling

Queue modelling can be used to predict queue lines and waiting times. In literature the only application for queue modelling at an airport for baggage was found at the check in counters. (Parlar and Sharafali, 2008) did research on this subject. Their goal was to optimize dynamic assignment of check in counters for a flight with a known amount of passengers. For this research they created their own passenger interval arrival rate given over different subintervals. The arrival rate of passenger can be connected to the arrival rate of baggage. The research was only done for a single flight and did not include all check in streams.

5.5. Comparison

The four modelling types explained above all have their own advantages and disadvantages. Predictive modelling requires a lot of data as input for the model to find the patterns it needs to predict, it also needs another dataset to validate this. Predictive modelling cannot show what might happen if another scenario is implemented. This is something that simulation can do. Physical real world test can be expensive for airports and might interfere with the day to day operations. By using simulation modelling it becomes possible to test different solutions and see what effect they have on the performance of a system. If all the scenarios are known and simulated it might be possible to optimize specific KPIs with given constraints. This is where optimization modelling would come in to play. Queue modelling can only be used on specific parts of the baggage journey and will not be able to predict the capacity and efficiency of the whole system.

5.6. Conclusion

The goal of the model will be to test different scenarios of cold buffering to see what effect it has on the peak at the infeed of the system. The process of arriving airplanes, unloading and transportation must be modeled bases on real life data in order to validate this process. After that the cold buffer should be added and then the effect of this process will be seen. Based on the arguments given in the comparison of the models it is clear that simulation modelling will achieve the goal better than the others. Queue modelling cannot be used for the entirety of this process, not all the constraints are known to optimize this process and with predictive modelling it is not possible to test different scenarios. The way the simulation model will be set up is explained further in chapter 6.

6 Simulation modelling

This chapter further elaborates the development of the simulation model. Firstly the goal of the model will be explained, secondly the system boundaries and inputs will be given. After that the objects that are needed in the simulation model will be elaborated.

6.1. Main goal

The main goal of the simulation model is to replicate a part of the baggage journey in order to find the pressure on the infeed capacity of the system. The baggage journey, including cold buffering, is shown in figure 6.1 and fully explained in chapter 2.

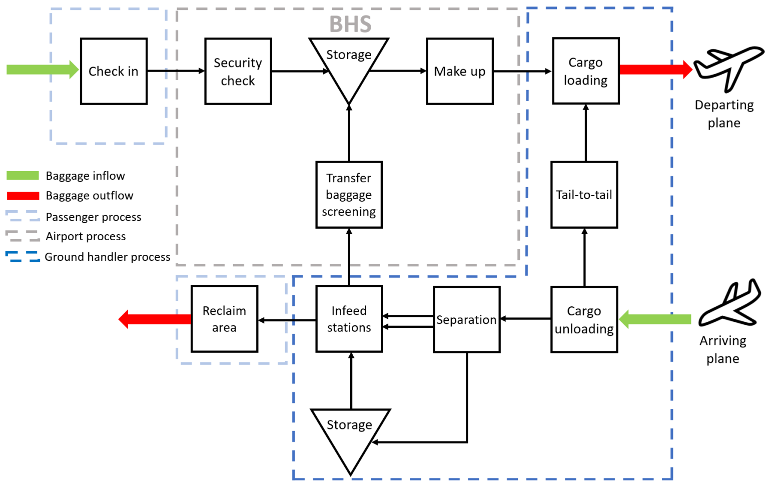


Figure 6.1: The baggage journey with cold buffering

The simulation model should be able to simulate the process for transfer baggage from an arriving airplane until it arrives at an infeed point. The model should be able to compare the performance of the system when the cold buffer is being used and when it is not being used. The performance of the system is measured by the pressure on the capacity during the morning rush hours. In chapter 4 it was discussed that there are several different ways of implementing the cold buffer, the simulation model must be able to simulate different scenarios.

6.2. Model boundaries with input and output

As said in the main goal, the aim of the model is to simulate the part of the baggage journey from the arriving airplane to the infeed points of transfer baggage. In figure 6.2 this journey is shown. It is simplified from figure 6.1, the tail to tail and reclaim baggage is taken out. The input of the model will be airplane data. This contains the AIBT of the arriving airplane, the body type of the airplane the gate location, the amount of bags on board with a known amount of hot and cold bags and the infeed location is predetermined. The output of the model is an individual infeed time for each bag that arrives at the airport. These infeed times

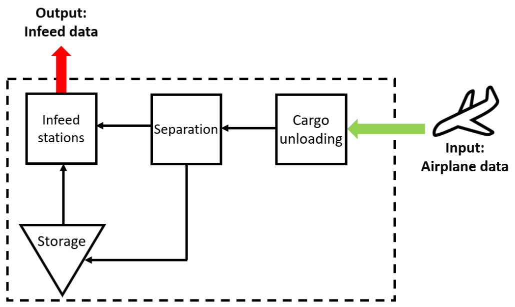


Figure 6.2: Boundaries of the simulation model

can be grouped together in timeslots in order to create an infeed distribution over the day and find the peak. The infeed locations of the bags are known in the simulation so the output of the system can be shown for the total system and for each individual infeed point. The last output of the model will be the amount of tugs it takes to transport all the baggage. So the three outputs are:

- Infeed time of each individual bag
- Infeed location of each individual bag
- Number of tugs necessary to transport all the bags

6.3. Model parameters

The baggage journey from landing to infeed is shown in figure 6.3 with specific time steps.

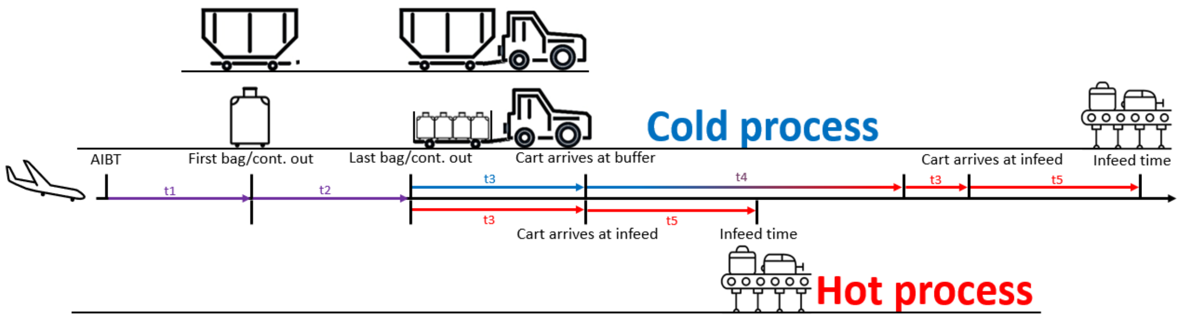


Figure 6.3: transfer baggage journey from airplane to infeed with timesteps

6.3.1. Processing time from in blocks till unloading: t1

t1 is the time it takes between when the airplane is in blocks, the AIBT and the moment the unloading process is able to begin. During a turnaround several processes happen and the unloading of the airplane needs to be prepared. In an existing Schiphol model this time was chosen from a uniform distribution between 120 and 180 seconds. Personal measurements that were taken were more around 240 seconds. t1 is the same for WIBO and NABO unloading. For the simulation the uniform distribution between 120 and 180 seconds was chosen.

6.3.2. Total unloading time: t2

t2 is the time it takes to unload one bag or one container. t2 for bulk baggage is chosen, by the existing Schiphol model, from a uniform distribution between 6 and 7.5 seconds. Personal measurements confirm this. t2 for containers is chosen, by the existing Schiphol model, from a triangular distribution between 40 and 100 seconds with a mode at 60 seconds.

6.3.3. Driving time: t3

t3 are the driving times from the gate to the infeed area or to the buffer area. There are several sources for these driving times from the Schiphol model, KLM data, personal measurements and other trials. All the driving times are exact times.

6.3.4. Buffer time: t4

t4 is the time chosen to buffer the baggage. This is a variable that is depended on the design choices that are made.

6.3.5. infeed time: t5

t5 is the time it takes to unload a bag from the cart or the container and put it in the infeed station. This time was set at 3.5 seconds in the existing Schiphol model

6.3.6. Number of bags on the cart or in the container

The number of bags that fit in the cart or in the container is a variable that is based on the size of the bags. Sometimes there are more smaller bags and so more fit in in. The existing Schiphol model takes a triangular distribution for both. For the cart it is between 20 and 40 bags with a mode at 30. For containers it is between 30 and 42 with a mode at 38.

6.4. Components in the simulation model with their PDL

In order to create the simulation model different components must be modelled. These components are shown in table 6.1 with a one line description. The components are explained in further detail below with their attributes and Process description language (PDL).

Component	1-line description
AirplaneGenerator	Creates airplane at specified times
Airplane	Airplane with a body type (wide or narrow), number of bags and specified gate
Container	Containers are unloaded from WIBO airplanes and brought to next location
Cart	Carts are filled with bulk baggage from NABO airplanes and brought to next location
Tug	The tug transports the cart or containers to the next location
Infeed	End destinations of the baggage in this model
Buffer	Storage for cold baggage

Table 6.1: Simulation objects

6.4.1. AirplaneGenerator

The AirplaneGenerator generates the airplanes that land on the airport. The times at which an airplane is generated is given by the input data. From the same input data the arrival area (gate) is taken as well as the intended infeed area and the amount of bags inside the plane. The arrival areas are the same as in figure 4.4b

Attributes:

ArrivalTime	Arrival time of the generated airplane.
ArrivalArea	Arrival area of the generated airplane.
BagsPerPlane	Number of bags on the generated airplane.
EntryArea	Infeed location for the bags from the generated airplane.
BodyType	Body type of the generated airplane, WIBO or NABO.

PDL:

```
Process
    Repeat
        newAirplane=TAirplane
        newAirplane.AA=ArrivalArea
        newAirplane.EA=EntryArea
        newAirplane.BO=BodyType
        newAirplane.NrB=BagsPerPlane
        Wait till next ArrivalTime
```

6.4.2. Airplane

When the airplane is created it is generated at the AIBT, as seen in figure 6.3 the airplane must then wait a certain time period before it can be unloaded. This time period is chosen from a uniform distribution with a upper and lower bound. After that the airplane will be unloaded with carts or containers based on the body type of the container.

Attributes:

LowerBoundT1	Lowerbound of the uniform distribution for T1
UpperBoundT1	Upperbound of the uniform distribution for T1
NrB	Number of bags on the plane
AA	Arrival area of the airplane
EA	Entry area of the airplane
BO	Body type of the airplane

PDL:

```
Process
    Repeat
        T1=sample(uniform(LowerBoundT1, UpperBoundT1))
        Hold T1
        If BodyType == NABO
            Select Cart
            Enter Cart.MyQueue
        If BodyType ==WIBO
            Select Container
            Enter Container.MyQueue
```

6.4.3. Container

The containers are loaded out of of the airplane and placed on specific carts that are designed for containers. The amount of bags that fit inside a container is based on a triangluar distribution with a lowerbound a uperbound and a mode. The amount of containers inside the aircraft can be determined by the total amount of bags in the plane and number of bags that fit inside the container. The total unloading time T2 is based on the amount of containers inside the airplane and the time it takes to unload one. The time it takes to unload one container is chosen from a uniform distribution with a lowerbound and an upperbound. After the containers are unloaded they will be picked up by tugs to be transported to the infeed locations.

Attributes:

LowerBoundT2	Lowerbound of the uniform distribution for T2 for containers
UpperBoundT2	Upperbound of the uniform distribution for T2 for containers
LBContainer	Lowerbound of the triangular distribution for bags in a container
UBContainer	Upperbound of the triangular distribution for bags in a container
ModeContainer	Mode of the triangular distribution for bags in a container
Container.MyQueue	The container Queue

PDL:

```
Process
        Repeat
                FirstContainer = Container.MyQueue.pop
                FirstContainer.EntryArea = newAirplane.EA
                FirstContainer.ArrivalArea = newAirplane.AA
                BagsinContainer = sample(triangular(LBContainer,UBContainer,ModeContainer))
                NrofContainers= ceil(newAirplane.NrB/BagsinContainer
                T2 = sample(uniform(LowerBoundT2, UpperBoundT2))
                Hold (T2*NrofContainers)
                Select Tug
                Enter Tug.MyQueue
```

6.4.4. Cart

The cart process undergoes a similar process as the container. Firstly all the bags must unloaded. The times it takes to unload one bag is given by a uniform distribution with a lowerbound and an upperbound. The amount of carts that are needed to unload all the bags on to is based on the amount of bags that fit on one cart. This is based on a triangular distribution wit a lowerbound an upperbound and a mode. After all the bags are unloaded the carts will be picked up by a tug and brought to the infeed location.

Attributes:

LBT2	Lowerbound of the uniform distribution for T2 for carts
UBT2	Upperbound of the uniform distribution for T2 for carts
LBCart	Lowerbound of the triangular distribution for bags on a cart
UBCart	Upperbound of the triangular distribution for bags on a cart
ModeCart	Mode of the triangular distribution for bags on a cart
Cart.MyQueue	The cart Queue

PDL:

```
Process
        Repeat
                FirstCart = Cart.MyQueue.pop
                FirstCart.EntryArea = newAirplane.EA
                FirstCart.ArrivalArea = newAirplane.AA
                BagsonCart = sample(triangular(LBCart,UBCart,ModeCart))
                NrofCarts= ceil(newAirplane.NrB/BagsonCart
                T2 = sample(uniform(LBT2, UBT2))
                Hold (T2*NrofCarts)
                Select Tug
                Enter Tug.MyQueue
```

6.4.5. Tug

The tugs pick up the containers and carts and drive them to the the correct infeed areas. The time it takes to drive them there is based on arrival area and the infeed area. A table is made with the driving times from all the arrival and entry areas and based on those the correct driving time will be chosen. After that the time the tug arrives at the entry area.

Attributes:

Tug.MyQueue	The tug Queue
DrvingTimes	A table with all the driving times from all the arrival areas to the entry areas

PDL:

```
Process
    Repeat
        FirstCart = Tug.MyQueue.pop
        FirstCart.EntryArea = newAirplane.EA
        FirstCart.ArrivalArea = newAirplane.AA
        FirstCart.NrB = newAirplane.NrB
        Hold (DrivingTimes[ArrivalArea,EntryArea])
        If EntryArea == E
          Select infeedpoint E
          Enter InfeedE.MyQueue
        If EntryArea == D
          Select infeedpoint D
          Enter InfeedD.MyQueue
        If EntryArea == Z
          Select infeedpoint Z
          Enter InfeedZ.MyQueue
```

6.4.6. Infeed

In the tug process the correct infeed area is chosen, the process at these three infeed areas is the same. The tug arrives at an infeed area and picks a infeed point if one is available. If that is the case the infeedprocess starts and all the bags from the carts or containers are fed in to the system. The time it takes to feed in one bag is the InfeedTime. Every bag gets an individual feed in time.

Attributes:

Infeed.MyQueue	The infeed Queue
InfeedTime	The time it takes to infeed one bag

PDL:

```
Process
    Repeat
        FirstTug = Infeed.MyQueue.pop
        FirstTug.NrB = FirstCart.NrB
        Hold (UnloadingTime*FirstTug.NrB)
        Carts, Tugs and infeed point queue become available again
```

6.4.7. Buffer

Carts and containers are brought in by tugs to the buffer to be stored here. After a buffering period the bags are brought to the infeedstations.

Attributes:

Buffer.MyQueue	The Buffer Queue
DrivingTimes	A driving time from the buffer to the infeed area
BufferTime	The time the cold bags should be buffered
BufferArea	The infeedarea where the buffer will be located

PDL:

```
Process
        Repeat
                FirstCart = Buffer.MyQueue.pop
                FirstCart.EntryArea = newAirplane.EA
                FirstCart.BufferArea = newAirplane.BA
                FirstCart.NrB = newAirplane.NrB
                Hold (BufferTime)
                Hold (DrivingTimes[BufferArea,EntryArea])
                If EntryArea == E
                  Select infeedpoint E
                  Enter InfeedE.MyQueue
                If EntryArea == D
                  Select infeedpoint D
                  Enter InfeedD.MyQueue
                If EntryArea == Z
                  Select infeedpoint Z
                  Enter InfeedZ.MyQueue
```

6.5. Verification of the model

In order to verify this model certain parameters will be changed which have a logical real world effect. For instance if the driving times are multiplied by a factor 100, the infeed times of the bags should become later. This is tested with multiple relationships given in table 6.2

Changed input	Change	Expected effect	Effect	Check
Driving times	x100	Delayed infeed times	The graph shifts to the right	✓
Unloading times	x100	Delayed infeed times	The graph shifts to the right	✓
Infeed times	x0	Earlier infeed times	The graph shifts to the left	✓
Less tugs available	x0.25	Delayed infeed times	The graph shifts to the right	✓

Table 6.2: Different verification tests ran on the model

The results are plotted in figure 6.4. The blue line in figure 6.4 is the result of the model with normal chosen input parameters. This is the base line to see what changes occur during a test. The first test is to multiply the driving times by a 100. The expected effect of this change is that baggage will arrive later at the infeed points. In figure 6.4 test 1 is shown with an orange line and has moved to the right, most bags will be fed in after midday and a lot of bags are fed in a day later, which is not shown in this graph. The second test was kind of similar to first test but this time the unloading times of the bags from an airplane were multiplied by a 100. This should yield a similar effect to test 1. The result of test 2 is shown in black in figure 6.4, the line is also shifted to the right. Test 3 was to see if changing times to something lower would also work. The infeed times at the infeed stations were set to 0. The expected effect is that bags are fed in earlier and that the graphs shifts to the left. The effect is shown with the yellow line in figure 6.4, it has indeed shifted to the left. The last test, test 4 has to do with the available amount of tugs. If the available amount of tugs is lower than the amount needed it is expected that bags will have to wait on the apron and thus be fed in later in to the system. The total amount of available tugs was divided by 4. The result of this simulation is shown with the green line in figure 6.4, bags are fed in later during the day compared to the baseline and the amount of bags fed in per quarter hour remains similar during the day because this is now limited by the capacity of the tugs.

6.6. Data validation

In order to validate the accuracy of the model the input data itself must first be validated. Inaccurate input data could be a source of inaccuracy of the output of the simulation model (Robinson, 1997). The input data set contains several timestamps of the baggage journey of one bag. The infeed time, outfeed time, scheduled and actual arrival times and scheduled and actual departure times are given. Based on these times certain checks can be done. The baggage cannot enter the BHS earlier than actual arrival time of the airplane. This was the case for certain data entries and these were deleted from the data set. Another check that was done was to see if the arrival date was correct. It was known that airplanes that depart before midnight but arrive after midnight sometimes are given the departure day as arrival day.

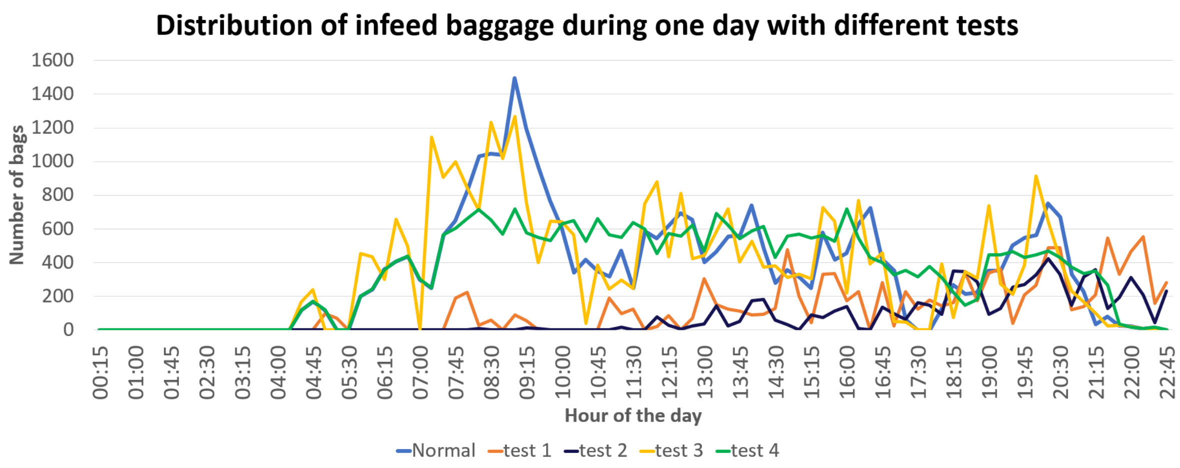
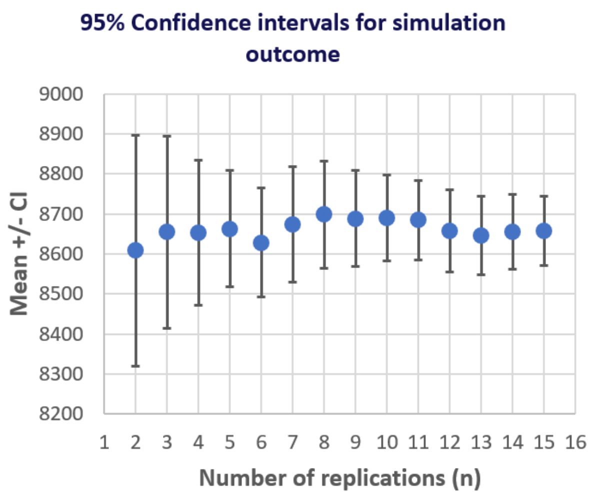


Figure 6.4: Different verification tests ran on the model

The time difference between infeed time and arrival time was calculated, if this took longer than one day it was assumed that the wrong date had been entered. These data entries were also deleted. The remaining data was plotted in figure 4.7. This day seemed to follow the trend of the 5 peaks shown in figure 4.1

6.7. Sensitivity for random seed

In the PDLs of the components in the simulation models it can be seen that some stochastic variables are chosen using a distribution. This distribution is based on randomness and simulation models normally only use one seed to be able to reproduce the outcome of this model. If another seed is set at the start of the simulation the randomness will change. The random seed has an effect on the waiting time of airplane once it lands until unloading can begin, it also has an effect the unloading time itself and the amount of carts and containers needed. To test the sensitivity of the model multiple replications were done to see what the effect was. The outcome that was tested were the total number of bags fed in during morning peak hours for the infeed stations, so between 7:30 and 9:30. In total 15 replications were done. The results of the different replications were compared and their mean, standard deviation and confidence interval were calculated. The confidence interval was calculated to include 95% of the results in the interval. The results can be found in section 6.7 and section 6.7. The confidence interval is slowly getting smaller after almost each replication added. Only at n=7 and n=12 it goes up slightly. Based on the figure and percentage that the confidence interval is over the mean it chosen to do 13 replications to ensure a low sensitivity of the random seed of the outcome without making the simulation longer than it has to be. After 13 replications the mean has become stable and the changes after that are minimal. The run time of the model is one day and it takes the model roughly 4 minutes to simulate this.

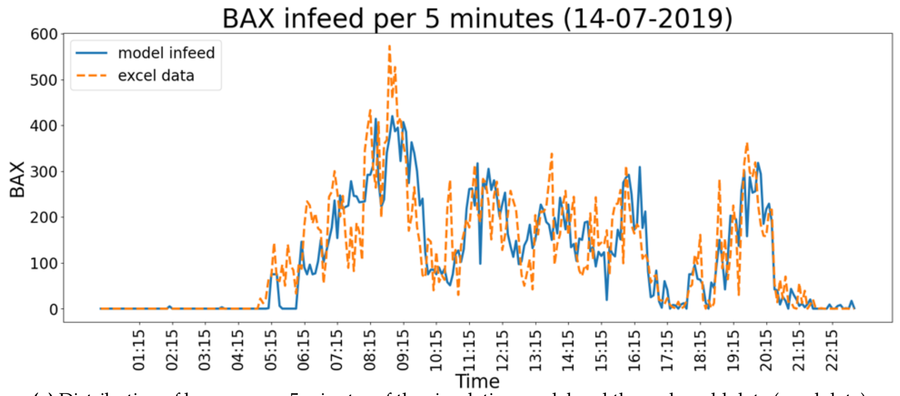


n	mean	std	CI	CI/mean
2	8608,50	208,50	288,96	3,36%
3	8654,67	212,83	240,83	2,78%
4	8653,25	184,33	180,64	2,09%
5	8663,20	166,07	145,56	1,68%
6	8628,17	170,64	136,54	1,58%
7	8674,43	194,42	144,03	1,66%
8	8698,38	192,59	133,45	1,53%
9	8688,89	183,54	119,91	1,38%
10	8689,70	174,14	107,93	1,24%
11	8684,36	166,89	98,63	1,14%
12	8658,00	182,15	103,06	1,19%
13	8645,38	180,37	98,05	1,13%
14	8655,93	177,92	93,20	1,08%
15	8657,13	171,95	87,02	1,01%

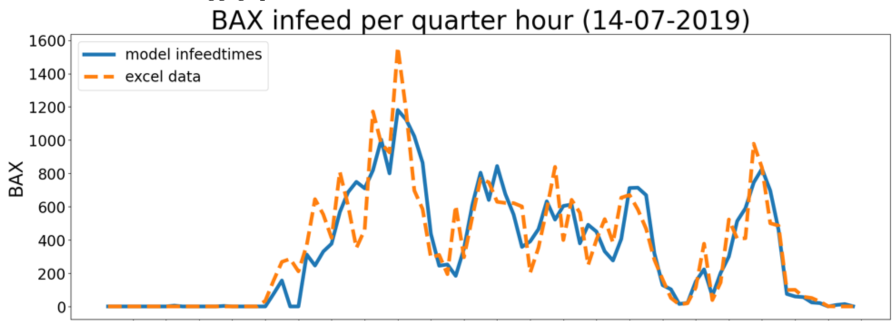
Figure 6.5: Testing sensitivity to randomness with several replications

6.8. Validation of the model

The model was run with all the available input data. This was the week of data from the 14th until the 21th of July 2019 and showed infigure 4.6. The model simulated the new infeed times of all the bags that arrived by airplane without cold buffering in order to be able to validate the model. A graph is made of the distribution over a day of the amount of bags that are fed in to the system per quarter hour and per five minutes. In figure 6.6 the two graphs are shown for the baggage infeed on the 14th of July 2019. The blue line shows the output of the simulation model and the orange dotted line is the real world data for that day. The simulation model output has peak moments on the same times as the real world data but the morning peak at 9:00 is not as high. This happens for both the fifteen and five minutes interval plots.



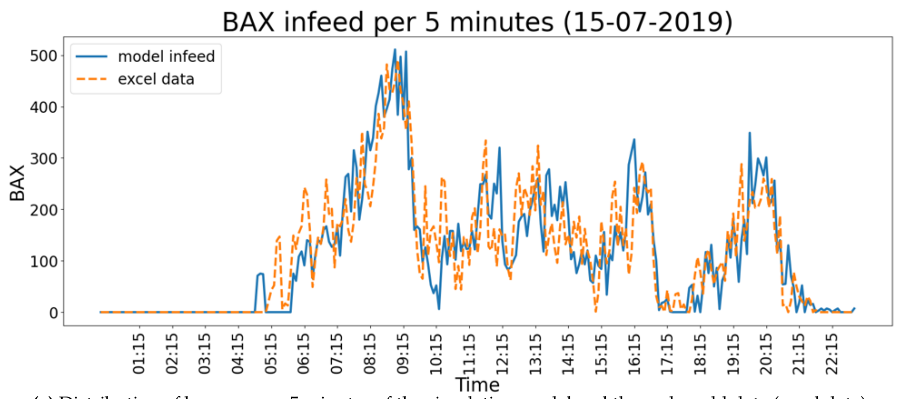
(a) Distribution of baggage per 5 minutes of the simulation model and the real world data (excel data)



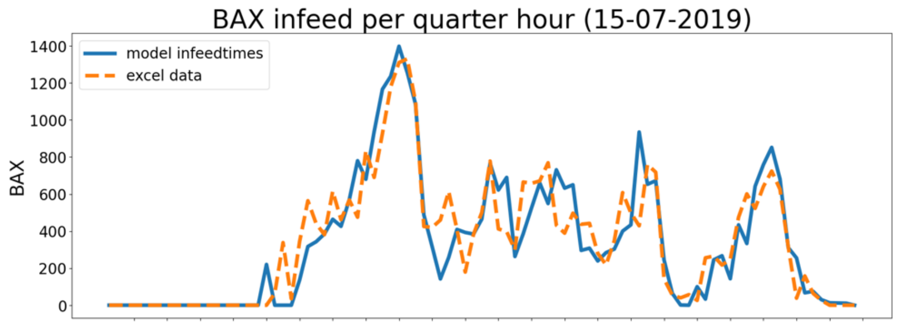
(b) Distribution of baggage per quarter hour of the simulation model and the real world data (excel data)

Figure 6.6: Baggage infeed on 14-07-2019

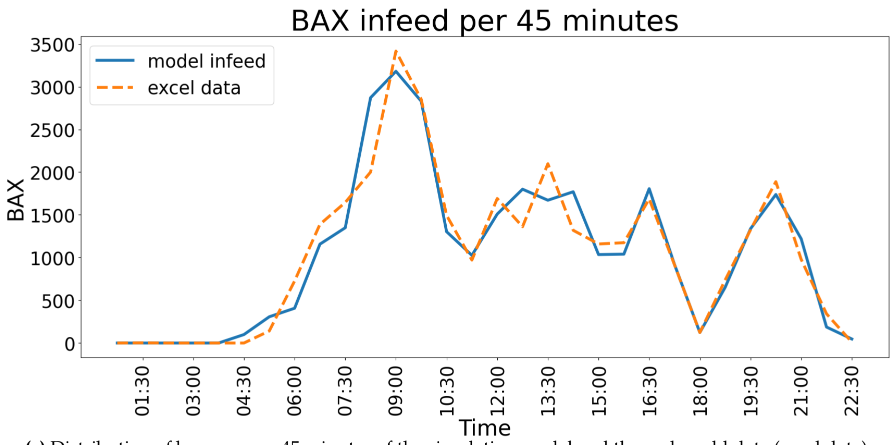
In figure 6.7 the two graphs are shown for the baggage infeed on the 15th of July 2019. The trend during the day seems similar again between the model output and the real world data. The peaks are also of similar height, especially compared to figure 6.6. For the 15th of July a figure is also made made showing a distribution where the buckets are 45 minutes instead of 5 or 15. This is shown in figure 6.7c. What can be seen here is that the two peaks at 12:00 and 14:00 are showing up like one peak. Increasing the bucket size has as a downside that the definition disappears. The decrease of bucket size to five minutes increases the definition but adds too much fluctuation to the graph.



(a) Distribution of baggage per 5 minutes of the simulation model and the real world data (excel data)



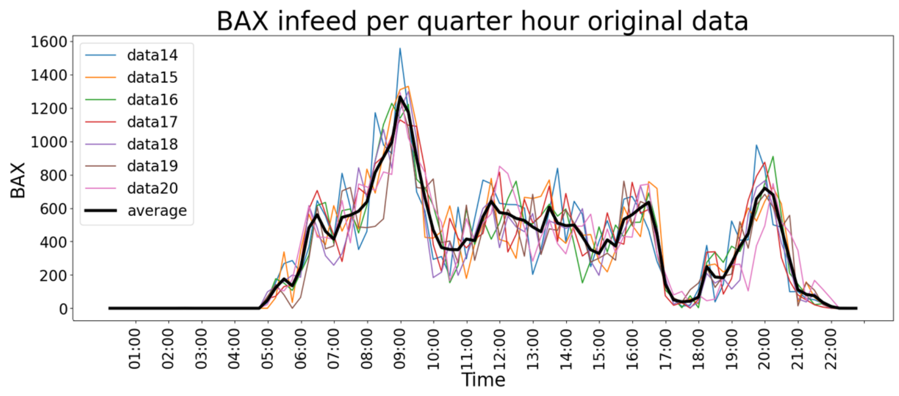
(b) Distribution of baggage per quarter hour of the simulation model and the real world data (excel data)



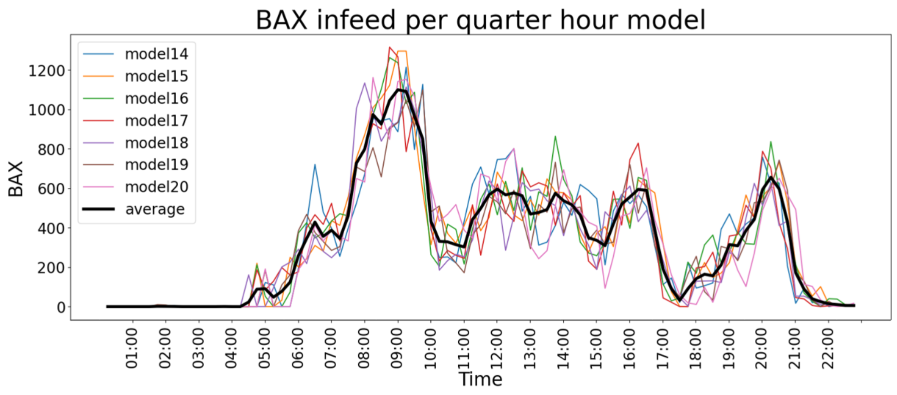
(c) Distribution of baggage per 45 minutes of the simulation model and the real world data (excel data)

Figure 6.7: Baggage infeed on 15-07-2019

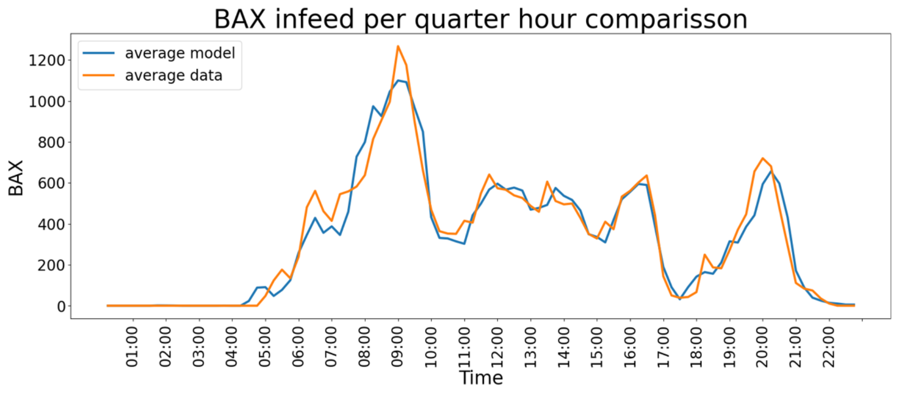
In figure 6.8 the results of all the days is shown. In figure 6.8a is the real world data plotted. The average of all 7 days is shown with the thicker black line. In figure 6.8b is the simulation model output plotted. The average of all 7 days is shown with the thicker black line. These two averages are also shown against each other in figure 6.8c. Here it can be seen that the visual difference has become smaller than in the previous plots of 14 and 15 July.



(a) Distribution of baggage per quarter hour of the real world data



(b) Distribution of baggage per quarter hour of the simulation model



(c) Averages of the simulation model and real world data distribution

Figure 6.8: Comparison of simulation result with real world data

The results of the model is also quantified. This is done with three different calculations. The first one was to check the absolute difference between the model output and the data per time interval. This is done at each timepoint and the average over that is shown in table 6.3. The second calculation was to look at the relative difference by dividing the calculated difference of the first step by the actual amount of bags from the real world data. The third calculation was to look at difference between the output of the model and the real world data but divided by the average number of bags that are fed into the system. All the difference are calculated for each interval and the total average of that is taken. The results are shown in table 6.3.

	Absolute difference	Relative Absolute difference	Relative mean absolute difference
14-07-2019 (15 minutes)	102	27.9%	22.2%
15-07-2019 (5 minutes)	42	75.5%	26.3%
15-07-2019 (15 minutes)	96	34.4%	20.5%
15-07-2019 (45 minutes)	168	31.8%	13.4%
Whole week (15 minutes)	49	18.7%	11.5%

Table 6.3: Performance output of the model

Based on the results from the calculations and the visual output of the model a choice must be made for a time interval to process the further simulation results. Quantitatively the best result was shown in data for the whole week with a 15 minute interval. This graph also shows all the 5 peaks along the day very clearly and at the the same timestamp as in the real world data. This is a problem with the 45 minutes interval, the peaks are less clear and the absolute differences get bigger. With a 45 minute time interval it will become harder to know if a bag has arrived to the BHS on time because the transfer time could be less than 45 minutes. The 5 minute interval has the worst quantitative and visual results. For these reasons it is chosen to continue with the 15 minute interval as a standard display and measurement for the peaks.

6.9. Conclusion

The goal of the simulation is to simulate a process from the moment an airplane land until the moment the transfer bags are fed in to to the system. The different sub processes were explained in this chapter such as waiting time for an airplane, unloading time, driving time and feed in times. In order to simulate the whole process different elements are needed. These elements are the airplane generator, the airplane, carts, containers, tugs, buffers and infeed stations. Of all these elements there attributes and PDLs were given in order to be able to program this in a simulation model. After that a first model was build without a buffer in order to validate and verify this model and the chosen parameters with real world data. During this process the model was verified using different experiments and the effect of changing the random seed was tested. It was concluded that at least 13 replications should be done to minimize the effect of the changing random seed. During the validation process it was concluded that the outcome of the simulation should be looked at using 15 minutes intervals to have a good balance between fluctuation and definition in the figures.

7 Experimental set-up

In this chapter the configurations and scenarios that will be simulated with the simulation model are discussed. A configuration is a combination of design options. The design option were first shown in chapter 4 in table 4.2. There is a baseline configuration and there were three configurations chosen. Here these choices are further elaborated. A scenario is a change in the input of the simulation model. It is a possible prediction of the future that should be tested to see if an experiment will have the same impact.

7.1. Configurations

In the conclusion of chapter 4, the different design options were discussed. Along with experts from Schiphol Airport three configurations were chosen to simulate in the model. These configurations are:

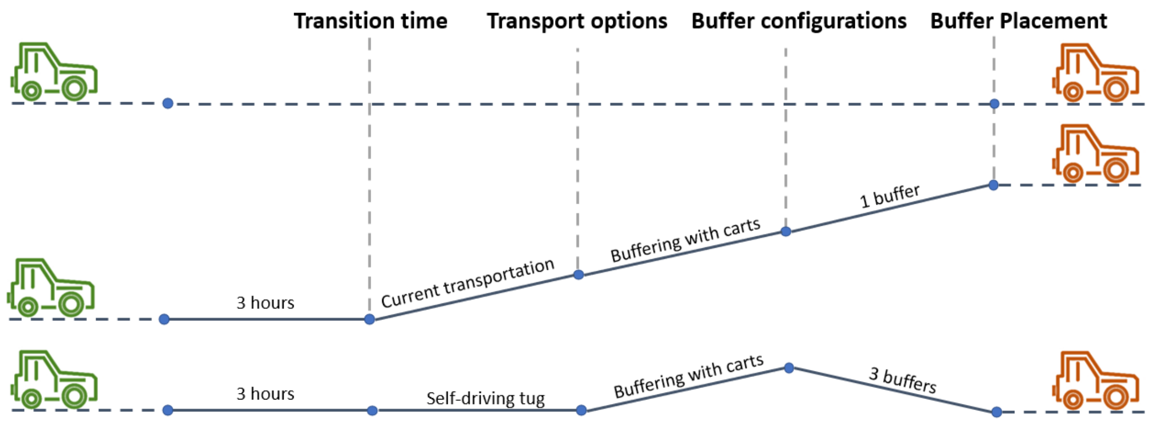


Figure 7.1: experiment configurations

In all configurations there are a minimal number of cold bags a flight must contain in order to seperate baggage for this flight. Otherwise to much work must be done with a minimal result. The minimum number of bags differs for bulk baggage and containerized baggage. The minimum for bulk baggage is 5 bags. The minimum for containerized baggage is 25 bags. The reason this is higher is due to the fact that airlines want to fill up containers as much as possible. If there are less than 25 bags on board the container will also be filled with hot baggage and then the container cannot be buffered anymore without fully unloading the container.

7.1.1. Configuration 0: No buffer

In order to see the effect of cold buffering the simulation must also be ran without a buffer in place. That is configuration 0.

7.1.2. Configuration 1: 1 Buffer with normal transport

The chosen design option combination for configuration 1 is visualized in figure 7.1. The transition time is set at 3 hours. If the time between scheduled arrival and scheduled departure is longer than 3 hours, a bag will be classified as cold. The decision to put this at 3 hours was made on the basis that on average it takes bags around one hour to get in to the BHS and go through it and that some bags should be buffered for two hours do be fed in after the peak. If the transition time was set at 2 hours there would be a higher chance that bags would be placed in the buffer for too long and miss their connecting flight. With a transition time at 4 hours less bags would be classified as cold and this would lower the potential amount of peak shaving. This configuration is meant to represent the trial KLM ran. They ran the trial using extra tugs and carts for transportation of cold baggage so that way of transportation was chosen for this configuration. In the case of current transportation the driving times times in the model can remain normal. During the trial all the colds carts and containers were parked at an one location. So one buffer location is chosen. In this experiment the buffer location will be placed at a further location due to the limited available space at the infeed locations

7.1.3. Configuration 2: Autonomous tugs

The chosen design option combination for configuration 2 is visualized in figure 7.1. The goal of this configuration is to see what will be the on the infeed points while using autonomous tugs. Personnel shortage at airports is a global problem (Pole, 2022). One way to keep the operation running is automation. In this experiment it chosen to use a transition time of 3 hours for the same reason as in experiment 1. The transportation method is self driving tugs. The buffer configuration will be to parts the carts and containers. At Schiphol airport the BHS can bring transfer bags from the infeed areas to the make-up areas in the same location but also to make-up areas at other locations. This is done through the 'backbone'. It is more convenient if transfer bags are fed in to the system at the same location as their make-up area is. Otherwise more bags need to go through the backbone. For that reason it was chosen to use 3 buffer locations in this experiment to buffer the bags just before their correct infeed locations.

7.2. Scenarios

To see if the chosen configurations will be future proof three possible future scenarios are chosen after talks with personnel from Schiphol.

7.2.1. Scenario 0: Baseline scenario

In order to see the change which a future scenario brings the baseline must first be shown. For that reason scenario 0 exists which is the current data without any changes.

7.2.2. Scenario 1: Increase in flights

The different experiments all have different transportation methods. The number of carts and tugs needed to bring all the bags form the airplanes to the infeed stations is an important measurement for the on time performance of the system. The number of flight movements is expected to increase at Schiphol airport. The expected growth is 15% and this will most likely arrive during the morning peak. The simulation model must find out how much tugs are needed to transport all these bags and how much capacity of the infeed stations is needed with and without buffering. To increase the incoming flights by 15% in the morning peak extra flights must be added to the dataset. In the dataset that is currently being used of 15 July, 94 airplanes land between 7:00 and 9:00 o'clock. An increase of 15% would mean that 14 extra airplanes must land in this period. These 14 extra flights will be taken from the data set of the previous day to ensure that a realistic number of bags are in the airplane and that their transfer times are realistic.

7.2.3. Scenario 2: Increase of bags at individual infeed points

Schiphol airport has three main transfer baggage infeed points, South hall, D-hall and E-hall. The D-hall machinery will no longer be supported for maintenance from 2028 on wards. This might cause it to shut down. What will happen to the usage of those two infeed hall capacities if all the bags must be rerouted to these other two infeed halls. On the 15th of July, 416 airplanes land with transfer baggage in them. 228 flights are fed in at the E-hall, 134 flights are fed in at the D-hall and 54 flights are fed in at the South-hall. These 134 flights that were supposed to go to the D-hall must be divided between E and South. Four times more bags are now going to the E-hall compared to the South-hall. To keep this ratio the same 108 flights of the 134 should be fed in at E and 26 flights should be fed in at South. This change must be made to the input data.

7.2.4. Scenario 3: Bulk baggage gets containerized

The third future scenario is that more flights will start to bring in the baggage in containers compared to bulk baggage. KLM has announced that their cityhoppers, which are Boeing 737 will be replaced with Airbus A320. The airbus A320 will use containers to transport their baggage. To change this in the input data the flights that are currently Boeing 737 will be labeled as containerized flights.

7.3. Conclusion

The three experiments should be simulated in the three scenarios. This will result in 9 combinations. In this 9 results the peak reduction will be calculated which is the goal of cold buffering but it must be checked how many resources this needs in order to perform well. The results can be filled in table 7.1.

	Configuration 0: No buffer	**Configuration 1: Normal tugs 1 buffer**	**Configuration 2: Autonomous tugs 3 buffers**
Scenario 0: Normal arrival	Experiment 1	Experiment 2	Experiment 3
Scenario 1: More flights in peak hours	Experiment 4	Experiment 5	Experiment 6
Scenario 2: Less infeed capacity	Experiment 7	Experiment 8	Experiment 9
Scenario 3: More containers	Experiment 10	Experiment 11	Experiment 12

Table 7.1: Overview of the experiments

8 Results

In this chapter the outcomes of the simulation experiments will be presented. Per experiment it will be shown what the effect is oh changing the input data for each scenario and the effect of each chosen configuration will be shown as well.

8.1. Configuration 0: No buffering

In order to see what the effect of implementing a buffer will be the situation must be simulated without one present. This was defined as configuration 0. Within configuration 0 there a 4 scenarios that change the input on the system. Scenario 0 is the normal input data. Experiment 1 was defined as the combination of configuration 0 and scenario 0. The result can be seen in figure 8.1

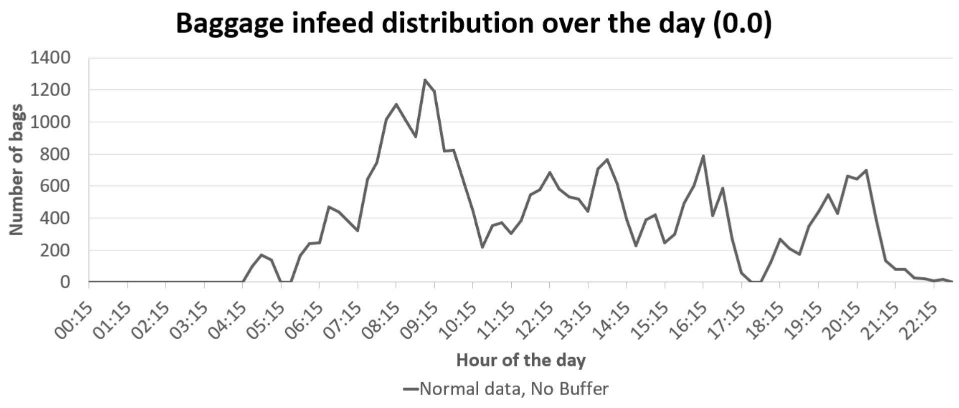


Figure 8.1: Result experiment 1

This is the same result as in chapter 7 from the sensitivity analysis and data validation. The peak here lies at 1261. This is 55.4% of the total infeed capacity. In chapter 6 the process of the model was explained. In short it says that there is a sequence of queues a bag goes through in order to go from the apron to the infeed station. Another output from the model is how many tugs it takes process all the baggage. In total 12 tug queues were in use.
In scenario 1 there is an increase of incoming flights during the morning peak of 15%. This was simulated in experiment 4, this result is shown in figure 8.2.

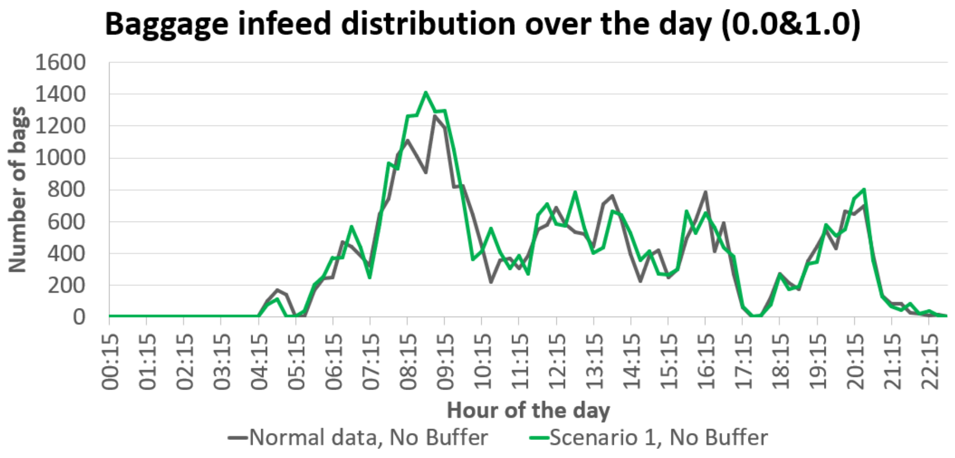


Figure 8.2: Result experiment 4

In figure 8.2 the result of experiment 4 is shown compared to original situation of experiment 1. Because of the increase of incoming flights the morning peak has also increased. The peak of experiment 4 is 1409 bags fed in a quarter hour. This is 61.9% of the total capacity. In order to process all the bags 13 tug queues were in use.
The second scenario was removal of an infeed location and thereby increasing the the pressure on the remaining infeed locations. In the input data the bags that were supposed to go the infeed location D were rerouted to either E or South. Experiment 7 was testing this scenario without a buffer, the result is shown in figure 8.3.

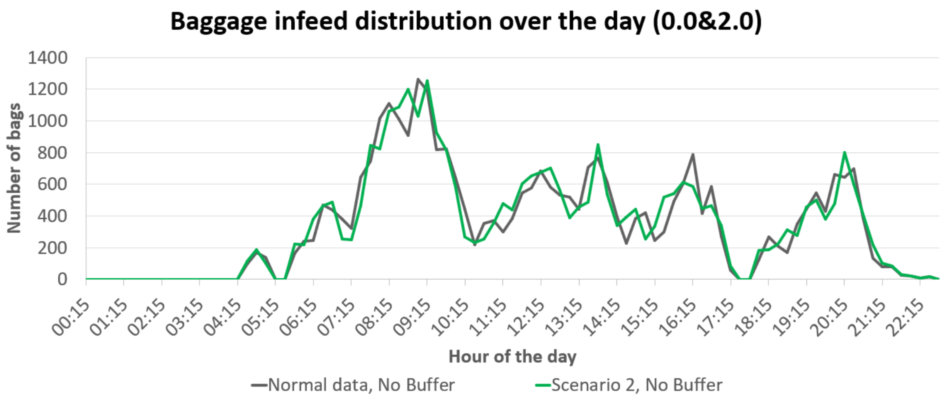


Figure 8.3: Result experiment 7

The result of experiment 1 is also shown in figure 8.3 to compare the two. The amount of bags that arrive and that are fed in remain the same in the two experiments and the graphs look similar. The peak of experiment 7 is 1254 bags per quarter hour. The infeed capacity has dropped from 2275 bags per quarter hour to 1400 bags. This means that 89.6% of the total capacity is being used. 11 tug queues were used to process all the bags. In figure 8.4 the infeed distributions of infeed location South and E-hall are shown. There peaks lay at 350 and 1009 respectively. The peak at the E-hall is above the maximum of 900 bags per quarter hour. The maximum of the South infeed hall is 500 bags per quarter hour.

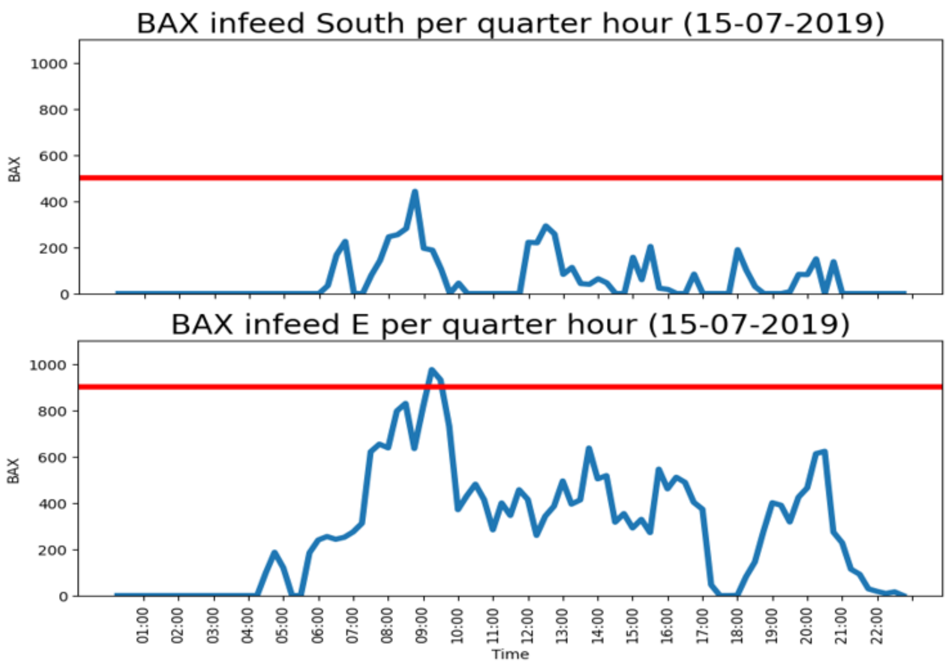


Figure 8.4: Result experiment 7 per infeed location

In scenario 3 the same amount of flights and baggage arrives but relatively more flights contain containerized baggage than bulk baggage. Experiment 10 is the combination of scenario 3 and configuration 0. The result is shown in figure 8.5. Experiment 0 is also shown in figure 8.5 to compare the differences. Those are minimal. The peak of experiment 10 is 1216 bags per quarter hour. Which is 54.5% of the total infeed capacity. 12 tug queues were used to process all the bags.

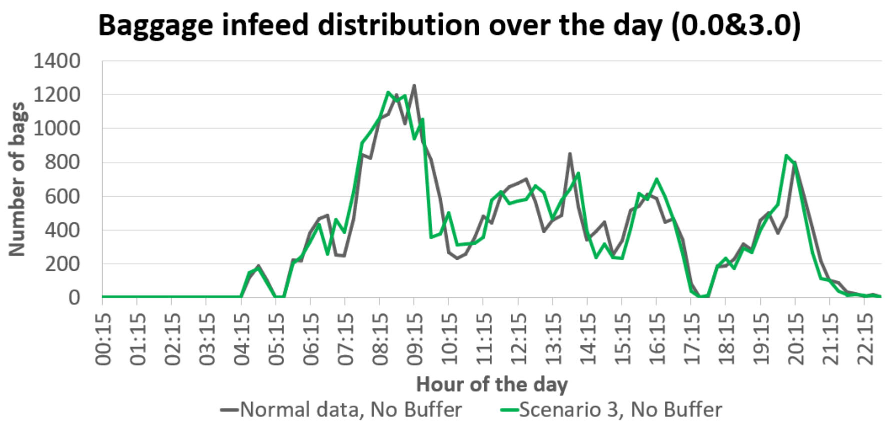


Figure 8.5: Result experiment 10

8.2. Configuration 1: 1 Buffer with normal transport

In this section the results of the experiments with configuration 1 will be given. In configuration 1 it was chosen to use normal transportation to one buffer location. At this buffer location the carts and containers are parked for two hours and then brought to the South infeed location were the cold bags are fed in to the system. The first experiment was experiment 2. This is a combination of scenario 0 and configuration 1. The results are shown in figure 8.6.

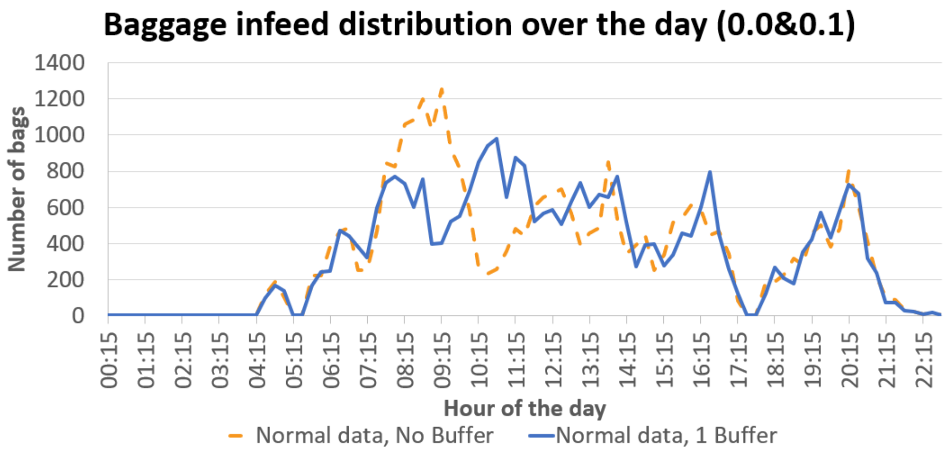


Figure 8.6: Result experiment 2

In figure 8.6 experiment 1 and 2 are shown, both are in the same scenario so with the same input data. The morning peak lowered by taking out the cold baggage and buffering it for two hours. After that the bags were transported to the South infeed location, feeding the bags back in here did create a new peak. The peak is set 978 bags, which were fed in between 10:30 and 10:45AM. In this experiment 43.0% of the total capacity is used. The peak reduction is calculated by subtracting the the number of bags in the old peaks from the numbers of bags in the new peak and this number is than divided by the number of bags in the old peak. In this case the outcome is 22.4% peak reduction. In this experiment the amount of used tug queues and old tug queues were also counted to get a general indication of the required changes to implement a buffer. There were 10 tug queues en 29 cold tug queues in order to process all the baggage.

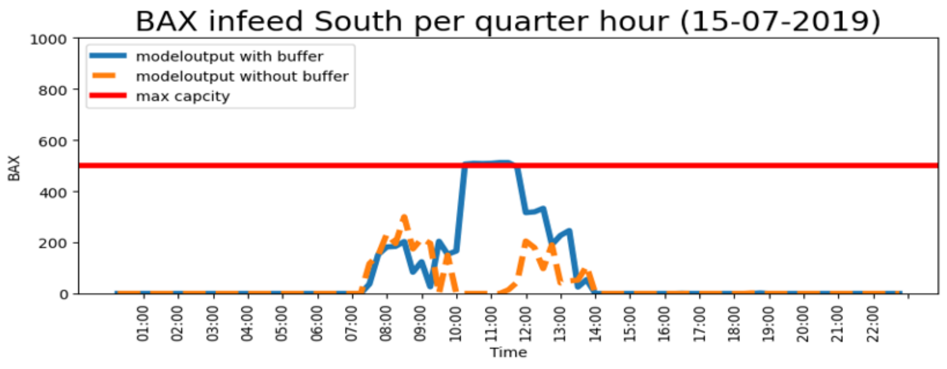


Figure 8.7: Result experiment 2 individual infeed location

Because all the cold baggage is fed in at the South infeed location it is important to see what happens to infeed capacity there. This is shown in figure 8.7. In figure 8.7 the orange dotted line is the infeed distribution of the model without the use of a buffer. The blue line is the distribution of the model with the usage of a buffer. The red line is the capacity of the infeed station. The maximum capacity is reached in this situation so bags have to wait longer at the infeed station queue before being able to fed in to the system.

The next experiment that was ran was experiment 5. This experiment is the combination of scenario 1 with configuration 1, so more incoming flights during the peak hours and the usage of one buffer. The result of experiment 5 is plotted alongside experiment 4 in figure 8.8.

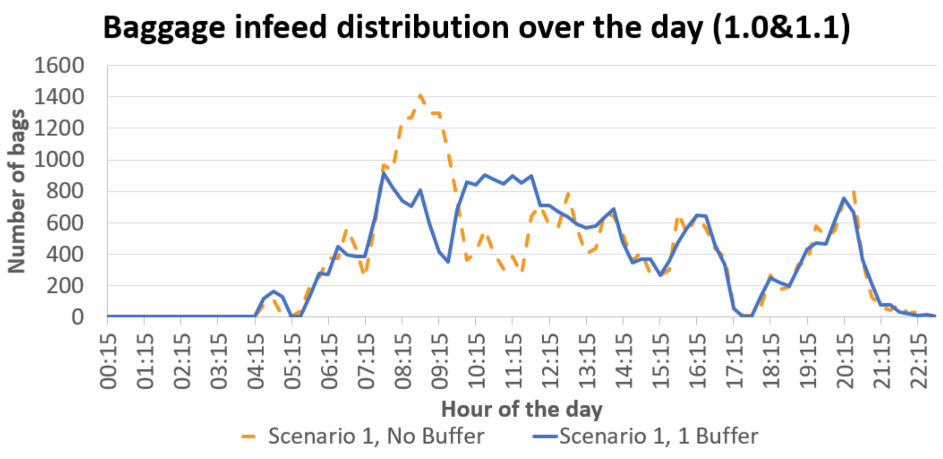


Figure 8.8: Result experiment 5

The peak of experiment 5 is 917 bags per quarter hour. This is 40.3% of the total available capacity. In experiment 4, maximum usage of the total system capacity was 61.9%. The peak reduction was 34.9%. There were 12 tug queues to process all the hot baggage and 32 cold tugs to transport cold baggage to the buffer location. Because this buffer configuration only makes use of one buffer at one infeed location it is important to show what happens there, this is shown figure 8.9.

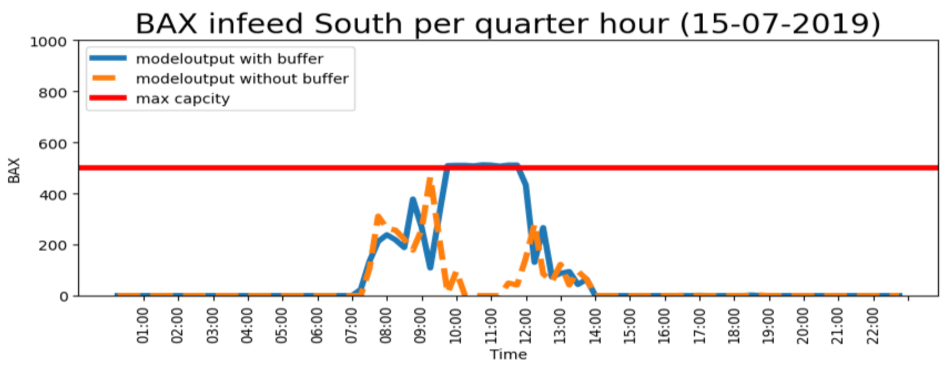


Figure 8.9: Result experiment 5 individual infeed location

The full capacity of the south infeed hall is needed to feed in all the cold baggage after the buffering period. In scenario 1 more airplanes arrived during the morning peak which also increased the number of cold bags that could be buffered. In scenario 0 3789 cold bags arrived during the morning peak, in scenario 1 there 4581 cold bags. The increase in the number of bags can also be seen in the width of the buffer peak in figure 8.9. The South infeed hall is working until 12:00 AM at full capacity.

The next experiment that was ran was experiment 8. In experiment 8 scenario 2 was combined with configuration 1. Scenario 2 was the future possibility where infeed hall D would not be available and all the bags are rerouted to the E and South infeed hall. The result of this experiment can be seen in in figure 8.10. The peak of experiment is 839 bags per quarter hour. This is a new peak that occurs during the buffering period which replaces the morning peak. With a peak at 839 bags per quarter hour 60.0% of the total capacity is in use at that moment. Without buffering in scenario 2 90% of the capacity was in use at the peak moment of operation. The peak reduction was 33.1%. There were 10 tug queues to process all thee hot baggage and 29 cold tugs to transport cold baggage to the buffer location. Because the D-hall is not in use and the South hall is used as the buffer infeed location the infeed distributions of the South and E-hall are also shown in figure 8.11. What can be observed in these individual infeed location graph is that the the cold bags in the morning peak that were causing problems at the E-hall are fed in during the buffering period in South. This results in that all the bags can be fed in to the system on time while one infeed location is out of order. Without buffering this would not be possible.

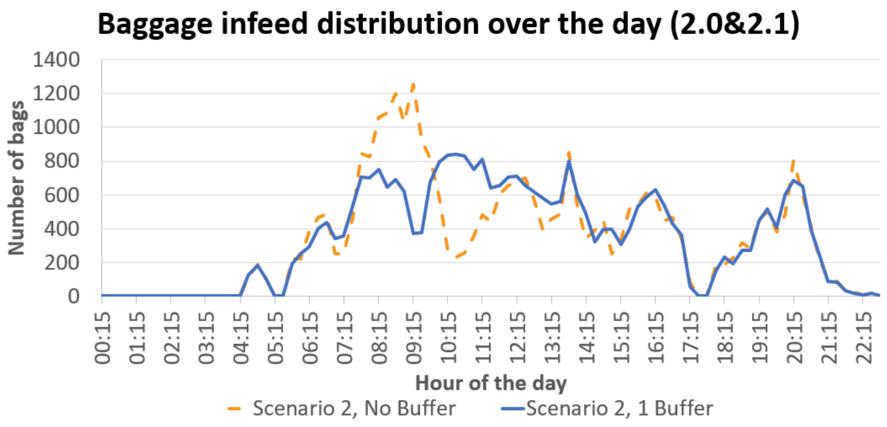


Figure 8.10: Result experiment 8

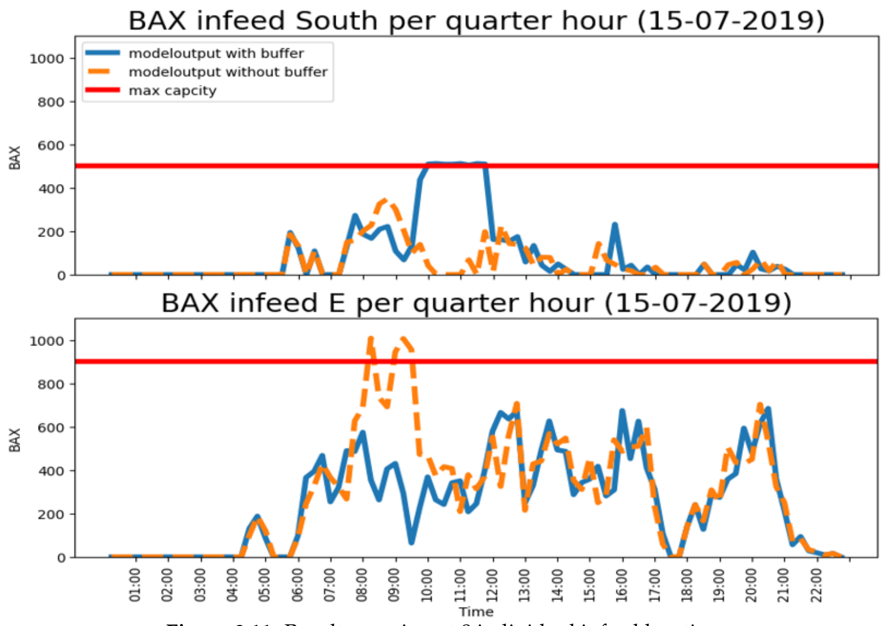


Figure 8.11: Result experiment 8 individual infeed location

The next experiment is the final one with configuration 1. It is experiment 11. This experiment is the combination of configuration 1 with scenario 3. In scenario 3 the amount of flights stay the same but more flights are transporting containerized baggage than in scenario 0. The result of experiment 11 is shown in figure 8.12. The new peak is 873 bags per quarter hour at 11:00 AM. At this peak the infeed system uses 38.4% of the total capacity. Without buffering in scenario 3 53.5% of the capacity was in use at the peak moment of operation. The peak reduction was 28.2%. There were 9 tug queues to process all thee hot baggage and 28 cold tugs to transport cold baggage to the buffer location.

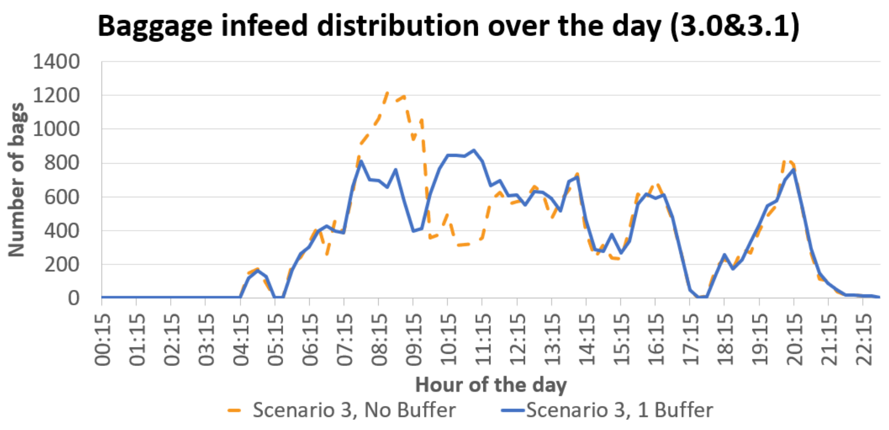


Figure 8.12: Result experiment 11

The South station infeed distribution is shown in figure 8.13. This shows that less bags are fed in during the buffering period in experiment 11 compared to experiment 8 and 5 because of the smaller width in the buffering period. This is in concurrence with the amount of cold bags that were predetermined to go to the South infeed hall in these three different scenarios.

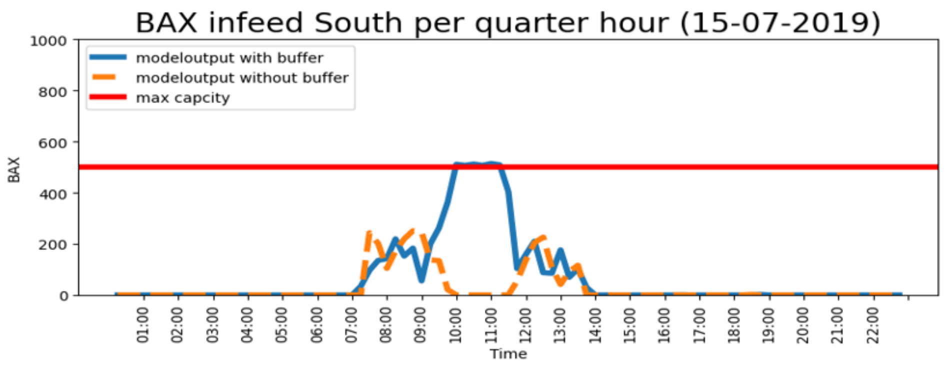


Figure 8.13: Result experiment 11 individual infeed location

8.3. Configuration 2: 3 buffers with autonomous tugs

In this section the results of the experiments are shown which used used configuration 2 as buffer configuration. Configuration 2 was the use of autonomous tugs and 3 separate smaller buffer location close to the infeed points. In this way the baggage that was meant to go to for instance the E-hall will be buffered in the E-buffer and then fed in to the E infeed stations. The first experiment with configuration 2 is experiment 3. This experiment combines the baseline scenario 0 with usage of buffer configuration 2. The result of this experiment is shown in figure 8.14. The maximum peak here is 1032 at 10:45AM. At this moment the system uses 45.4% of the total system capacity. The peak reduction was 18.2%. In order to process all the baggage in the simulation 10 tugs were used for the hot baggage and 30 cold tugs were used for the cold baggage.

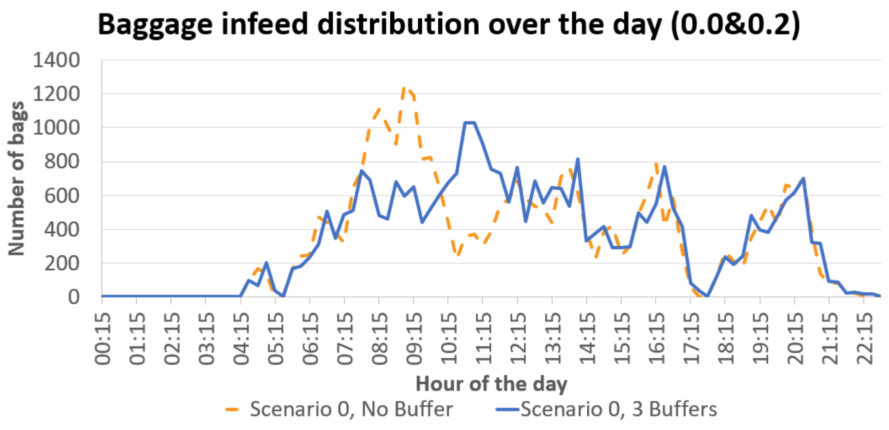


Figure 8.14: Result experiment 3

Because all three infeed locations are in use the individual infeed distributions are plotted and shown in figure 8.15. The individual infeed point infeed distributions look a lot different than those from the experiments with only 1 buffer from configuration 1. None of the infeed points reach there maximum capacity. Most cold bags were already intended on going to the E-hall to be fed in to the system. So the peak is higher there compared to the South and D-hall infeed figures.

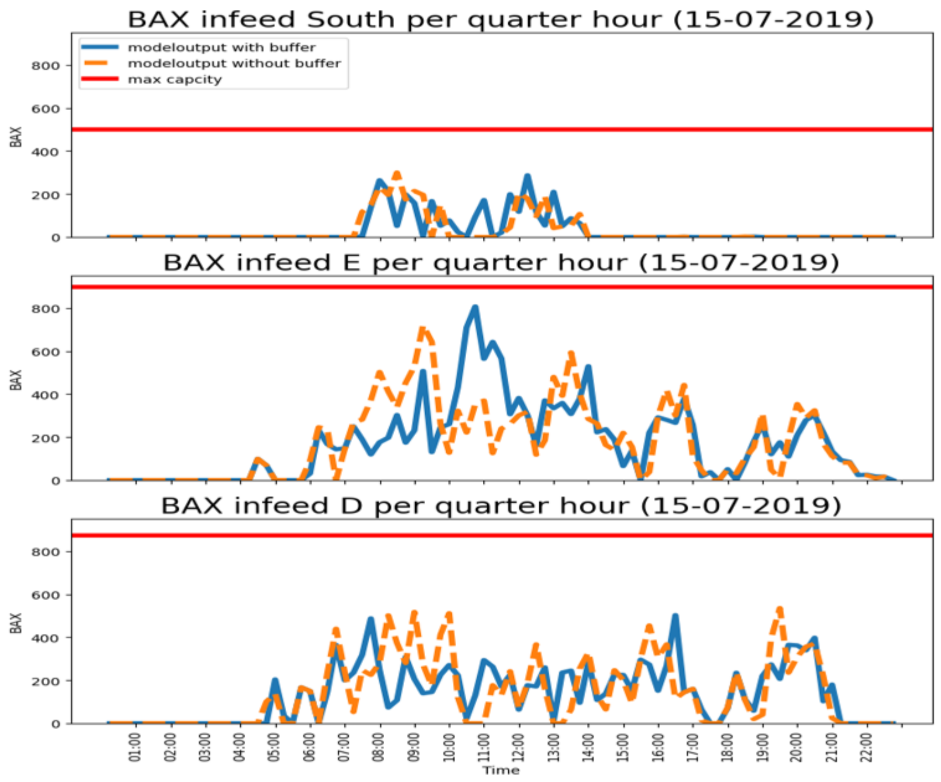


Figure 8.15: Result experiment 3 individual infeed location

The next experiment is experiment 6. This is a combination of configuration 2 with scenario 1. The result of this experiment is shown in figure 8.16. The peak of this experiment is 963 bags per quarter hour at 10:45AM. At this moment the system uses 42.3% of the total system capacity. The peak reduction was 31.6%. The individual infeed distributions are similar to those of experiment 3 shown in figure 8.15. In order to process all the baggage in the simulation 12 tugs were used for the hot baggage and 34 cold tugs were used for the cold baggage.

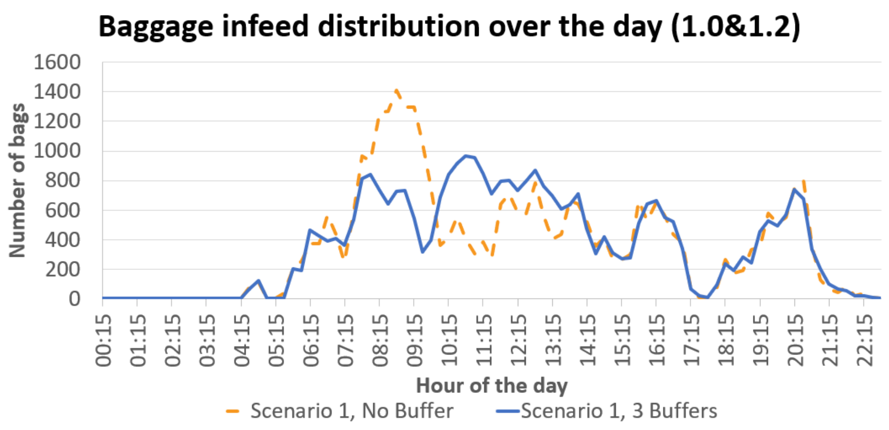


Figure 8.16: Result experiment 6

The next experiment is experiment 9. This is a combination of configuration 2 with scenario 2. Technically this configuration has 3 buffer location but in this scenario only 2 infeed points are used so only 2 buffer locations were used as well. The peak of this experiment is 895 bags per quarter hour at 11:00AM. At this moment the system uses 63.9% of the total system capacity. The peak reduction was 28.6%. There were 10 tug queues to process all thee hot baggage and 30 cold tugs to transport cold baggage to the buffer location. The result of this experiment is shown in figure 8.17.

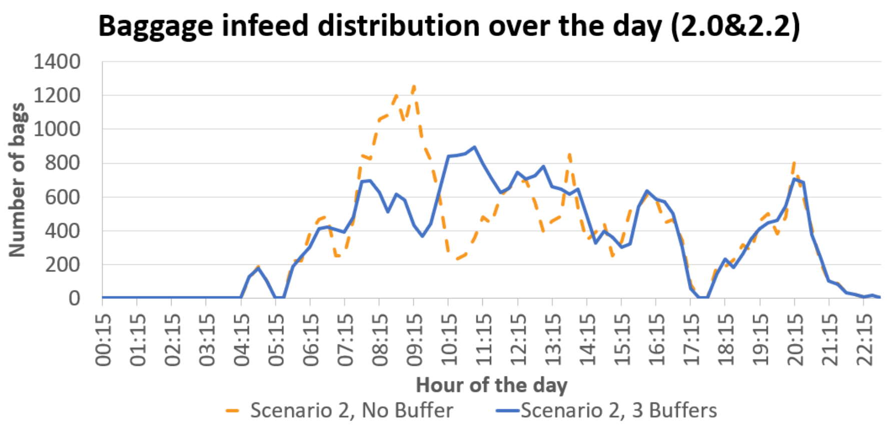


Figure 8.17: Result experiment 9

For experiment 9 the infeed distribution at the South hall and E-hall were individually plotted. This can be seen in figure 8.18. Because most of the baggage that was supposed to go to the D-hall went to the E-hall the E-hall maximum infeed capacity is reached. The South hall is able to process the increase of baggage with out using maximum capacity. The infeed in this graph even goes above to maximum limit. This can happen happen in this model because the maximum capacity is based on a specified number of bags in one hour, the data point before and after the peaks compensate. In the real world this would cause a longer waiting time in the queue before the infeed point but in the model they are released too soon.

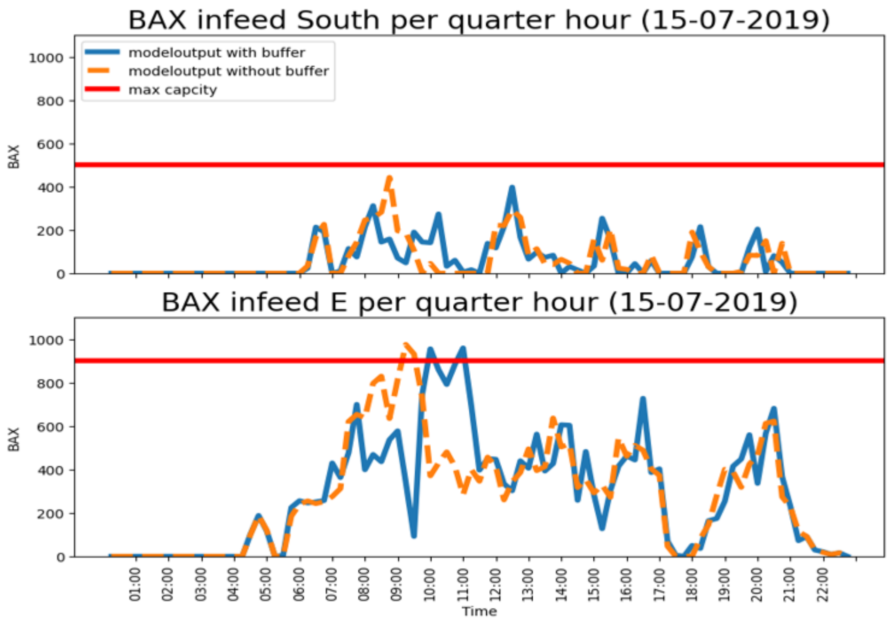


Figure 8.18: Result experiment 9 individual infeed location

The last experiment is experiment 12. This is a combination of configuration 2 with scenario 3. The result of experiment 12 is shown in figure 8.19. The peak of this experiment is 848 bags per quarter hour at 11:00AM. At this moment the system uses 37.3% of the total system capacity. The peak reduction was 30.2%. There were 9 tug queues to process all thee hot baggage and 29 cold tugs to transport cold baggage to the buffer location. The individual distributions were similar to those of experiment 3 in figure 8.15.

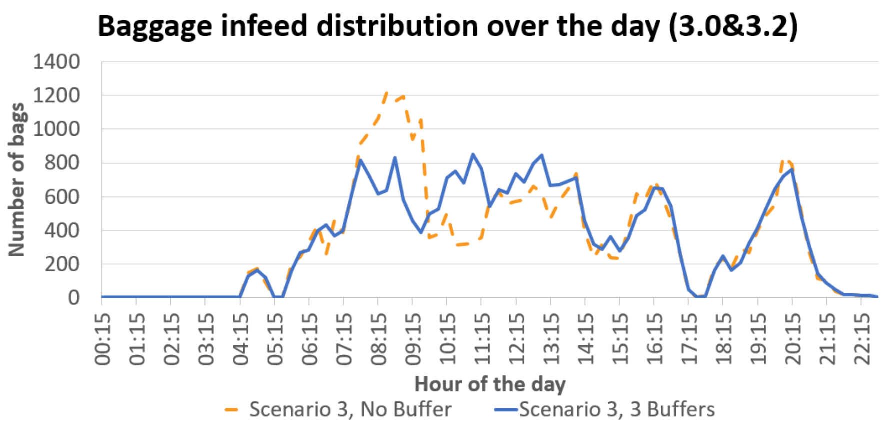


Figure 8.19: Result experiment 12

8.4. Conclusion

All the experiments were ran successfully. The results will be further discussed in chapter 10. An overview of the numerical results can be found in table 8.1. The highest peak that occurs in each experiment is shown and the number of tug queues is given for each experiment. Based on the peaks the peak shaving can be found. The peaks of each experiment are visualised in figure 8.20. The peak shaving per scenario is shown between no buffering, buffering with configuration1 and with configuration 2. The peak shaving percentages are also displayed in the figure.

	Configuration 0: **No buffer**	**Configuration 1:** **Normal tugs 1 buffer**	**Configuration 2:** **Autonomous tugs 3 buffers**
Scenario 0: **Normal arrival**	Peak: 1261 Tugs: 12 Cold Tugs: 0	Peak: 978 Tugs:10 Cold Tugs: 29	Peak: 1032 Tugs: 10 Cold Tugs: 30
Scenario 1: **More flights in peak hours**	Peak: 1409 Tugs: 13 Cold Tugs: 0	Peak: 917 Tugs: 12 Cold Tugs: 32	Peak: 963 Tugs: 12 Cold Tugs: 34
Scenario 2: **Less infeed capacity**	Peak: 1254 Tugs: 12 Cold Tugs: 0	Peak: 839 Tugs: 10 Cold Tugs: 29	Peak: 895 Tugs: 10 Cold Tugs: 30
Scenario 3: **More containers**	Peak: 1216 Tugs: 12 Cold Tugs: 0	Peak: 873 Tugs: 9 Cold Tugs: 28	Peak: 848 Tugs: 9 Cold Tugs: 29

Table 8.1: Effect of each experiment on every scenario

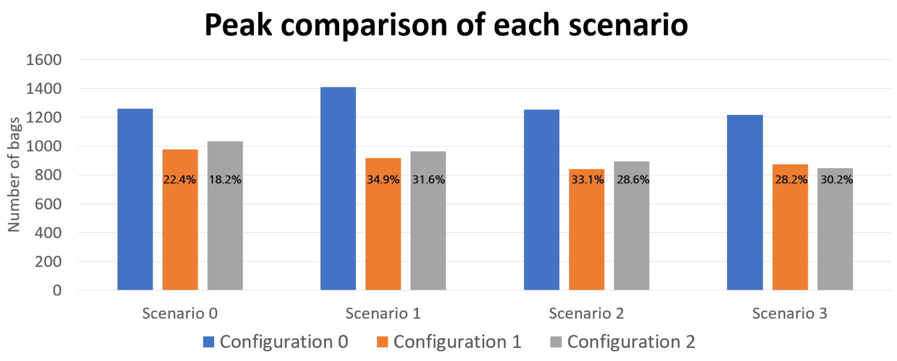


Figure 8.20: Overview of the peaks of each configuration per scenario compared to each other

9 Implementation at Schiphol airport

The last research question that was stated in the introduction was what would be the most desired implementation for Schiphol airport. This question will be answered in this section of the report based on the simulation outcomes from chapter 8 and the implementation costs. Those costs are shown below.

9.1. Implementation strategies

The simulation showed the results for the configurations where all the cold bags were taken to a buffer. Schiphol airport might not require to buffer all the cold bags but just a part of it in order to reach the desired peak shaving. In the simulation the data that is looked at is from the 15th of July 2019. On this day between 7:00 and 9:00 AM 16 airplanes arrive with containerized baggage, which are carrying more than 25 cold transfer bags each. 68 airplanes arrive with bulk baggage, which are carrying more than 5 cold transfer bags each. The 16 containerized airplanes have 2492 cold bags in total and the 69 bulk baggage airplanes carry 1825 cold bags. The reasons that the containerized flights are lower in number but carry more cold transfer baggage is due to the fact that they can carry more baggage. They are mostly used for intercontinental flights, which carries more transfer baggage and containerized flights arrive earlier compared to must bulk baggage flights so the transfer times are also higher. Based on this five implementations strategies are made. The first one is to only buffer cold baggage from containerized baggage. The second strategy is to buffer all the baggage from the containerized airplanes and the top 25% of the bulk baggage airplanes. The top being the 25% that carries the most cold bags. The third strategy is to buffer all the baggage from the containerized airplanes and the top 50% of the bulk baggage airplanes. The fourth strategy is to buffer all the baggage from the containerized airplanes and the top 75% of the bulk baggage airplanes. The fifth strategy is to buffer all the baggage from the containerized airplanes and all baggage of the bulk baggage airplanes. An overview of the implementation strategies is shown in table 9.1.

	Implementation	**Cold bags**
Strategy 1	All containers	2494
Strategy 2	All containers, 25% Bulk	3322
Strategy 3	All containers, 50% Bulk	3835
Strategy 4	All containers, 75% Bulk	4142
Strategy 5	All containers, All bulk	4319

Table 9.1: Overview of the implementation strategies

9.2. Implementation costs

The costs to set up a cold buffer process differ per configuration. In configuration 1 there is one buffer and there transport is done manually. Handlers are needed at the buffer location to make sure everything is done correctly and drivers are needed to drive the tugs. Because the configurations uses a simple set up so this does not require additional costs. The cost for a driver is on average €80 per hour. The cost for a handler is on average €65 per hour. The difference in cost is due the depreciation costs of the tugs, which is included in the total driver costs. All the personnel has to work a minimum of three hours per shift. In the trial KLM ran on cold buffering they needed four drivers and one handler at the storage location to process the cold bags from fifteen airplanes, all the baggage was containerized. Based on this number of drivers and the number of cold tugs that were found out in the simulation results the total number of drivers can be derived for each chosen strategy. In the second configuration autonomous tugs were used. There are several suppliers of autonomous tugs. Two companies were approached and the cost of a autonomous tug at company 1 was €100.000 and at the second company €225.000. It is not known what the deprecation period of an autonomous tug is. The legal depreciation maximum is 20%. Based on this 20% a cost per day is calculated of €56 for the first company and €125 for the second. The are other costs involved with the usage of autonomous tugs such as maintenance and charging but these cost are not yet known and left out of scope. The implementation costs will be given for configuration 1, configuration 2a with the lower priced autonomous tug and for configuration 2b with the higher priced autonomous tugs. For each configuration the cost is calculated per strategy for the duration of one day, based on that the costs are calculated on a monthly and yearly basis. The results are shown in table 9.2. The costs per bag are calculated as well by dividing the cost per day by the amount of cold bags that are buffered. These are in the table but also visualized in figure 9.1.

Configuration 1	**Tugs**	**Handlers**	**Cost per day**	**Cost per month**	**Cost per year**	**Bags**	**Cost per bag**
Strategy 1	5	1	€ 1.395	€ 41.850	€ 502.200	2494	€ 0,5593
Strategy 2	11	1	€ 2.835	€ 85.050	€ 1.020.600	3322	€ 0,8534
Strategy 3	17	1	€ 4.275	€ 128.250	€ 1.539.000	3835	€ 1,1147
Strategy 4	23	1	€ 5.715	€ 171.450	€ 2.057.400	4142	€ 1,3798
Strategy 5	29	1	€ 7.155	€ 214.650	€ 2.575.800	4319	€ 1,6566

Configuration 2A	**Tugs**	**Handlers**	**Cost per day**	**Cost per month**	**Cost per year**	**Bags**	**Cost per bag**
Strategy 1	5	3	€ 473	€ 14.183	€ 170.200	2494	€ 0,1896
Strategy 2	12	3	€ 862	€ 25.850	€ 310.200	3322	€ 0,2594
Strategy 3	18	3	€ 1.195	€ 35.850	€ 430.200	3835	€ 0,3116
Strategy 4	24	3	€ 1.528	€ 45.850	€ 550.200	4142	€ 0,3690
Strategy 5	30	3	€ 1.862	€ 55.850	€ 670.200	4319	€ 0,4310

Configuration 2B	**Tugs**	**Handlers**	**Cost per day**	**Cost per month**	**Cost per year**	**Bags**	**Cost per bag**
Strategy 1	5	3	€ 820	€ 24.600	€ 295.200	2494	€ 0,3288
Strategy 2	12	3	€ 1.695	€ 50.850	€ 610.200	3322	€ 0,5102
Strategy 3	18	3	€ 2.445	€ 73.350	€ 880.200	3835	€ 0,6375
Strategy 4	24	3	€ 3.195	€ 95.850	€ 1.150.200	4142	€ 0,7714
Strategy 5	30	3	€ 3.945	€ 118.350	€ 1.420.200	4319	€ 0,9134

Table 9.2: Implementation costs for the different strategies and configurations for one year

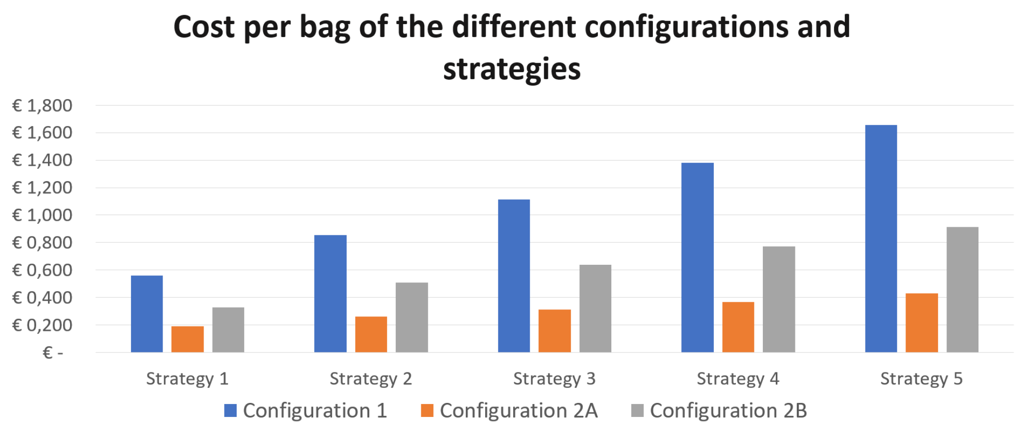


Figure 9.1: Individual costs of buffering per bag

9.3. Implementation locations

In order to store all the cold bags a certain area is needed. Based on the amount of cold bags inside the airplane, a number of containers and carts is derived. For the 16 containerized flights 73 containers are needed for the cold transfer baggage. For the 68 flights with bulk baggage 95 carts are needed for the cold transfer baggage. A baggage container has a maximum length of 1.92m and width of 1.37m. The containers are transported on dollies which are a bit bigger than the container itself. The carts in which the bulk baggage is loaded have a length of 2.63m and width of 1.28m. Both the dolly and the cart have a drawbar attached to it with a maximum length of 1 meter. Including the drawbar, one containers covers 4.00 m^2 and one cart covers 4.65 m^2. Because there are 95 carts and 73 containers a total of 733.44 m^2 is needed in order to park all of them. In configuration 1 this is all needed at one location and in configuration 2 this is divided over 3 locations.

9.4. Conclusion

From the results of the simulations it followed that configuration 1 had a higher overall peak shaving but the pressure on the individual infeed points was also higher compared to configuration 2. This is not suitable for operations because a small mistake could then lead to huge jams in the system. Based on the cost analysis configuration 2 is less expensive than configuration 1 regardless of the choice of company for autonomous tugs. Using autonomous tugs would also save the need for personnel which becomes harder to find nowadays. At a busy airport it might also be more convenient to find three smaller storage spaces to park the cold carts and containers. Due to all these arguments the most desired implementation for Schiphol airport would be configuration 2.

10 Discussion

All the results from the experiments were given in the previous chapter, in this chapter the results are further discussed and insights will be drawn from them. At the end of this chapter the limitations in this research will be discussed.

10.1. Key findings and interpretations

A numerical result of all the experiments was presented in table 8.1. What can be observed here is that all the experiments where cold buffering took place there was a significant peak reduction. The minimum is 18.2% in experiment 3 and the maximum is 34.9% in experiment 5. Configuration 1 has an average peak reduction of 29.7% and configuration 2 has an average peak reduction of 27.2%. In chapter 4 the potential was calculated to be 23%. This calculation did not take any minimal batch sizes of bags in to account. The reason the peak reduction is on average higher in configuration 1 compared to configuration 2 is because in configuration all the cold bags are buffered and then brought to the South infeed hall, during the simulation without a buffer the South infeed hall had almost no infeed between 9:30 and 11:30 giving it more space to feed in bags. In configuration 2 all three infeed points were used but because most bags were predetermined to go to the E-hall this is also were most of the cold bags were fed in. During the simulation without a buffer the E-hall was still quite busy with the infeed of normal transfer baggage between 9:30 and 11:30. Especially is scenario 2 this became a limitation because the D-hall was out of order in that scenario and E-hall became even busier. The difference between configuration 1 and 2 is the biggest here of all the scenarios.

Scenario 1 had an increase in flights that arrive during the hours of 7:00 and 9:00 and without buffering this would increase the morning peak from 1261 to 1409 bags per quarter hour. This is an increase of 11.7% at the busiest moment. With buffering the morning peaks of this scenario were comparable to the other scenarios where there was no increase in number of flights. Scenario 3 had a lower peak reduction compared to scenario 1 and 2. This is most likely due to the minimal limit of cold bags an airplane should contain before it is qualified for the hot and cold baggage separation. This limit was 5 bags for bulk baggage and 25 bags for containerized baggage. Because the limit for containerized flights is higher less flights will qualify in this scenario and thus less baggage will be separated and buffered.

Configuration 1 had a higher peak reduction on average compared to configuration 2 but the South infeed hall has to operate at maximum capacity for several hours in all the scenarios with configuration 1 in order to feed in all the buffered baggage. In Experiment 5 and 8, the combination of configuration 1 with scenario 1 and 2 respectfully, it can be seen that the width of the buffering period is wider than in experiment 2 and 11. This means that bags have to wait at the infeed point before being able to be fed in to the system. This might bring problems later on in the system with on time performance of the bags at the make-up stations. Based on the time individual infeed location work at maximum capacity in order to process all the bags configuration 2 would be the preferable solution in this aspect of operations.

Another aspect of operations is the resources needed in order to carry out all the work. An indication of this could be amount of tugs needed. In the simulation where buffers were implemented the cold carts and containers that arrived during the morning peak were picked up by 'cold tugs' and the hot baggage and baggage that arrived outside the peak were picked up by normal tugs. Because this was a simulation model tug queues and cold tug queues were made that could be filled with carts and containers. In the carts and container process were lines of code that would make sure they pick the shortest waiting line. In configuration 0, without buffering, this resulted in 12 used queues for scenario 0, 2 and 3 and 13 used queues in scenario 1. This increased because more flights needed to be handled.

In configuration 1 and 2 the cold tug queues were also used. All with quite similar results. Scenario 1 had 29 cold tug queues for configuration and 30 for configuration 2. Autonomous tugs drive slower than normal tugs, this is likely the reason why one more cold tug queue was needed. A similar increase in cold tugs can be seen in the other scenarios between configuration 1 and 2. An increase in cold tug queues can be seen in scenario 1 because there are more cold bags in this scenario. A decrease can be seen in scenario 3 because there are less cold bags or less tugs because more bags fit in a container than in a cart and the transport is more efficient. While the outcome of the infeed times in the simulation model looks realistic the amount of tug queues is not directly linked to the actual amount of tugs and tug drivers needed. A lot more than 12 tugs are driving around during morning operations and the increase in tugs needed including cold tugs does not feel logical. The way tug queues and cold tug queues are chosen is the same and the process of that queue is similar as well.

10.2. Research limitations

In chapter 3 a third buffer configuration was suggested using autonomous carts and individual buffer times for cold baggage. In order to limit the running time of the simulation all the bags were batched per flight and only the amount of cold and hot bags onboard was known. The data of the individual bag such as it transfer time was not taken in to account after the batching process. In order to make a simulation with autonomous carts work this needed to change but then a completely new simulation model should be build. This simulation model would then not be comparable to buffer configurations 1 and 2. It was decided to not build this new simulation model but the outcome of this configuration would have been really interesting because of the individual buffering times for all the bags. In this way the morning peak could be lowered but it could also be made sure that a new peak during the buffer infeed moments would not occur. Currently this would also be the most far fetched configuration based on the technology and free space available at the airports.

Airports are a difficult ecosystem, in order to create a realistic representation of the baggage handling process minimal changes were made to the chosen infeed locations of the baggage. It is only done in scenario 2 where the D-hall is closed down and bags are redirected to the South and E-hall with the same ratio as they had before. The reason to keep the same ratio was to try and not change a chosen infeed hall distribution chosen by planning experts and in order to keep at realistic as possible. If the chosen infeed location could change in the buffer simulation based on capacity and availability the peak reduction could be increased. But this was a simulation model to represent the current situation with a small change and not a optimization model to find the perfect way of cold buffering. Using an optimization model the needed tugs could also be more accurately predicted.

11 Conclusion and Recommendations

In this chapter the research will be concluded and recommendations on future proceedings will be given. The framework for this will be the answers to the sub questions derived in chapter 1 in order to answer the main research question:

What is the effect of cold buffering of transfer baggage and in what ways can this be done?

11.1. Conclusion

In order to answer the main research question several sub questions had to be found out first. The first question was:*What does a baggage journey currently look like and what are the KPIs?*
A visualisation was made of the baggage journey with in and outflow of baggage. This is shown figure 11.1.

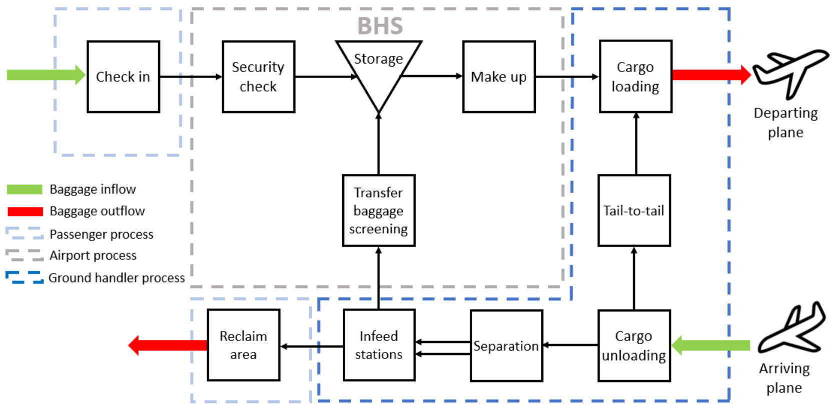


Figure 11.1: Visulaization of the baggage joruney

Baggage comes in either by passenger at the check in area or by arriving airplane. From the check in the baggage goes through the baggage handling system to make up. Baggage that arrives by airplane is unloaded and separated in to either transfer or reclaim baggage. Baggage can arrive arrive as loose pieces in bulk or inside a container. Reclaim baggage is transported to the infeed points of the reclaim area where passenger can pick it up and leave the airport. Transfer baggage is transported to infeed points where it can enter the BHS and join the check in baggage at the make up stations. At the make up stations the baggage that belongs to one flight is collected and brought to the departing airplane. Three main key performance indicators of the whole system where chosen. These were: System capacity, throughput time and number of mishandled bags.

The second research question was: *What is cold buffering and what are the design options to set up a cold buffering process?*
In transfer baggage a distinction can be made between three types of bags. All of them are categorized by a temperature. The first type is super hot. Super hot transfer baggage must be brought directly from the arriving airplane to a departing airplane without going trough the BHS. The transfer time is shorter than the throughput time. This baggage is also called tail to tail baggage and the process is shown in figure 11.1. The other two types of transfer baggage do have enough time to go through the BHS. They are called hot and cold transfer baggage The distinction between these two is the total transfer time, an airport can decide what the transition time is where cold baggage turns hot. In the current baggage journey there is no separation between hot and cold baggage, all the bags are collected from the airplane

and brought to the transfer infeed points without any prioritisation. The idea of a cold buffering process is to separate the hot from the cold baggage before going to the infeed points and to bring the cold transfer baggage to a buffer storage. It will be stored for a specified amount of time and it will than be brought to the infeed points. The goal of this is to shave of the peak of the infeed of baggage. Baggage infeed is not constant during the day and especially in the morning a huge peak can arise. In order to set up a cold buffering process research was done on different ways of transportation and transportation of baggage. For both topics three main options were chosen. For transportation there were the options of: Normal transportation with tug drivers who pull carts and containers, autonomous tugs who pull carts and containers and the last option was self driving autonomous carts and containers. The first option is the way transfer baggage is currently being transported at most airports. The other two options depend more the development of autonomy around and on airports. The three main options for a buffer storage were: a simple parking space where carts and containers are stored, the second option was a space where carts and containers could be parked but baggage can be swapped around based on the transfer time, a priority is given in the buffer. The third option was an advanced buffer storage where carts and containers are fully unloaded and bags are stored individually. Based on the departure time of the transferring flight bags will have a personal buffering time and will be released after that. Choosing an option for this is based on the amount of available space and the preferred amount of buffer storage locations. Together with the transition time all design options were worked out. They can be found in table 11.1.

Transition time	**Transport options**	**Buffer configuration**	**Buffer placement**
x hours	Current transportation	Buffering with carts	1 location
y hours	Self-driving tug	Buffering with bins	2 locations
z hours	Self-driving baggage cart	Buffering individual bags	3 locations

Table 11.1: Design options

All design options could theoretically be combined which would give 81 configurations. It was chosen to focus on three main ones which would be realistic configurations to build at airports and especially at Schiphol airport and are logical combinations of design options. The three configurations are:

- 3 hours - Current transportation - Buffering with carts - one location.
- 3 hours - Self-driving tug - Buffering with carts - three locations.
- 2 hours - Self-driving baggage carts- Buffering individual bags - one location.

The third research question was: *What would cold buffering look like at Schiphol Airport?*
At Schiphol airport there three main transfer infeed locations, E-hall, D-hall and the South hall. All three locations have a different infeed capacity and available space for a buffer storage. For the first buffer configuration the buffer would be placed at the South hall, this one has the lowest capacity but a lot of that is still unused and it has the most available space and could store all the cold bags. For the second option three smaller buffer locations would be used and these would fit at the three infeed locations. The third option needs a lot more space and infrastructure and that will not fit close to an infeed station. A nearby location must be found where the cold bags could be brought to. After buffering the cold bags from this location could go to all three infeed stations.

The fourth research question was: *What is potential of cold buffering on peak baggage load?*
A week of transfer baggage data was supplied by Schiphol airport. In the data the arrival time and flight number of the each bags was known, as well as the infeed time, departure time of transferring flight and more. Based on this a data analysis was done on the infeed times of bags transfer during the day of the whole system and per individual infeed point and the transfer times were calculated. To all the cold bags that arrived between 7:00 and 9:00 two hours were added to the infeed time to see what effect this would have. This resulted in a peak reduction of 27%. In order to test out the different design options a model had to made.

The fifth research question was: *In what way could the cold buffering process be modelled?*
Literature research was done on multiple types of modeling. These types were predictive modelling, simulation modelling, optimization modelling and queue modelling. The four modelling types all have their own advantages and disadvantages. Predictive modelling looks for patterns in existing data and based on that predicts the outcome with a given other input cannot show what might happen if another configuration is implemented. This is something that simulation can do. Physical real world test can be expensive for airports and might interfere with the day to day operations. By using simulation modelling it becomes possible to test different solutions and see what effect they have on the

performance of a system. With queue modelling this is also possible but this will only model a small part of the system and it is harder to connect with the other parts of the system. In order to optimize a model all the constraints must first be known this is not yet the case for a model that wants to try out cold buffering. In the end it was decided to go for a simulation model. The goal of this simulation model is model the process from the moment an airplane arrives until the moment a transfer bag is fed in to the system, reclaim and tail to tail baggage is left out of scope. A transfer bag undergoes several processes which take a certain amount of time during this journey. These steps are shown in figure 11.2.

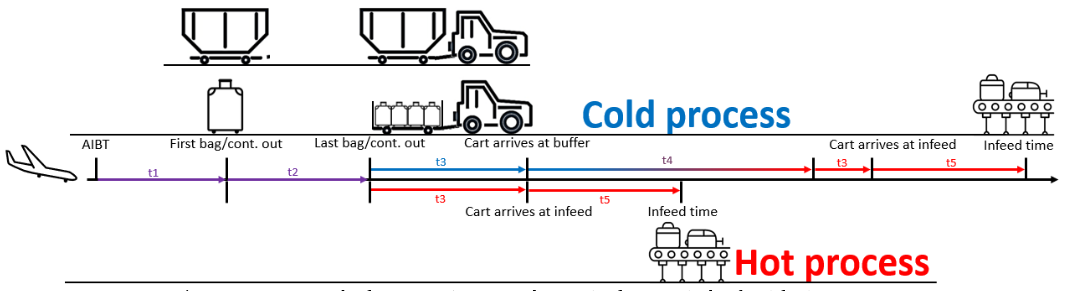


Figure 11.2: transfer baggage journey from airplane to infeed with time steps

The first step is the from actual in block time (AIBT) until the moment unloading can begin. This time is defined as t1. The AIBT is the moment an airplane has arrived at the apron and the blocks are placed under the wheels to prohibit any movement. The total unloading time is t2. t2 is depended on the amount of bags in an airplane and if these are in bulk or containers. After unloading two different processes start, either the cold process or the hot process. In the cold process a tug picks up the carts or containers and drives them to a buffer in driving time t3. Then the bags are buffered for the chosen buffer time t4. After that the bags must go to the infeed points, this takes some driving time t3 again. The last step is the infeed process, the time this takes is defined by t5. The hot process skips the buffer and drives directly to the infeed points. To build the simulation model all the elements in the mentioned process must be made. These elements are airplanes, carts, containers, tugs, buffer storages and infeed points.All these elements have their own attributes which contain the time steps or capacity and all the elements have their own process description language (PDL). This is the way a cold buffering process can be modelled. The output of the simulation model are the KPIs of the cold buffering process. The first output is the infeed time of each individual bag. Based on these infeed times the distribution over a day can be made and the infeed peak can be determined for the whole system. The second output of the model will be the infeed location of each individual bag. Based on the infeed location and time the pressure on the individual infeed points can be determined. The last output is the amount of tug it takes to process all the transfer baggage.

The sixth research question was: *What would be the effect of the different design options on the KPIs?* Several experiments were simulated in the model. The experiments were defined as a combination of a scenario with a configuration. The scenarios are different options for the input of the model and the configurations are different combinations of design options that are in the model. There were four scenarios. A scenario with the normal input data without any changes, this was chosen as a baseline scenario to compare the others against. The second scenario is an increase of arriving flights during the morning peak, this was chosen because there is a continuous growth in airplane movements across the world. The third scenario is one where one of the infeed locations will become unavailable so there is an increased pressure on the other locations. The fourth scenario is one where relatively more baggage will be containerized instead of bulk baggage. The three configurations are no buffering, buffering with normal tugs and one buffer and the last configuration is to buffer with autonomous tugs and 3 buffer locations. This resulted in 12 experiments in total. The outcome of the simulation model were all the infeed times of the individual bags in the system. These were grouped together in time slots of 15 minutes. Figures were made comparing the experiments of buffering and not buffering in each scenario to see if a peak reduction occurred. The experiments with configuration 1 had a an average peak reduction of 29.7%. The experiments with configuration 2 had a an average peak reduction of 27.2%. But if a closer look is taken in to individual infeed capacities of the infeed stations it was found that in the experiments with configuration 1 more time was spent operating at full capacity for the South infeed station. This is less desirable for an airport because this makes the process more fragile. On average 30 extra tugs were needed to transport all the transfer baggage in configuration 1. In configuration 2 this was 31 tugs.

The last research question was: *What would be the most desired implementation for Schiphol airport?*

From the results of the simulations it followed that configuration 1 had a higher overall peak shaving but the pressure on the individual infeed points was also higher compared to configuration 2. This is not suitable for operations because a small mistake could then lead to huge jams in the system. Based on the cost analysis configuration 2 is less expensive than configuration 1 regardless of the choice of company for autonomous tugs. Using autonomous tugs would also save the need for personnel which becomes harder to find nowadays. At a busy airport it might also be more convenient to find three smaller storage spaces to park the cold carts and containers. Due to all these arguments the most desired implementation for Schiphol airport would be configuration 2.

All these subquestions were answered in order to answer the main questions: *What is the effect of cold buffering of transfer baggage and in what ways can this be done?*
The effect of cold buffering is reduction of the morning peak by distributing the infeed of the transfer bags over a longer time period. This will reduce the pressure on the infeed stations and leave room to either downscale the amount of machine in use or upscale the amount of flights that land at an airport. The cold buffering process can be set up in a lot of ways, several design options were discussed in this research and two configurations were tested out in simulation.

11.2. Recommendations

I recommend that further research should be done on the last defined configuration that I was not able to test in this model. The reason I was unable to do this was that the model looked at bags as a batch and this configuration needed individual pieces. The individual pieces is what makes this configuration interesting because it could increase the number of transfer bags that could be labeled cold. Future research in this topic of cold buffering would be really interesting, there is still a huge literature gap and I think there is a lot of optimization possible within the configurations. This is something an optimization model would be perfect for. In order to do this all the constraints must be known but these can be figured out with the current simulation model. Next to that it would also be possible to apply parameter tuning on the current model to see if it can become more realistic.

References

AlKheder, S., Alomair, A., & Aladwani, B. (2019). Hold baggage security screening system in kuwait international airport using arena software. *Ain Shams Engineering Journal*. https://doi.org/10.1016/j.asej.2019.10.016

Bardinas, F., Caguioa, J., Cariño, S., Saguid, W., Tonga, M., & Pineda, E. (2019). Airport management system: Core transaction 2 (iata boarding pass printing/iata bag tag printing, departure control, arrival control, interactive seat maps). *Ascendens Asia Singapore – Bestlink College of the Philippines Journal of Multidisciplinary Research*.

Barnes, S. (2018). Klm baggage and hold containers on the apron at schiphol airport. https://www.alamy.com/klm-baggage-and-hold-containers-on-the-apron-at-schiphol-airport-image212405802.html

Barth, T. C., Holm, J. T., Larsen, J. L., & Pisinger, D. (2021). Optimization of transfer baggage handling in a major transit airport. *Operations Research Forum*, 2. https://doi.org/10.1007/s43069-021-00058-z

Bolijn, R. (2018). *Improving productivity of driving processes for special baggage handling at klm royal dutch airlines*. www.mtt.tudelft.nl

Boute, S. (2016). *A future baggage reclaim innovating around the passenger at the a-area*.

Brandt, F. (2017). The air cargo load planning problem.

Bruno, G., & Genovese, A. (2010). A mathematical model for the optimization of the airport check-in service problem. *Electronic Notes in Discrete Mathematics*, *36*, 703–710. https://doi.org/10.1016/j.endm.2010.05.089

Bryndin, E. (2021). Robotization of service with goods and products via automatic cabinet. *International Robotics & Automation Journal*, *7*(1), 20–22.

Cavada, J. P., Cortés, C. E., & Rey, P. A. (2017). A simulation approach to modelling baggage handling systems at an international airport. *Simulation Modelling Practice and Theory*, *75*, 146–164. https://doi.org/10.1016/j.simpat.2017.01.006

Cura. (2020). Movus autonomous airside cargo vehicle. https://cura.design/works/movus/

Gosling, G. D., & Barton, D. K. (n.d.). Implications of autonomous vehicles for airport planning, design, and operations. In *International conference on transportation and development 2022* (pp. 154–167). https://doi.org/10.1061/9780784484371.015

Guti, M. (2018). Kar met bagage bij een luchthaven. https://www.istockphoto.com/nl/foto/kar-met-bagage-bij-een-luchthaven-gm987785970-267867201?phrase=luggage

Hafilah, D., Radja, A., Rakoto-Ravalontsalama, N., Lafdail, Y., Hafilah, D. L., Radja, A. C., & Rakoto, N. (2017). *Modeling and simulation of baggage handling system in a large airport. the 18th asia pacific industrial engineering and management system conference*. https://hal.archives-ouvertes.fr/hal-01686750

Holm, T., Larsen, L., Barth, T. C., Holm, J. T., & Larsen, J. L. (2013). *A model for transfer baggage handling at airports*.

Hsu, C. I., Chao, C. C., & Shih, K. Y. (2012). Dynamic allocation of check-in facilities and dynamic assignment of passengers at air terminals. *Computers and Industrial Engineering*, *63*, 410–417. https://doi.org/10.1016/j.cie.2012.04.003

Huang, E., Liu, I., & Lin, J. T. (2018). Robust model for the assignment of outgoing flights on airport baggage unloading areas. *Transportation Research Part E: Logistics and Transportation Review*, *115*, 110–125. https://doi.org/10.1016/j.tre.2018.04.012

Huang, E., Mital, P., Goetschalckx, M., & Wu, K. (2016). Optimal assignment of airport baggage unloading zones to outgoing flights. *Transportation Research Part E: Logistics and Transportation Review*, *94*, 110–122. https://doi.org/10.1016/j.tre.2016.07.012

Jacobs, S. (2021). *Challenges regarding the implementation of an autonomous baggage tractor on amsterdam airport schiphol*.

JEC. (2017). Jec constructs baggage handling systems for arrival bags delivery and early bag storage. https://www.jardines.com/en/news-and-views/jec-constructs-baggage-handling-systems-arrival-bags-delivery-and-early-bag-storage

Le, V. T., Zhang, J., Johnstone, M., Nahavandi, S., & Creighton, D. (2012). A generalised data analysis approach for baggage handling systems simulation. *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, 1681–1687. https://doi.org/10.1109/ICSMC.2012.6377979

Lee, S., & Seo, S.-W. (2022). Sensor fusion for aircraft detection at airport ramps using conditional random fields. *IEEE Transactions on Intelligent Transportation Systems*, 1–13. https://doi.org/10.1109/TITS.2022.3157809

Leone, K., & Liu, R. (2005). The key design parameters of checked baggage security screening systems in airports. *Journal of Air Transport Management*, *11*, 69–78. https://doi.org/10.1016/j.jairtraman.2004.09.004

LHR. (2017). Lhr airports limited storage. https://i.dailymail.co.uk/i/pix/2013/12/23/article-2528042-1A45D60600000578-20_964x659.jpg

Lin, J. T., & Liou, I. C. C.-C. (2015). A simulation-based analysis for inter release problem in airport baggage handling systems. In K. J., H. Xiaoxia, H. Y. G. Mitsuo, & Kim (Eds.). Springer Berlin Heidelberg.

Lin, J. T., Shih, P. H., Huang, E., & Chiu, C. C. (2015). Airport baggage handling system simulation modeling using sysml. *IEOM 2015 - 5th International Conference on Industrial Engineering and Operations Management, Proceeding*. https://doi.org/10.1109/IEOM.2015.7093764

Loenhout, S. M. A. P. V. (2019). *Optimizing outbound baggage handling at klm*. http://repository.tudelft.nl/.

Ma, Q., Bi, J., Sai, Q., & Li, Z. (2021). Research on prediction of checked-baggage departed from airport terminal based on time series analysis. *Proceedings - 2021 7th Annual International Conference on Network and Information Systems for Computers, ICNISC 2021*, 264–269. https://doi.org/10.1109/ICNISC54316.2021.00055

Malandri, C., Briccoli, M., Mantecchini, L., & Paganelli, F. (2018). A discrete event simulation model for inbound baggage handling. *Transportation Research Procedia*, *35*, 295–304. https://doi.org/10.1016/j.trpro.2018.12.008

Markus, D., & Frey, M. (2014). *Models and methods for optimizing baggage handling at airports*.

Parlar, M., & Sharafali, M. (2008). Dynamic allocation of airline check-in counters: A queueing optimization approach. *Management Science*, *54*, 1410–1424. https://doi.org/10.1287/mnsc.1070.0842

Pole, J. (2022). Summer travel at risk as europe's airports face mass staff shortages. https://www.euronews.com/travel/2022/06/22/europes-airports-struggle-with-mass-staff-shortages-as-travel-sector-faces-summer-of-disco

Robinson, S. (1997). *Simulation model verification and validation: Increasing the users' confidence*. Aston University. https://www.informs-sim.org/wsc97papers/0053.PDF

Royal-Schiphol-Group. (2019). Bagage op schiphol.

Silven, A. A. E. (2018). *Autonomous transport robots in baggage handling systems a study on the use of autonomous individual transport robots in baggage handling systems at medium-sized regional airports operating in a point-to-point network*. https://repository.tudelft.nl/

Staat. (2022). Vliegverkeer schiphol. https://magazines.ilent.nl/staatvan/2021/01/

Tarau, A. N., Schutter, B. D., & Hellendoorn, J. (2009). *Centralized, decentralized, and distributed model predictive control for route choice in automated baggage handling systems* (3).

van Wingerden, A. (2022). *A baas ecosystem for aas*.

Wuisman, I. (2016). *Simulating the performance of the integral transfer baggage handling process at klm*. www.mtt.tudelft.nl

Zeinaly, Y., Schutter, B. D., & Hellendoorn, H. (2012). A model predictive control approach for the line balancing in baggage handling systems. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, *45*, 215–220. https://doi.org/10.3182/20120912-3-BG-2031.00043

A
Paper

Paper starts on the next page.

Reducing peak stress on the Baggage Handling System

A research in to the effect of buffering cold transfer baggage at Schiphol airport

F.N.S. van der Grift, C. Roos, M.B. Duinkerken, R.R. Negenborn

***Abstract*— Air transport has gone from a once in a lifetime experience for the extremely rich to a day to day method of transportation for everybody. With the increase in passenger movements the amount of baggage that goes through airports has increased as well. This could cause problems for airports if they have a limited capacity in one of the parts of the baggage journey. An important part is transfer baggage, if something goes wrong the transfer bag misses their connecting flight which is an expensive problems for the airlines and airports. To cope with the increase in demand of the capacity there are several solutions. The solution that will be explored in this paper is to make a more efficient use of the capacity that is available by making a prioritization in transfer baggage based on transfer time. The process in called cold buffering and the aim is flatten the infeed peaks. In order to explore this potential solution the baggage journey was explored and part of journey was modelled in a simulation. From the model it followed that it is possible to obtain 28.4% of peak shaving by using the cold buffering.**

Keywords: Simulation modelling, airport, baggage, peak shaving, cold buffering.

I. Introduction

Amsterdam Schiphol Airport is a large hub airport used by 71,7 million passengers in 2019. Although the passenger numbers dropped in 2020 and 2021 due to COVID19, Amsterdam Schiphol Airport was Europe's busiest airport in 2021. Each year, Schiphol handles some 53 million pieces of baggage. This can vary from day to day from around 120,000 items on a slow day to 180,000 during extremely busy periods, such as the start of a holiday season. Almost 40 percent of all these items are transfer baggage(Royal-Schiphol-Group, 2019). Due to an increase in demand of travel and an upward trend in flight movements, the demand on the baggage handling system has increased. KLM is responsible for the biggest share of flight movements on Schiphol airport. Their business model uses Schiphol airport as an important transfer hub. To improve customers travel experiences, KLM tries to keep the transfer time a short as possible. To do this they first plan to let the intercontinental flights come in so that these passengers can transfer to the European flights that come in a bit later in a second wave. This all happens in the morning which creates a morning peak in incoming flights which can be seen in Figure 1. The upper figure shows number of flights arriving at Schiphol airport per 15 minutes interval. The numbers are averages over the busy summer months of July till September in 2018 and 2019. In the second figure the number of transfer bags that are fed in per 15 minutes interval can bee seen. The peaks in arriving flights are followed by peaks at the infeed stations of transfer baggage, especially in the morning.

Due to this morning peak, problems could occur at the

Delft University of Technology, Faculty of 3mE, Mekelweg 2, 2628CD Delft

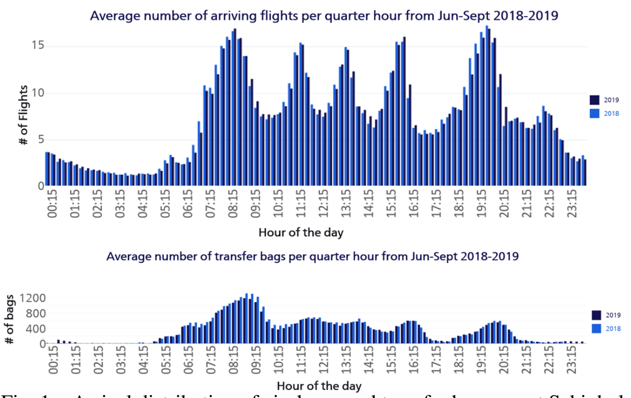


Fig. 1. Arrival distribution of airplanes and transfer baggage at Schiphol airport (Royal-Schiphol-Group, 2019).

infeed points of the BHS which also affects the throughput of the whole system at the storage and the makeup areas. This can cause delays in flights or lost baggage which both cost a lot of money for the airlines and the airport. In order to cope with this problem airports could either increase their infeed capacity or use the available capacity more efficiently. The aim of this research is to explore potential solutions of making more efficient use of the existing capacity. The solutions will be explored by first doing a literature review on the topic of the baggage journey and modelling at airports in section II. Secondly the design options to set up a cold buffering process are presented in section III. In the next section, section IV, the baggage data gathered from Schiphol airport is explored. After that the set up of the model will be explained with the chosen experiments in section V. The results from the experiments are presented in section VI. Those results will be discussed in section VII. And the end the research will be concluded and recommendations will be given in section VIII.

II. Literature Review

The literature review of this research consists of three main categories, the baggage journey, peak shaving and airport modelling. These topics will be discussed in this section of the paper.

A. The baggage journey

The baggage journey is a route bags can take at an airport. The journey for a bag can start at two places, these two places are indicated with green arrows in Figure 2. The first start is the check in desk. This is indicated by the 1a in Figure 2. Baggage is brought to the airport by passenger for their trip. The bags can be checked in at a human operated desk or a self check in (Parlar and Sharafali, 2008). A label is attached to the back so that it can be tracked along it's journey (Bardinas et al., 2019). After the baggage is checked in it enters the baggage handling system (BHS). First it has to go through a security check (Barros and Tomber, 2007). This happens at 2a in Figure 2. After that the BHS brings the baggage to either a storage or the correct make up station at 3 and 4 in Figure 2 respectively. Baggage is placed in the storage if the make up stations are not open yet. Make up stations open based on the departure time of an airplane. Make up stations are where the baggage is sorted by flight on laterals or carousels (Hafilah et al., 2017). The baggage is unloaded, mostly manually, and placed in carts or containers to continue their journey to the departing aircraft, where it will end their journey at this airport(Cavada et al., 2017),(Ganzewinkel, 2021). This is indicated by the red arrow.

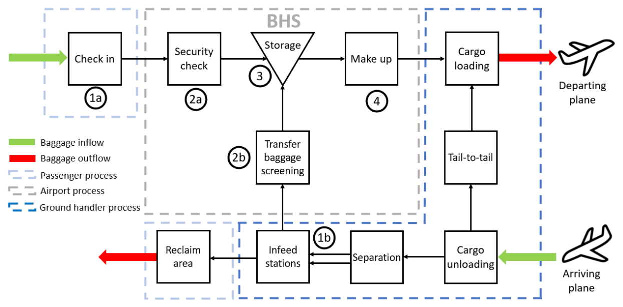


Fig. 2. Visualisation of the baggage journey

The second way baggage can start their journey at an airport is by arriving by airplane. This is indicated with the second green arrow in Figure 2. Baggage from the airplane can have two destinations, either it has arrived at final destination and it is reclaim baggage of it is transfer baggage and it has to be brought to another airplane. Usually this is done through the BHS but if the transfer time is shorter than the throughput time the bag must be transported tail to tail. The other transfer baggage and reclaim baggage is separated from each other and brought to infeed points. The infeed points for transfer baggage are connected with the BHS and this will join the checked in baggage. The transfer baggage can be split up in to two categories, hot and cold transfer baggage. The 'temperature' is based on the transfer time. Bags with a longer transfer time are called cold and bags with a shorter transfer time are called hot. Airports can decide for themselves what the transition time is where cold baggage turns hot. The reclaim baggage is brought to infeed points which are directly connected with the reclaim belts where passengers can pick up their baggage and leave the airport(Boute, 2016). thereby ending the baggage journey. This is indicated with the second red arrow in Figure 2. The key performance indicators of this whole journey are the throughput capacity, the throughput time and the amount of mishandled bags.

B. Peak shaving

In section I a distribution of the infeed of transfer baggage was shown. In Figure 1 it can be seen that the infeed of baggage is not evenly distributed over the day. If it would be possible to distribute this peak along the day the system could handle more baggage in total. This process is called peak shaving and this could be done with several solutions to both sides of the peak(van Wingerden, 2022).

To pull the peak to the 'left', baggage must enter the system earlier. This is only possible with check in baggage because transfer baggage cannot arrive earlier. There are several solutions to get baggage to the airport earlier. One of them is to do a home pick up of the baggage. This creates a better traveling experience for the passenger because they don't need to carry their baggage from home to the airport and the baggage arrives earlier at the airport (Vos, 2019).

To pull the peak to the 'right', baggage must enter the system later. This can be done with transfer baggage. One solution for this is called cold buffering. The aim of cold buffering is to pull the morning peak to the right by inserting the cold baggage later when there is less pressure on the BHS. This does come with its own challenges. The first one being the separation of the bags on the VOP. Unloading of the plane is currently mostly done manually as fast as possible. The airline wants to turn around the plane as fast as possible and deliver the baggage as fast as possible to their next destination(van der Lande, 2011). The second challenge is to have a system in place where the cold baggage can be buffered outside the BHS, a storage location. After the buffering time the baggage has tunred hot and must be brought to the BHS(van Dalen, 2010; van Wieren, 2021).

C. Airport modelling

In order to test out different solutions it is convenient to have a model to test the performance of that solution on the real world system without have to run an actual trial. Several ways of modelling were found in the literature such as predictive modelling, simulation, optimization and queue modelling.

Predictive modelling is a technique that uses machine learning, historical data and existing data to predict and forecast the most likely future outcome. It analyses the historical data to generate patterns. The existing data and the patterns are then used to predict the future events. This has been used to predict the arrival times of check in baggage Ma et al., 2021. With this prediction personnel could be more efficiently allocated. Another use of the predictive model is the predictive control of routing choices for baggage handling (Tarau et al., 2009).

Simulation modelling is a technique where a digital copy of a real world process is created to analyse and predict it's performance. The goal of the simulation is to accurately predict the outcome of different scenarios and to see how the system reacts. For example at Santiago Airport in Chile the amount of passengers has outgrown the BHS's capacity, their simulation is model of the whole baggage journey divided in to three parts, check in, behaviour, transportation and baggage loading area (Cavada et al., 2017). Here different ideas can be tested and their influences on other processes are also shown.

Optimization modelling attempts to find the optimal solution to a single or multiple complex equations using a set of constraints. This has been used to optimize to assignment of make-up areas to outgoing flights(Huang et al., 2016, Huang et al., 2018). The goal is to minimize the amount of unassigned flights. The BHS makes us of the assigning of flight by sorting the bags and transporting it to the right place.

Queue modelling can be used to predict queue lines and waiting times. This has been used in literature to to optimize dynamic assignment of check in counters for a flight with a known amount of passengers(Parlar and Sharafali, 2008). The queue modelling was used to derive arrival rates.

Based on the literature research it was chosen to explore the potential solution of cold buffering with simulation model. Cold buffering is chosen because there is a gap in the literature about that subject and the simulation model is chosen to test that because this can make a digital copy of the real world process and test out different configurations.

III. Design options

A cold buffer process can be set up in many ways, for this research four main categories were chosen. The first category is the transition time. The transition time is the time that is chosen to differentiate cold transfer baggage from hot transfer baggage. This time can be based on the time it normally takes transfer baggage to go from the airplane to the infeed stations, the time it takes to go through the BHS and the time bags should be buffered. The chosen transition time also has an effect on the amount of bags that are classified as cold. If the time is set shorter than there is more cold transfer baggage. For example, the average time it takes baggage to go to the infeed points and be put through the BHS at Schiphol airport is around 45 minutes. If baggage is buffered for 1 hour and a safety margin is taken the transition time can be set at two hours, if the buffer time is 2 hours the transition time should be 3 hours.

The second category is the way of transportation. Baggage can be transported in several ways, three main options were looked at. The first option is the current way of transportation at airports. After the plane is unloaded the baggage is in carts as bulk baggage or containerized. These carts and containers are picked up by tugs with drivers and brought to the infeed point. With the implementation of a cold buffer storage extra tugs will be needed to drive the cold bags there and the hot bags to the infeed points. This process could also be fully or partly automated. The second design option for transportation is the use of autonomous tugs. These have been tested at Schiphol airport (Jacobs, 2021). The autonomous tug drives a bit slower but the tests were successful. A third option is the use of autonomous carts(Lee and Seo, 2022). These have as advantage that they can drive individually which will save time because a full cart can start driving immediately and does not have to wait for the entire loading and unloading process.

The third category is the buffer storage set up. A buffer storage can be set up in many ways. Three options were worked out. The first one being a simple parking space where carts and containers are stored. This could be done at many places around an airport and does not require a lot of construction. The second option is a space where carts and containers could be parked but baggage can be swapped around based on the transfer time, a priority is given in the buffer. The third option is an advanced buffer storage where carts and containers are fully unloaded and bags are stored individually. Based on the departure time of the transferring flight bags will have a personal buffering time and will be released after that. This requires more space and construction.

The fourth category is the amount of buffer that are placed on an airport. This is depended on the size of the airport, the set up of the storage and the available space at an airport. If a set up with simple parking is chosen as the buffer storage and there is enough space it is easier to build several storages. If the option is to build a buffer storage that is a small baggage handling system it would need a lot of space and won't be build multiple times at one airport.

From all the categories a design option can be picked which are combined in a configuration. All the different design options are shown in Table I

Transition time	**Transport options**	**Buffer configuration**	**Buffer placement**
x hours	Current transportation	Buffering with carts	1 location
y hours	Self-driving tug	Buffering with bins	2 locations
z hours	Self-driving baggage cart	Buffering individual bags	3 locations

TABLE I
Design options

IV. Data analysis

A week of transfer baggage data was provided by the IT&Data department of the Schiphol Group. This dataset contained all the available data per individual bag such as arrival times, infeed times, departure times of the next flight, flight numbers, aircraft types of the flights and landing and infeed areas. Infeed distributions were made for all the individual days the week. An example of this can be seen in Figure 3, which shows the distribution of infeed bags on the 14th of July in timeslots of 15 minutes. With the data it was also possible to look at the infeed

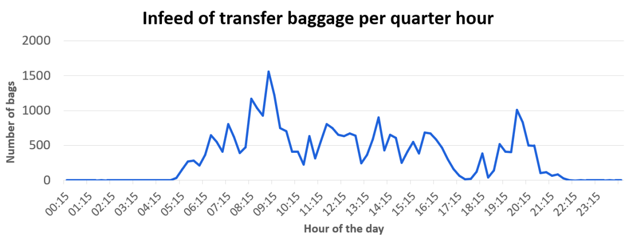


Fig. 3. Distribution of baggage entering the BHS

distributions of the individual infeed point E, D and South. This is shown in Figure 4. The transfer times of all the

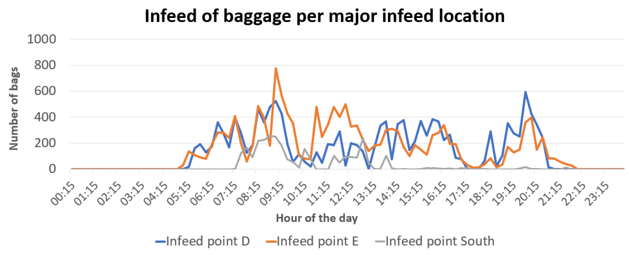


Fig. 4. Distribution of baggage entering the BHS at individual infeed points

bags were calculated on this day of the dataset. All the bags that had a longer transfer time than three hours were classified as cold. Two hours were added to the infeed time for all the cold bags that arrived between 7:00 and 9:00 to see what effect this would have. This resulted in a peak reduction of 27%. This is a peak reduction in the overall infeed. The original peak of 1558 is reduced to 1130 bags that are fed in per quarter hour. In total 7209 bags were fed in to the system between 8 and 10 am. Of these 7209 bags, 3093 were cold and 4116 were hot. Of the 3093 cold bags, 1402 came from containers and 1691 from bulk baggage.

V. Model set up

In section II it was decided that the best way to test out cold buffering would to create a simulation model. This section explains the different steps that are modelled, which choices were made and which experiments will be simulated.

A. Model

The simulation model should be able to simulate the process for transfer baggage from an arriving airplane until it arrives at an infeed point. The model should be able to compare the performance of the system when the cold buffer is being used and when it is not being used. The performance of the system is measured by the height of the peak of infeed of baggage. In Figure 5 the whole process is shown. The input of the model is a day of data from the given dataset of Schiphol airport and the baggage is batched together by flight. The model will have three main outputs. The first is the infeed time of each individual bag. The second output is the infeed location of each individual bag. The third output is the extra amount of tugs that the simulation needs to transport the cold bags. If an airplane arrives between 7:00 and 9:00 AM the baggage is separated between cold and hot transfer baggage and the cold transfer baggage is brought to the buffer. The hot baggage is brought to the infeed point immediately. The output of the model is an infeed time for every bag that has gone through this process. Each step in the process takes a certain amount of time, these steps are shown with their time steps in Figure 6. All the chosen times are based on existing models Schiphol airports uses for their process and personnel measurements taken on excursions to the airside.

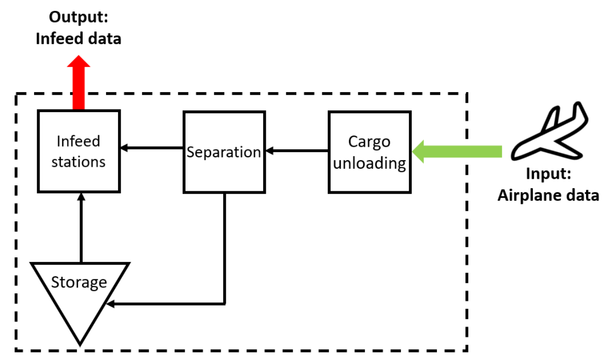


Fig. 5. Boundaries of the simulation model

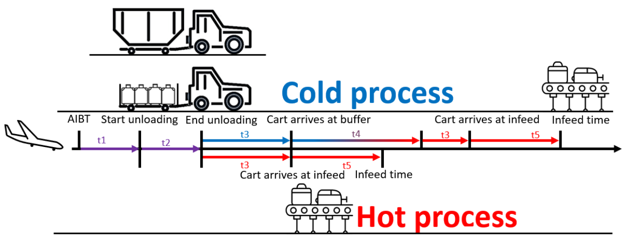


Fig. 6. transfer baggage journey from airplane to infeed with timesteps

The first step is the from actual in block time (AIBT) until the moment unloading can begin. This time is defined as t1. The AIBT is the moment an airplane has arrived at the apron and the blocks are placed under the wheels to prohibit any movement. t1 is chosen from a uniform distribution between 120 and 180 seconds. The total unloading time is t2. t2 is depended on the amount of bags in an airplane and if these are in bulk or containers. Unloading time for bulk baggage is chosen from a uniform distribution between 6 and 7.5 seconds per bag and the unloading time of a containers is chosen from a triangular distribution between 40 and 100 seconds with a mode at 60 seconds. After unloading two different processes start, either the cold process or the hot process. In the cold process a tug picks up the carts or containers and drives them to a buffer in driving time t3. Then the bags are buffered for the chosen buffer time t4. After that the bags must go to the infeed points, this takes some driving time t3 again. The last step is the infeed process, the time this takes is defined by t5. The infeed times are based on the machine capacity. It takes around 3.6 seconds per bag. The hot process skips the buffer and drives directly to the infeed points.

To build the simulation model all the elements in the mentioned process must be made. These elements are airplanes, carts, containers, tugs, buffer storages and infeed points.All these elements have their own attributes which contain the time steps or capacity and all the elements have their own process description language (PDL).

B. Experiments

From the research in to the design options two configurations were chosen, the first configuration is a buffer process that has a transition time of three hours, normal transport and one buffer location where carts and containers are parked. The second configuration has a transition time of three hours, autonomous tug transportation and three buffer locations where carts and containers can be parked. Next to the normal input three other possible future scenarios were devised that could be used as input. One scenario saw an increase in arriving flights during the morning peak hours, the second scenario saw an increase in bags to individual infeed locations because another infeed location would become unavailable. The last scenario saw the same amount of arriving flights but an increase in containers over bulk baggage. All scenarios were combined with the configurations as well as configuration without a buffer. This resulted in 12 different experiments.

VI. Results

In this section of the report the verification ,validation and the results of the experiments are shown.

A. Verification

In order to verify the model four tests were run. In these tests the parameters of the model were changed, to goal was to see if the effect of the change matched expected change. The results of the test are shown in Figure 7. The blue line is the baseline, the verification is done on experiment 1. This was the combination of normal input data and no buffer.

The first test is to multiply the driving times by a 100. The expected effect of this change is that baggage will arrive later at the infeed points. In Figure 7 test 1 is shown with an orange line and has moved to the right, most bags will be fed in after midday and a lot of bags are fed in a day later, which is not shown in this graph. The second test

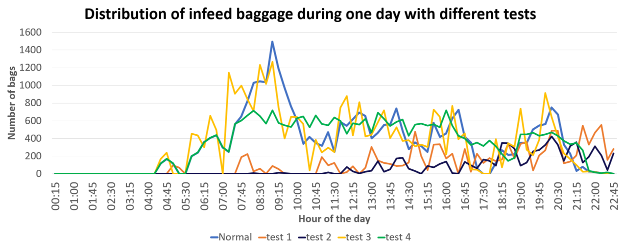


Fig. 7. Different verification tests ran on the model

was kind of similar to first test but this time the unloading times of the bags from an airplane were multiplied by a 100. This should yield a similar effect to test 1. The result of test 2 is shown in black in Figure 7, the line is also shifted to the right. Test 3 was to see if changing times to something lower would also work. The infeed times at the infeed stations were set to 0. The expected effect is that bags are fed in earlier and that the graphs shifts to the left. The effect is shown with the yellow line in Figure 7, it has indeed shifted to the left. The last test, test 4 has to do with the available amount of tugs. If the available amount of tugs is lower than the amount needed it is expected that bags will have to wait on the apron and thus be fed in later in to the system. The total amount of available tugs was divided by 4. The result of this simulation is shown with the green line in Figure 7, bags are fed in later during the day compared to the baseline and the amount of bags fed in per quarter hour remains similar during the day because this is now limited by the capacity of the tugs.

Randomness is introduced in the model by the use distributions, in order for experiments to be reproducible the seed is constant but this can have an effect on the outcome. The model was run 15 times in order to see what the effect of the random seed is and how many replications should be done. The outcome that was test was the amount of bags that were fed in between 7:30 and 9:30 AM. This is the mean in each case. Based on those means the standard deviation and 95% interval are calculated. These are all plotted in Figure 8. Based on the result it was chosen to do 13 replications of each experiment to minimize the effect of the random seed but without increasing the modelling time too much.

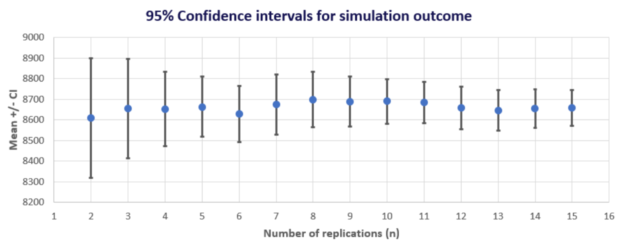


Fig. 8. Testing sensitivity to randomness with several replications

B. Validation

Because the real world infeed times of the bags were available it was possible to validate the model without a buffer. All the days were simulated and the average was taken. The average output is plotted against the real world data average in Figure 9. What can be observed is that trends are the same and the peaks are at the same timestamps. In order to quantify the performance of the model three calculations were made. The first one was to check the absolute difference between the model output and the data per time interval. The absolute difference was averaged over all the time intervals and result was 49 bags. The second calculation was to look at the relative

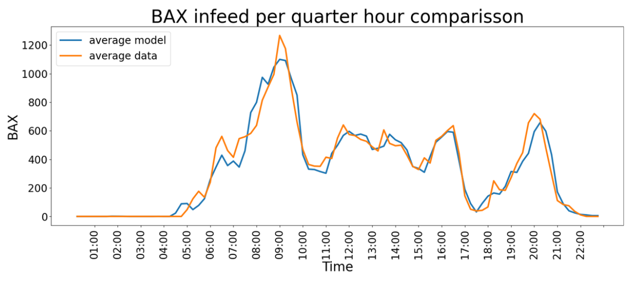


Fig. 9. Comparison of simulation result with real world data

difference by dividing the calculated difference of the first step by the actual amount of bags from the real world data. This relative difference was calculated and the result was 18.7.%. This results is a bit skewed. The third calculation was to look at difference between the output of the model and the real world data but divided by the average number of bags that are fed into the system. The average of this result was calculated and the outcome was 11.5%. Based on this outcome it was chosen that the simulation presented the real world process well enough to start running the experiments.

C. Results of the experiments

In section V all the experiments were explained. Experiment 1 was a combination of configuration 0, nu buffer and scenario 0, the normal data set. The highest peak in this experiment was 1261 bags per quarter hour. This is set as the baseline number to compare the experiments in this scenario with different configurations. Experiment 2 was the combination of configuration 1 and scenario 0. The peak is set 978 bags, which were fed in between 10:30 and 10:45AM. The peak reduction is calculated by subtracting the the number of bags in the old peaks from the numbers of bags in the new peak and this number is than divided by the number of bags in the old peak. In this case the outcome is 22.4% peak reduction. Experiment 2 was the combination of configuration 2 and scenario 0. The results of this experiment is shown in Figure 10.

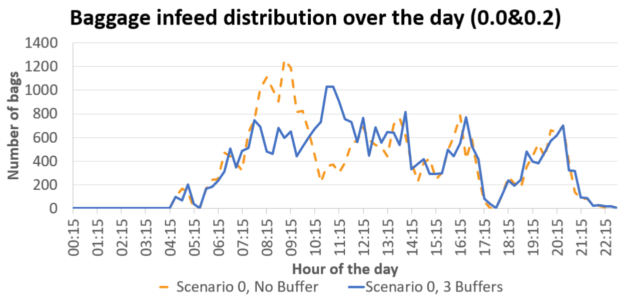


Fig. 10. Result experiment 3

This shows the infeed distribution of the whole system. The maximum peak here is 1032 at 10:30 and 10:45AM. At this moment the system uses 45.4% of the total system capacity. The peak reduction was 18.2%. The data is also plotted per infeed station in Figure 11. In the figures the orange dotted line is the baseline experiment and the blue line is the outcome of the simulation with a buffer. Experiment 4 was the combination of configuration 0 and scenario 1. This was a new baseline experiment because the input data changed. There was an increase in incoming flights. Experiment 5 was the combination of configuration 1 and scenario 1. The result of this experiment is shown in Figure 12. The peak of experiment 5 is 917 bags per quarter hour. The peak reduction was 34.9%. Because

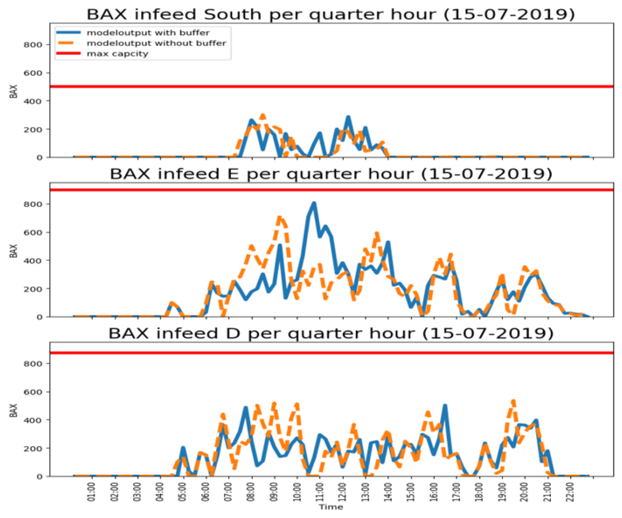


Fig. 11. Result experiment 3 infeed point South

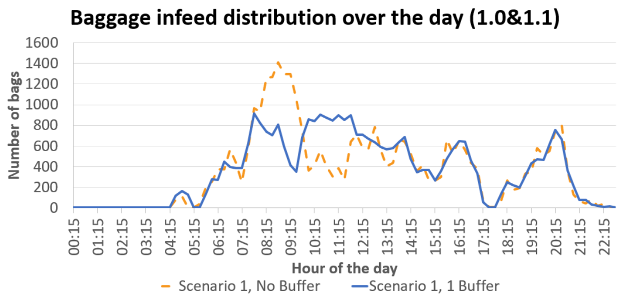


Fig. 12. Result experiment 5

the cold bags are buffered at South infeed hall this is were all the bags will be fed in. In Figure 13 the distribution of bags at the South infeed station can be seen. The infeed station is working at full capacity to feed in all the bags.

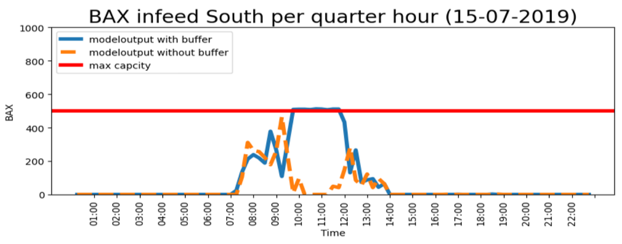


Fig. 13. Result experiment 5 infeed point South

As said in the modelling section of this paper, the last output is the amount of tugs is takes transport the cold bags. These are called the cold tugs in Table II. The results of the other experiments can be found in Table II.

The results of peak reduction are visualized in Figure 14. The maximum peak is shown for each experiment clustered per scenario. In the bars the actual peak reduction per is shown of each configuration in that scenario.

	Configuration 0: No buffer	**Configuration 1: Normal tugs 1 buffer**	**Configuration 2: Autonomous tugs 3 buffers**
Scenario 0: Normal arrival	Peak: 1261 Tugs: 12 Cold Tugs: 0	Peak: 978 Tugs:10 Cold Tugs: 29	Peak: 1032 Tugs: 10 Cold Tugs: 30
Scenario 1: More flights in peak hours	Peak: 1409 Tugs: 13 Cold Tugs: 0	Peak: 917 Tugs: 12 Cold Tugs: 32	Peak: 963 Tugs: 12 Cold Tugs: 34
Scenario 2: Less infeed capacity	Peak: 1254 Tugs: 12 Cold Tugs: 0	Peak: 839 Tugs: 10 Cold Tugs: 29	Peak: 895 Tugs: 10 Cold Tugs: 30
Scenario 3: More containers	Peak: 1216 Tugs: 12 Cold Tugs: 0	Peak: 873 Tugs: 9 Cold Tugs: 28	Peak: 848 Tugs: 9 Cold Tugs: 29

TABLE II

EFFECT OF EACH EXPERIMENT ON EVERY SCENARIO

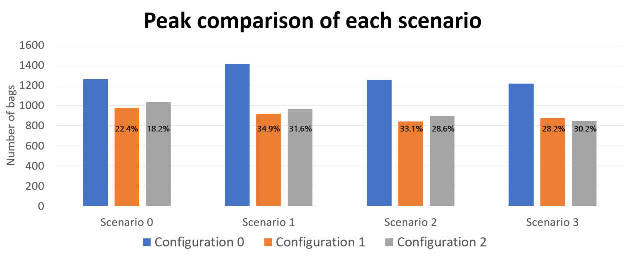


Fig. 14. Overview of the peaks of each configuration per scenario compared to each other

D. Cost Analysis

The costs to set up a cold buffer process differ per configuration. In configuration 1 there is one buffer and there transport is done manually. Handlers are needed at the buffer location to make sure everything is done correctly and drivers are needed to drive the tugs. Because the configurations uses a simple set up so this does not require additional costs. The cost for a driver is on average €80 per hour. The cost for a handler is on average €65 per hour. The difference in cost is due the depreciation costs of the tugs, which is included in the total driver costs. All the personnel has to work a minimum of three hours per shift. In the trial KLM ran on cold buffering they needed four drivers and one handler at the storage location to process the cold bags from fifteen airplanes, all the baggage was containerized. Based on this number of drivers and the number of cold tugs that were found out in the simulation results the total number of drivers can be derived for each chosen strategy. In the second configuration autonomous tugs were used. There are several suppliers of autonomous tugs. Two companies were approached and the cost of a autonomous tug at company 1 was €100.000 and at the second company €225.000. It is not known what the deprecation period of an autonomous tug is. The legal depreciation maximum is 20%. Based on this 20% a cost per day is calculated of €56 for the first company and €125 for the second. The are other costs involved with the usage of autonomous tugs such as maintenance and charging but these cost are not yet known and left out of scope.

VII. Discussion

All the results from the experiments were given in section VI, in this chapter the results are further discussed and insights will be drawn from them.

A numerical result of all the experiments was presented in Table II. What can be observed here is that all the experiments where cold buffering took place there was a significant peak reduction. The minimum is 18.2% in experiment 3 and the maximum is 34.9% in experiment 5. Configuration 1 has an average peak reduction of 29.7% and configuration 2 has an average peak reduction of 27.2%. In section III the potential was calculated to be 27%. This calculation did not take any minimal batch sizes of bags in to account. The reason the peak reduction is on average higher in configuration 1 compared to configuration 2 is because in configuration all the cold bags are buffered and then brought to the South infeed hall, during the simulation without a buffer the South infeed hall had almost no infeed between 9:30 and 11:30 giving it more space to feed in bags. In configuration 2 all three infeed points were used but because most bags were predetermined to go to the E-hall this is also were most of the cold bags were fed in. During the simulation without a buffer the E-hall was still quite busy with the infeed of normal transfer baggage between 9:30 and 11:30. Especially is scenario 2 this became a limitation because the D-hall was out of order in that scenario and E-hall became even busier. The difference between configuration 1 and 2 is the biggest here of all the scenarios.

Scenario 1 had an increase in flights that arrive during the hours of 7:00 and 9:00 and without buffering this would increase the morning peak from 1261 to 1409 bags per quarter hour. This is an increase of 11.7% at the busiest moment. With buffering the morning peaks of this scenario were comparable to the other scenarios where there was no increase in number of flights. Scenario 3 had a lower peak reduction compared to scenario 1 and 2. This is most likely due to the minimal limit of cold bags an airplane should contain before it is qualified for the hot and cold baggage separation. This limit was 5 bags for bulk baggage and 25 bags for containerized baggage. Because the limit for containerized flights is higher less flights will qualify in this scenario and thus less baggage will be separated and buffered.

Configuration 1 had a higher peak reduction on average compared to configuration 2 but the South infeed hall has to operate at maximum capacity for several hours in all the scenarios with configuration 1 in order to feed in all the buffered baggage. In Experiment 5 and 8, the combination of configuration 1 with scenario 1 and 2 respectfully, it can be seen that the width of the buffering period is wider than in experiment 2 and 11. This means that bags have to wait at the infeed point before being able to be fed in to the system. This might bring problems later on in the system with on time performance of the bags at the make-up stations. The amount of tugs that is needed to transport all the cold baggage does not significantly change between scenario 1 and 2. The number of tugs does change a lot from configuration 0, it seems that the simulation can make use of the tugs way more efficiently it this simulation. This is something that needs to be looked at in further research. Based on the total time individual infeed location work at maximum capacity, in order to process all the bags, configuration 2 would be the preferable solution in this aspect of operations.

VIII. Conclusion and recommendations

The aim of this research was to explore possible solutions to make more efficient use of the available capacity at airports. After the literature review one potential solution was chosen to explore further. This potential solution was called cold buffering. In order to explore this solution the baggage journey was studied. This gave an overview of the possible ways baggage could enter, leave or go through the baggage handling system at an airport. The subprocess of baggage handling on the airside was explored. This was important for the cold buffering process. In this process transfer baggage had to be separated based on transfer times. If the transfer time is longer that the set transition time the bag is classified as cold, if the transfer time is lower the bag is classified as hot. The cold baggage can be fed in to the system later because their connecting flights are leaving later than those of the hot baggage. This should lower the morning peak at the infeed stations. In order to test this idea a model had to made, several modelling techniques were researched but the decision was made to create a simulation model. Different design options were explored to be used as configuration in the model. Three configurations and four possible input scenarios were chosen which gave 12 experiments to test in the simulation model. All the experiments were simulated. The outcomes of the new infeed times were grouped together in time slots of 15 minutes. Based on the configuration without a buffer a morning peak was found for each scenario. After that the new peak was found with the two configurations for each scenario and based on the old and the peak a peak reduction was calculated. The peak reduction on average for the experiments with configuration one were 29.7% and with configuration two 27.2%. In configuration 1 it could be seen that the South infeed hall had to work at maximum capacity for a couple of hours to process all the cold baggage that had to be fed in there after the buffer period. In configuration 2, where three buffer locations were used, the individual infeed points were not working at full capacity if all were in use.

The recommendation for implementation is to implement a cold buffer process that can still feed in baggage at all the infeed points. To transport all the cold baggage the recommendation is to use autonomous tugs, they drive a bit slower but that just means less time in the buffer storage for cold baggage. Another advantage of using autonomous tugs is less personnel, which there is a shortage of at the moment. The cold bags can be stored for a shorter time because a trough can be seen in the infeed distribution between the morning peak and the new buffer peak. This could also mean that the transition time could be lowered by the same amount to increase the number of bags that would be classified cold. But at a couple of experiments the buffer peak was already higher the highest point in the figure so this would not attribute to peak shaving.

The recommendation for further research is to explore the other design configurations in the simulation model. Now that the model is set up other configurations can be tested easily. Future research in this topic of cold buffering would be really interesting, there is a lot of optimization possible within the configurations. This is something an optimization model would be perfect for. In order to do this all the constraints must be known but these can be figured out with the current simulation model. Next to that it would also be possible to apply parameter tuning on the current model to see if it can become more realistic.

References

Bardinas, F., Caguioa, J., Cariño, S., Saguid, W., Tonga, M., & Pineda, E. (2019). Airport management system: Core transaction 2 (iata boarding pass printing/iata bag tag printing, departure control, arrival control, interactive seat maps). *Ascendens Asia Singapore – Bestlink College of the Philippines Journal of Multidisciplinary Research.*

Barros, A. D., & Tomber, D. (2007). Quantitative analysis of passenger and baggage security screening at airports. *Journal of Advanced Transportation*, *41*, 171–193. https://doi.org/10.1002/atr.5670410204

Boute, S. (2016). *A future baggage reclaim innovating around the passenger at the a-area.*

Cavada, J. P., Cortés, C. E., & Rey, P. A. (2017). A simulation approach to modelling baggage handling systems at an international airport. *Simulation Modelling Practice and Theory*, *75*, 146–164. https://doi.org/10.1016/j.simpat.2017.01.006

Ganzewinkel, S. H. J. V. (2021). *Optimizing the baggage handling process for flight handling at schiphol airport.*

Hafilah, D., Radja, A., Rakoto-Ravalontsalama, N., Lafdail, Y., Hafilah, D. L., Radja, A. C., & Rakoto, N. (2017). *Modeling and simulation of baggage handling system in a large airport. the 18th asia pacific industrial engineering and management system conference*. https://hal.archives-ouvertes.fr/hal-01686750

Huang, E., Liu, I., & Lin, J. T. (2018). Robust model for the assignment of outgoing flights on airport baggage unloading areas. *Transportation Research Part E: Logistics and Transportation Review*, *115*, 110–125. https://doi.org/10.1016/j.tre.2018.04.012

Huang, E., Mital, P., Goetschalckx, M., & Wu, K. (2016). Optimal assignment of airport baggage unloading zones to outgoing flights. *Transportation Research Part E: Logistics and Transportation Review*, *94*, 110–122. https://doi.org/10.1016/j.tre.2016.07.012

Jacobs, S. (2021). *Challenges regarding the implementation of an autonomous baggage tractor on amsterdam airport schiphol.*

Lee, S., & Seo, S.-W. (2022). Sensor fusion for aircraft detection at airport ramps using conditional random fields. *IEEE Transactions on Intelligent Transportation Systems*, 1–13. https://doi.org/10.1109/TITS.2022.3157809

Ma, Q., Bi, J., Sai, Q., & Li, Z. (2021). Research on prediction of checked-baggage departed from airport terminal based on time series analysis. *Proceedings - 2021 7th Annual International Conference on Network and Information Systems for Computers, ICNISC 2021*, 264–269. https://doi.org/10.1109/ICNISC54316.2021.00055

Parlar, M., & Sharafali, M. (2008). Dynamic allocation of airline check-in counters: A queueing optimization approach. *Management Science*, *54*, 1410–1424. https://doi.org/10.1287/mnsc.1070.0842

Royal-Schiphol-Group. (2019). Bagage op schiphol.

Tarau, A. N., Schutter, B. D., & Hellendoorn, J. (2009). *Centralized, decentralized, and distributed model predictive control for route choice in automated baggage handling systems* (3).

van der Lande, A. (2011). *An analysis of the influences on the operational performance of klm's baggage turnaround services.*

van Dalen, P. (2010). *Capaciteitsanalyse van de e-kelder en optimalisatie van een buffer management systeem (e-kelder early baggage storage system amsterdam airport schiphol).* TU Delft.

van Wieren, L. (2021). *Autonomous technologies for baggage handling an exploratory research on the baggage handling capacity challenges at amsterdam airport schiphol and the added value of autonomous technologies.*

van Wingerden, A. (2022). *A baas ecosystem for aas.*

Vos, Z. (2019). Trustworthy home pick-up service.

B Python code

```python
# Databricks notebook source
# MAGIC %md
# MAGIC # 1. Import modules

# COMMAND ----------

# # Install packages
#%pip install plotly==4.12 # we want to use fig.add_hline(y=0.9)
#%pip install cufflinks
%pip install pandas==1.1.5 #https://github.com/pandas-dev/pandas/issues/38286
%pip install openpyxl
%pip install image
%pip install salabim
%pip install matplotlib pillow


# COMMAND ----------

# import main libraries
import datetime as dt
from datetime import datetime, timedelta
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
import plotly.io as pio
import plotly.subplots as sp
import pyspark
from pyspark.sql import functions as sf
from pyspark.sql import Window
import re
import datetime as dt
import numpy as np
import matplotlib.pyplot as plt
import math
import random

# COMMAND ----------

# MAGIC %md
# MAGIC #2. load data

# COMMAND ----------

df =
↪  pd.read_excel("/dbfs/FileStore/shared_uploads/niek.grift@schiphol.nl/finaldf-2.xlsx",engine="openpyxl",)

# COMMAND ----------

# MAGIC %md
# MAGIC # 3. Generate temperature

# COMMAND ----------

df['SOBT_LT'] = pd.to_datetime(df['SOBT_LT'], format='%H:%M:%S')
#Put Scheduled of block time in right format
df['SIBT_LT'] = pd.to_datetime(df['SIBT_LT'], format='%H:%M:%S')
#Put Scheduled in block time in right format
df['AIBT_LT'] = pd.to_datetime(df['AIBT_LT'], format='%H:%M:%S')
#Put Actual in block time in right format
df['TIME_OF_ENTRY_LT'] = pd.to_datetime(df['TIME_OF_ENTRY_LT'], format='%H:%M:%S')
```

```
#Put System entry time in right format
df['SOBT_DAY']= pd.to_datetime(df['SOBT_DAY'], format='%m-%d-%Y')
#Put Scheduled of block time time on the right day
df['SIBT_DAY']= pd.to_datetime(df['SIBT_DAY'], format='%m-%d-%Y')
#Put Scheduled in block time time on the right day
df['dif_day'] = df['SOBT_DAY']-df['SIBT_DAY']
#Find the difference in days between arriving and departure of baggage
df['TRF_time']= df['SOBT_LT']-df['SIBT_LT']+df['dif_day']
#Find the time difference between arriving and departure of baggage compensated for the day

s = df['SIBT_LT'][0]
S = s.hour*3600
print(s,S)
Temperature = '3:00:00'
#Border time between hot and cold baggage
Peak_start  = '1900-01-01 7:00:00'
Peak_end    = '1900-01-01 9:00:00'

def datetime_range(start, end, delta):
    #Function creates a a data set for 0:00 till 23:00 with intervals of 15 minutes
    current = start
    while current < end:
        yield current
        current += delta
#set up of fucntion including days, months and year
dts = [dt.strftime('%Y-%m-%d %H:%M:%S') for dt in
       datetime_range(datetime(1900, 1, 1, 0), datetime(1900, 1, 1, 23),
       timedelta(minutes=15))]
#set up of function with just hours and minutes
dts2 = [dt.strftime('%H:%M') for dt in
       datetime_range(datetime(1900, 1, 1, 0), datetime(1900, 1, 1, 23),
       timedelta(minutes=15))]
dts_1 = [dt.strftime('%Y-%m-%d %H:%M:%S') for dt in
       datetime_range(datetime(1900, 1, 1, 0), datetime(1900, 1, 1, 23),
       timedelta(minutes=5))]
dts2_1 = [dt.strftime('%H:%M') for dt in
       datetime_range(datetime(1900, 1, 1, 0), datetime(1900, 1, 1, 23),
       timedelta(minutes=5))]
dts3= [dt.strftime('%Y-%m-%d %H:%M:%S') for dt in
       datetime_range(datetime(1830, 1, 1, 0), datetime(1830, 1, 1, 23),
       timedelta(minutes=15))]
dts4 = [dt.strftime('%H:%M') for dt in
       datetime_range(datetime(1830, 1, 1, 0), datetime(1830, 1, 1, 23),
       timedelta(minutes=15))]
dts5= [dt.strftime('%Y-%m-%d %H:%M:%S') for dt in
       datetime_range(datetime(1970, 1, 1, 0), datetime(1970, 1, 1, 23),
       timedelta(minutes=15))]
dts6 = [dt.strftime('%H:%M') for dt in
       datetime_range(datetime(1970, 1, 1, 0), datetime(1970, 1, 1, 23),
       timedelta(minutes=15))]
dts5_1= [dt.strftime('%Y-%m-%d %H:%M:%S') for dt in
       datetime_range(datetime(1970, 1, 1, 0), datetime(1970, 1, 1, 23),
       timedelta(minutes=5))]
dts6_1 = [dt.strftime('%H:%M') for dt in
       datetime_range(datetime(1970, 1, 1, 0), datetime(1970, 1, 1, 23),
       timedelta(minutes=5))]
#Turn data set in to pandas dataframe
dts=pd.to_datetime(dts, format = '%Y-%m-%d %H:%M:%S')
dts3=pd.to_datetime(dts3, format = '%Y-%m-%d %H:%M:%S')
dts5=pd.to_datetime(dts5, format = '%Y-%m-%d %H:%M:%S')
dts_1=pd.to_datetime(dts_1, format = '%Y-%m-%d %H:%M:%S')
dts5_1=pd.to_datetime(dts5_1, format = '%Y-%m-%d %H:%M:%S')
#Delete first value of dataset to match number of bins later in plot
dts2=dts2[1:92]
dts4=dts4[1:92]
dts6=dts6[1:92]
dts2_1=dts2_1[1:276]
dts6_1=dts6_1[1:276]
#Add temperature column with hot if TRF is smaller than the border time
df.loc[df['TRF_time'] <= Temperature, 'temp'] = 'hot'
df.loc[df['TRF_time'] > Temperature, 'temp'] = 'cold'

df['BO'] = np.where(df['AIRCRAFT_ARRIVAL_ciss']!= "E90", 'WIBO', 'NABO')
#Add bodytype column whit NABO if aircrafttype is E90 else is WIBO
df.loc[df['AIRCRAFT_ARRIVAL_ciss'] == "E75", 'BO'] = 'NABO'
```

```
#Replace wibo with nabo if the aircraftype is E75
df.loc[df['AIRCRAFT_ARRIVAL_ciss'] == "null", 'BO'] = 'null'
#Replace wibo with null if the aircraftype is null
df.loc[df['AIRCRAFT_ARRIVAL_ciss'] == "73W", 'BO'] = 'NABO'
#Replace wibo with nabo if the aircraftype is 73W
df.loc[df['AIRCRAFT_ARRIVAL_ciss'] == "73H", 'BO'] = 'NABO'
#Replace wibo with nabo if the aircraftype is 73H
df.loc[df['AIRCRAFT_ARRIVAL_ciss'] == "73J", 'BO'] = 'NABO'
#Replace wibo with nabo if the aircraftype is 73J
df.loc[df['AIRCRAFT_ARRIVAL_ciss'] == "E95", 'BO'] = 'NABO'
#Replace wibo with nabo if the aircraftype is E96
df.loc[df['AIRCRAFT_ARRIVAL_ciss'] == 321, 'BO'] = 'NABO'
#Replace wibo with nabo if the aircraftype is 321
count1 = df['BO'].value_counts()
count2 = df['temp'].value_counts()  #Count different entries in BO column
print(count1)
print(count2)


# COMMAND ----------

# MAGIC %md
# MAGIC # 4. Pick WIBO, NABO or full data set

# COMMAND ----------

#df = df[df['BO'] == "NABO"] #delete # to use NABO data set
#df = df[df['BO'] == "WIBO"] #delete # to use WIBO data set
#delete nothing to use full data set

# COMMAND ----------

# MAGIC %md
# MAGIC # 5. Creates count function and plot

# COMMAND ----------

df3 = df['TIME_OF_ENTRY_LT'].value_counts(bins = dts).sort_index()
#create dataframe that has the counts of the dataset per quarter hour
print(df3)
ax= plt.subplot()                                #Create plot of counted frame
plt.plot(dts2,df3)                               #Add data
plt.xlabel("Time")                               #Add X label
plt.ylabel("BAX")                                #Add Y label
plt.setp(ax.get_xticklabels(), rotation=90)      #Turn X data ticks
plt.xticks(np.arange(3, 92, 4))                  #Reduce ticks
plt.title("BAX infeed per quarter hour")         #add title
plt.show()

print('peak at', max(df3))





# COMMAND ----------

# MAGIC %md
# MAGIC #6. trying to buffer bags (all bags from the data set)

# COMMAND ----------

BufferTime = '2:00:00'                                      #Decide the time for how long the bags will
↪  be buffered
ZeroTime = '0:00:00'                                        #Zero time must be added to the hot bags in
↪  order to make sure the date is still the same (1830-01-01), see dts3
BufferTime=pd.to_datetime(BufferTime, format='%H:%M:%S')     #Set buffertime in correct pandas format so
↪  taht it's compattible with the data
ZeroTime=pd.to_datetime(ZeroTime, format='%H:%M:%S')         #Set zerotime in correct pandas format so
↪  taht it's compattible with the data
df['TIME_OF_ENTRY_LT2']=""                                  #Create extra column in df so that it can be
↪  used in the for loop
for i in range(0, len(df)):
    df['TIME_OF_ENTRY_LT2'][i]=
    ↪  pd.Timestamp((df['TIME_OF_ENTRY_LT'][i].asm8.astype(np.int64)+BufferTime.asm8.astype(np.int64)).astype('<M8[ns]'))
    ↪  if df['temp'][i]=='cold' else
    ↪  pd.Timestamp((df['TIME_OF_ENTRY_LT'][i].asm8.astype(np.int64)+ZeroTime.asm8.astype(np.int64)).astype('<M8[ns]'))
df['TIME_OF_ENTRY_LT2']= pd.to_datetime(df['TIME_OF_ENTRY_LT2'], format = '%Y-%m-%d %H:%M:%S')
#Explanation of the for loop
# line 6: The range is taken for the enitre length of the dataset to check every entry
# line 7: An if statement is put here, what is says is that the new time of entry LT2 is old time of entry
↪  + Buffertime if the temp is cold, otherwise it's the old time of entry + zero time
# All the .asm8.astype are added to first put the time in integers so that they can be added and then back
↪  to timedate pandas format

df4 = df['TIME_OF_ENTRY_LT2'].value_counts(bins = dts3).sort_index()#create dataframe that has the counts
↪  of the dataset per quarter hour
print(df4)
ax= plt.subplot()                                                  #Create plot of counted frame
plt.plot(dts4,df4, label='buffering')                              #Add data of buffering
plt.plot(dts2,df3, label='no buffering')                           #Add data of no buffering
plt.xlabel("Time")                                                 #Add X label
plt.ylabel("BAX")                                                  #Add Y label
plt.setp(ax.get_xticklabels(), rotation=90)                        #Turn X data ticks
plt.xticks(np.arange(3, 92, 4))                                    #Reduce ticks
plt.title("BAX infeed per quarter hour")                           #add title
plt.legend(loc="upper right")                                      #add legend
plt.show()
print('peak with no buffering=',max(df3))
print('peak with buffering=',max(df4))


# COMMAND ----------

# MAGIC %md
# MAGIC # 7. Look at the different infeed points

# COMMAND ----------

dfz= df.loc[df['ENTRY_HANDLING_AREA'].str.contains('Z').fillna(False)]  #create dataset that only contains
↪  bags that are entered at infeedpoint south
dfe= df.loc[df['ENTRY_HANDLING_AREA'].str.contains('E').fillna(False)]  #create dataset that only contains
↪  bags that are entered at infeedpoint E
dfd= df.loc[df['ENTRY_HANDLING_AREA'].str.contains('D').fillna(False)]  #create dataset that only contains
↪  bags that are entered at infeedpoint D
dfz3 = dfz['TIME_OF_ENTRY_LT'].value_counts(bins = dts).sort_index()    #create dataframe that has the
↪  counts of the dataset per quarter hour for the dataset of south
dfe3 = dfe['TIME_OF_ENTRY_LT'].value_counts(bins = dts).sort_index()    #create dataframe that has the
↪  counts of the dataset per quarter hour for the dataset of E
dfd3 = dfd['TIME_OF_ENTRY_LT'].value_counts(bins = dts).sort_index()    #create dataframe that has the
↪  counts of the dataset per quarter hour for the dataset of D
ax= plt.subplot()                                                       #Create plot of counted frame
plt.plot(dts2,dfz3, label="Z")                                          #Add south data
plt.plot(dts2,dfe3, label="E")                                          #Add E data
plt.plot(dts2,dfd3, label="D")                                          #Add D data
plt.xlabel("Time")                                                      #Add X label
plt.ylabel("BAX")                                                       #Add Y label
plt.setp(ax.get_xticklabels(), rotation=90)                             #Turn X data ticks
plt.xticks(np.arange(3, 92, 4))                                         #Reduce ticks
plt.title("BAX infeed per quarter hour")                                #Add title
plt.legend(loc="upper right")                                           #Add legend
plt.show()



# COMMAND ----------

# MAGIC %md
# MAGIC # 8. Creating a data det with morning peak data and create a data set with unique flight numbers

# COMMAND ----------

df_peak = df[df['SIBT_LT'] <= Peak_end]
df_peak = df_peak[df_peak['SIBT_LT'] >= Peak_start]
df_else1 = df[df['SIBT_LT'] < Peak_start]
df_else2 = df[df['SIBT_LT'] > Peak_end]
df_else = [df_else1,df_else2]
df_else = pd.concat(df_else)
```

```
#print(df_peak['SIBT_LT'])
count2 = df_peak['temp'].value_counts()  #Count different entries in BO column

#print(count2)
df_flights = df.drop_duplicates(subset = ['FLIGHT_DESIGNATOR_ARRIVAL'])
df_flights_peak = df_peak.drop_duplicates(subset = ['FLIGHT_DESIGNATOR_ARRIVAL'])
count1 = df_flights_peak['BO'].value_counts()
count2 = df_flights_peak['temp'].value_counts()  #Count different entries in BO column
#print(count1)
#print(count2)
#print(df_flights)
df_flightnumbers = df_flights['FLIGHT_DESIGNATOR_ARRIVAL']
df_flightnumbers=df_flightnumbers.reset_index(drop=True)
print(df_flightnumbers)

df7=[]
Count=[]
for i in range(0, len(df_flightnumbers)):
    df7 = df[df['FLIGHT_DESIGNATOR_ARRIVAL'] == df_flightnumbers[i] ]
    Count.append(df7['temp'].value_counts())

Count=pd.DataFrame(Count)
df_flightnumbers=pd.DataFrame(df_flightnumbers)
Count=Count.reset_index(drop=True)
df_flights=df_flights.reset_index(drop=True)
Count=Count.fillna(0)
print(Count)
#df8=[df_flightnumbers,Count]
df8 = df_flights.join(Count)
df8['total']=df8['cold']+df8['hot']
print(df8)

#CountHot = Count['hot']
#CountCold = Count['cold']
#print(df7['temp'])
#print(Count)
df4 = df_flights['SIBT_LT'].value_counts(bins = dts).sort_index()   #create dataframe that has the counts
↪  of the dataset per quarter hour
df5 = df_flights['AIBT_LT'].value_counts(bins = dts).sort_index()
ax= plt.subplot()                                                   #Create plot of counted frame
plt.plot(dts2,df4, label="SIBT")                                    #Add data
plt.plot(dts2,df5, label="AIBT")
plt.xlabel("Time")                                                  #Add X label
plt.ylabel("# of arrivals")                                         #Add Y label
plt.setp(ax.get_xticklabels(), rotation=90)                         #Turn X data ticks
plt.xticks(np.arange(3, 92, 4))                                     #Reduce ticks
plt.title("Planes landed per quarter hour")                         #add title
plt.legend(loc="upper left")
plt.show()

# COMMAND ----------

# MAGIC %md
# MAGIC #9. trying to buffer bags (on the peak of the data set)

# COMMAND ----------

BufferTimes = '2:00:00'                                       #Decide the time for how long the bags will
↪  be buffered, different to code in part 7 to make sure the format does not matter
ZeroTimes = '0:00:00'                                         #Zero time must be added to the hot bags in
↪  order to make sure the date is still the same (1830-01-01), see dts3
BufferTimes=pd.to_datetime(BufferTimes, format='%H:%M:%S')    #Set buffertime in correct pandas format so
↪  taht it's compattible with the data
ZeroTimes=pd.to_datetime(ZeroTimes, format='%H:%M:%S')        #Set buffertime in correct pandas format so
↪  taht it's compattible with the data
df_peak['TIME_OF_ENTRY_LT2']=""                               #Create extra column in df so that it can be
↪  used in the for loop
df_peak = df_peak.reset_index(drop=True)                      #Data was filtered out but pandas leaves the
↪  index the same, them it wont work in the for loop so the index is reset from 0 and the old index is
↪  dropped

for i in range(0, len(df_peak)):
        df_peak['TIME_OF_ENTRY_LT2'][i]=
        ↪  pd.Timestamp((df_peak['TIME_OF_ENTRY_LT'][i].asm8.astype(np.int64)+BufferTimes.asm8.astype(np.int64)).astype('<M8
        ↪  if df_peak['temp'][i]=='cold' else
        ↪  pd.Timestamp((df_peak['TIME_OF_ENTRY_LT'][i].asm8.astype(np.int64)+ZeroTimes.asm8.astype(np.int64)).astype('<M8[
#Explanation of the for loop
# line 8: The range is taken for the enitre length of the dataset to check every entry
# line 9: An if statement is put here, what is says is that the new time of entry LT2 is old time of entry
↪  + Buffertime if the temp is cold, otherwise it's the old time of entry + zero time
# All the .asm8.astype are added to first put the time in integers so that they can be added and then back
↪  to timedate pandas format
df_else=df_else.reset_index(drop=True)                          #Data was filtered out but pandas leaves the
↪  index the same, them it wont work in the for loop so the index is reset from 0 and the old index is
↪  dropped
for i in range(0, len(df_else)):

    ↪  df_else['TIME_OF_ENTRY_LT2'][i]=pd.Timestamp((df_else['TIME_OF_ENTRY_LT'][i].asm8.astype(np.int64)+ZeroTimes.asm8.ast
# In this for loop all the baggage outside the peak will not be buffered but the date is different so
↪  thats why zerotimes is also added here so that every data entry is on the same day again

df_new = [df_else,df_peak]       #Combine the data set of the peak and the data set outside the peak
df_new = pd.concat(df_new)       #Combine the data set of the peak and the data set outside the peak
#print(df_new)

df_new['TIME_OF_ENTRY_LT2']= pd.to_datetime(df_new['TIME_OF_ENTRY_LT2'], format = '%Y-%m-%d %H:%M:%S')
df6 = df_new['TIME_OF_ENTRY_LT2'].value_counts(bins = dts3).sort_index()      #create dataframe that has
↪  the counts of the dataset per quarter hour
ax= plt.subplot()                                                             #Create plot of counted
↪  frame
plt.plot(dts4,df6, label='buffering')                                         #Add data
plt.plot(dts2,df3, label='no buffering',linestyle='dashed')                   #Add data
plt.xlabel("Time")                                                            #Add X label
plt.ylabel("BAX")                                                             #Add Y label
plt.setp(ax.get_xticklabels(), rotation=90)                                   #Turn X data ticks
plt.xticks(np.arange(3, 92, 4))                                               #Reduce ticks
plt.legend(loc="upper left")                                                  #Add legend
plt.title("BAX infeed per quarter hour")                                      #add title
plt.show()
print('peak with no buffering=',max(df3))
print('peak with buffering=',max(df6))

# COMMAND ----------

# MAGIC %md
# MAGIC # 8. Salabim

# COMMAND ----------

DrivingTimes=
↪  pd.read_excel("/dbfs/FileStore/shared_uploads/niek.grift@schiphol.nl/DrivingTimes.xlsx",engine="openpyxl",)
↪  #Open data of the driving times from source
DrivingTimes = pd.DataFrame(DrivingTimes)
↪  #Make data a panda dataframe
SpeedFactor = 1
DrivingTimes =SpeedFactor*DrivingTimes
data= df8

#pd.read_excel("/dbfs/FileStore/shared_uploads/niek.grift@schiphol.nl/testdata2.xlsx",engine="openpyxl",)
↪  #Open data from source

data = pd.DataFrame(data)
↪  #Make data a panda dataframe
data['AIBT_LT'] = pd.to_datetime( data['AIBT_LT'], format='%Y-%m-%d %H:%M:%S')
↪  #Set landing to correct time format to work with in pandas
data['AIBT_LT'] = data['AIBT_LT'].dt.strftime('%H:%M:%S')
data['AIBT_LT'] = pd.to_datetime( data['AIBT_LT'], format='%H:%M:%S')
↪  #Set landing to correct time format to work with in pandas
data['AIBT_S'] = pd.to_timedelta(data['AIBT_LT'].dt.time.astype(str)).dt.total_seconds()
↪  #Convert landing time to seconds to work with this in pandas enviroment

data = data[data['ENTRY_HANDLING_AREA'] != 'C']
data = data[data['ENTRY_HANDLING_AREA'] != 'W']
count = data['ENTRY_HANDLING_AREA'].value_counts()
print(count)
#parameters
NRNABO = sum(data.BO =='NABO')                              #Determine number of NABO planes in dataset
```

```
NRWIBO = sum(data.BO =='WIBO')                          #Determine number of WIBO planes in dataset
NRplanes =len(data)                                     #Determine total number of planes in dataset
data['BO'] = data['BO'].map({'WIBO': 1, 'NABO': 0})     #Give WIBO flights a 1 and NOBO flights a 0 in
↪  order to use this data
data['GATE_AREA']=data['GATE_AREA'].map({'A gebied +
↪  B-platform':0,'B-pier':1,'C-steel':2,'C-kop':3,'D-steel':4,'D-vorkzuid':5,'D-vorknoord':6,'E-steel':7,'E-kop':8,'F-pier':
#Give every landing area a number so that becomes a callabel object
data['temp'] = data['temp'].map({'hot': 1, 'cold': 0})  #Give every temperature a number so that becomes
↪  a callabel object


data['ENTRY_HANDLING_AREA']=data['ENTRY_HANDLING_AREA'].map({'D':1,'E':0,'Z':2}) #Give every infeed area a
↪  number so that becomes a callabel object
data = data.sort_values('AIBT_S')                       #Sort the data from earliest flight to the latest
data = data.reset_index(drop=True)                      #Reorder the index

I = range(0, (NRplanes-2)   )                               #Create range for number of planes
J = range(0, 100)                                    #Create range for the nuber of Nabo planes
K = range(0, 5)                                    #Create a range for the number of Wibo planes
M = range(0, NRNABO)
N = range(0, NRWIBO)
print(sum(data['total']))
DataE = data[data['ENTRY_HANDLING_AREA'] == 0]
DataD = data[data['ENTRY_HANDLING_AREA'] == 1]
DataZ = data[data['ENTRY_HANDLING_AREA'] == 2]
print('E=',sum(DataE['total']),'D=',sum(DataD['total']),'Z=',sum(DataZ['total']))
print(sum(data['cold']))
data_peak = data[data['AIBT_LT'] <= Peak_end]
data_peak = data_peak[data_peak['AIBT_LT'] >= Peak_start]
data_peak = data_peak[data_peak['cold'] >= 25]
print(sum(data_peak['cold']))
count1 = data['BO'].value_counts()
print(count1)

# COMMAND ----------

# MAGIC %md
# MAGIC # 9. Simulation with buffer

# COMMAND ----------

import salabim as sim

#airplanes
#IAT = 300      #mean Interarivaltimes between airplanes
T1LB = 120      #Time between In block time and first bag/container removed lowerbound
T1UB = 180      #Time between In block time and first bag/container removed upperboundbound
T2NALB = 6      #Lowerbound
T2NAUB = 7.5
T2WLB= 55
T2WUB= 115
T2WMode= 75
BagLB = 1       #Lowerbound Amount of bags in airplane
BagUB = 80      #Upperbound Amount of bags in airplane


#Infeedstations
InfeedsE = 4
CapacityE = 900

InfeedsD = 5
CapacityD = 900

InfeedsZ = 2
CapacityZ = 1000

Loadingtime = 3.53
LoadingtimeD = 4
LoadingtimeE = 4
LoadingtimeZ = 3.6

Drivingtime = 30

#Carts
NumberofCarts = NRNABO
```

```
CartLB = 20
CartUB = 40
CartMode = 30

#Tugs
NumberofTugs = 100
NumberofColdTugs=100
#Containers
NumberofContainers = NRWIBO
ContLB = 30
ContUB = 42
ContMode = 38

#Buffer
NumberofBuffers = 100
Peakstarthours = 7
Peakstartseconds = Peakstarthours*3600
Peakendhours = 9
Peakendseconds = Peakendhours*3600
BufferTime = 7200

#output
WaitingTime=[]
UnloadingTime=[]
Cartss = []
Containerss=[]
DrivingTime=[]
InfeedTimeE=[]
BagsE=[]
InfeedTimeD=[]
BagsD=[]
InfeedTimeZ=[]
BagsZ=[]
Eentryarea=[]
Entry=[]
Bagsinplane=[]
Bagsinplane2=[]
Licenseplatemodel =[]
Arrivaltime=[]
Sum2=[]
Inter=[]
Inter2=[]
Inter3=[]
#===============================================================================================================

class TAirplaneGenerator(sim.Component):
    def setup(self, LB, UB, T1LB, T1UB ):
        self.ArrivalTime = np.zeros(NRplanes-1)
        self.ArrivalArea = np.zeros(NRplanes-1)
        #self.BagsPerPlane = sim.Uniform(LB, UB+1)
        self.BagsPerPlane =np.zeros(NRplanes-1)
        self.ColdBagsPerPlane =np.zeros(NRplanes-1)
        self.HotBagsPerPlane =np.zeros(NRplanes-1)
        self.T1 = sim.Uniform(T1LB, T1UB)
        self.BodyType = np.zeros(NRplanes-1)
        self.Temp = np.zeros(NRplanes-1)
        self.EntryArea = np.zeros(NRplanes-1)
        self.LicencePlate = np.zeros(NRplanes-1)
    def process(self):
        for i in I:
            InterarrivalTime = np.zeros(NRplanes-1)
            InterarrivalTime[i] = data['AIBT_S'][i+1]-data['AIBT_S'][i]
            #Inter.append(data['AIBT_S'][i+1]-data['AIBT_S'][i])
            #Inter2.append(self.InterarrivalTime[i])
            myIAT = InterarrivalTime[i]
            self.ArrivalTime[i] = data['AIBT_S'][i+1]

            self.ArrivalArea[i]=data['GATE_AREA'][i+1]
            self.EntryArea[i]=data['ENTRY_HANDLING_AREA'][i+1]
            self.LicencePlate[i]= data['LICENSE_PLATE_CODE'][i+1]
            LicensePlate=self.LicencePlate[i]
            EntryArea = self.EntryArea[i]
            ArrivalArea = self.ArrivalArea[i]
            ArrivalTime= self.ArrivalTime[i]
            self.Temp[i]= data['temp'][i+1]
            Temp= self.Temp[i]
            Temp_output = "cold" if  Temp==0 else "hot"
            self.BodyType[i]=data['BO'][i+1]
            self.BagsPerPlane[i] =data['total'][i+1]
            NrB = self.BagsPerPlane[i]
            print('bags on plane',NrB)
            self.ColdBagsPerPlane[i] =data['cold'][i+1]
            CNrB = self.ColdBagsPerPlane[i]
            print('cold bags on plane',CNrB)
            self.HotBagsPerPlane[i] =data['hot'][i+1]
            HNrB = self.HotBagsPerPlane[i]
            print('Hot bags on plane',HNrB)
            Bagsinplane2.append(NrB)
            #NrB = int(self.BagsPerPlane.sample())
            T1= self.T1.sample()

            BO= self.BodyType[i]
            BO_output = "WIBO" if  BO==1 else "NABO"
            print('BodyType=',BO_output)
            yield self.hold(till=data['AIBT_S'][i+1])
            NewAirplane = TAirplane(NrBag = NrB, CNrBag = CNrB, HNrBag = HNrB,T10 = T1, BoTy=BO, temp=Temp,
            ↪  EA=EntryArea, AA = ArrivalArea, LP=LicensePlate, AT=ArrivalTime)
            NewAirplane.enter(AllAirplanes)



 ↪  #==========================================================================================================
class TAirplane(sim.Component):
    def setup(self, NrBag,CNrBag,HNrBag, T10, BoTy, temp,EA, AA, LP, AT):
        self.BodyType1 = BoTy
        self.NrBags = NrBag
        self.CNrBags = CNrBag
        self.HNrBags = HNrBag
        self.T100 = T10

        self.Temperature = temp
        self.Entry_Area = EA
        self.Arrival_Area = AA
        self.LicensePlates = LP
        self.Arrivaltime = AT
    def process(self):
        FirstAirplane = AllAirplanes.pop()
        Inter3.append(env.now())
        yield self.hold(self.T100)
        WaitingTime.append(self.T100)

        #UnloadingTime.append(self.NrBags*self.T200N)
        Licenseplatemodel.append(self.LicensePlates)
        Bagsinplane.append(self.NrBags)
        Entry.append(self.Entry_Area)
        Arrivaltime.append(self.Arrivaltime)
        if self.BodyType1 == 0:
            ShortestLine = 0
            for i in range (1, NumberofCarts):
                if Carts[i].MyQueue.length () < Carts[ShortestLine].MyQueue.length ():
                    ShortestLine = i
            self.enter(Carts[ShortestLine].MyQueue)
            if Carts[ShortestLine].ispassive():
                Carts[ShortestLine].activate()
        if self.BodyType1 == 1:
            #yield self.hold(self.NrBags*self.T200N)
            #UnloadingTime.append(self.NrBags*self.T200N)
            ShortestLine = 0
            for i in range (1, NumberofContainers):
                if Containers[i].MyQueue.length () < Containers[ShortestLine].MyQueue.length ():
                    ShortestLine = i
            self.enter(Containers[ShortestLine].MyQueue)
            if Containers[ShortestLine].ispassive():
                Containers[ShortestLine].activate()
            ShortestLine = 0

#===================================================================================================================
class TCarts(sim.Component):
    def setup(self, CartLB, CartUB, CartMode, T2NALB, T2NAUB):
        self.BagsonCart = sim.Triangular(CartLB,CartUB,CartMode)
        self.MyQueue = sim.Queue (self.name () + '-WaitingLine')
        self.T2N = sim.Uniform(T2NALB,T2NAUB)
    def process(self):
        while True:
            BagsonCart = int(self.BagsonCart.sample())
            T2N= self.T2N.sample()
            while self.MyQueue.length() == 0:
                yield self.passivate()
            FirstAirplane = self.MyQueue.pop()
            self.NrBags=FirstAirplane.NrBags
            self.CNrBags=FirstAirplane.CNrBags
            print('cold bags in cart',self.CNrBags)
            self.HNrBags=FirstAirplane.HNrBags
            yield self.hold(self.NrBags*T2N)
            self.EntryArea = FirstAirplane.Entry_Area
            self.ArrivalArea = FirstAirplane.Arrival_Area
            self.tempcart = FirstAirplane.Temperature
            NRofCarts= math.ceil(FirstAirplane.NrBags/BagsonCart)
            Cartss.append(NRofCarts)
            #print('number of bags=',FirstAirplane.NrBags,'Bags fit on this cart',BagsonCart,' so Carts
            ↪  needed=',NRofCarts)
            print('time=', env.now(),'cold bags',self.CNrBags)
            self.pickup=FirstAirplane.Arrivaltime
            self.type = 0
            if Peakstartseconds <= self.pickup <= Peakendseconds:
                if self.CNrBags >= 5:
                    ShortestLine = 0
                    for i in range (1, NumberofColdTugs):
                        if ColdTug[i].MyQueue.length () < ColdTug[ShortestLine].MyQueue.length ():
                            ShortestLine = i
                    self.enter(ColdTug[ShortestLine].MyQueue)
                    if ColdTug[ShortestLine].ispassive():
                        ColdTug[ShortestLine].activate()

                    ShortestLine = 0
                    for i in range (1, NumberofTugs):
                        if Tug[i].MyQueue.length () < Tug[ShortestLine].MyQueue.length ():
                            ShortestLine = i
                    self.enter(Tug[ShortestLine].MyQueue)
                    if Tug[ShortestLine].ispassive():
                        Tug[ShortestLine].activate()
                    yield self.passivate
                else:
                    print('iets')
                    ShortestLine = 0
                    for i in range (1, NumberofTugs):
                        if Tug[i].MyQueue.length () < Tug[ShortestLine].MyQueue.length ():
                            ShortestLine = i
                    self.enter(Tug[ShortestLine].MyQueue)
                    if Tug[ShortestLine].ispassive():
                        Tug[ShortestLine].activate()
                    yield self.passivate
            else:
                print('time is above 10000')
                ShortestLine = 0
                for i in range (1, NumberofTugs):
                    if Tug[i].MyQueue.length () < Tug[ShortestLine].MyQueue.length ():
                        ShortestLine = i
                self.enter(Tug[ShortestLine].MyQueue)
                if Tug[ShortestLine].ispassive():
                    Tug[ShortestLine].activate()
                yield self.passivate
#====================================================================================================================
class TContainers(sim.Component):
    def setup(self, ContLB, ContUB, ContMode, T2WLB,T2WUB,T2WMode):
        self.BagsonCont = sim.Triangular(ContLB,ContUB,ContMode)
        self.T2W = sim.Triangular(T2WLB,T2WUB,T2WMode)
        self.MyQueue = sim.Queue (self.name () + '-WaitingLine')
    def process(self):
        while True:
            BagsonCont = int(self.BagsonCont.sample())
            T2W= self.T2W.sample()
            while self.MyQueue.length() == 0:
                    yield self.passivate()
            FirstAirplane = self.MyQueue.pop()
            self.NrBags=FirstAirplane.NrBags
            self.CNrBags=FirstAirplane.CNrBags
            self.HNrBags=FirstAirplane.HNrBags
            print('cold bags in cont',self.CNrBags)
            print('hot bags in cont',self.HNrBags)
            self.EntryArea = FirstAirplane.Entry_Area
            self.ArrivalArea = FirstAirplane.Arrival_Area
            self.tempcont = FirstAirplane.Temperature
            NRofConts= math.ceil(FirstAirplane.NrBags/BagsonCont)
            yield self.hold(NRofConts*T2W)
            Containerss.append(math.ceil(FirstAirplane.NrBags/BagsonCont))
            self.type =1
            #print('number of bags=',FirstAirplane.NrBags,'Bags fit in this container',BagsonCont,' so
            ↪  Containers needed=',NRofConts)
            self.pickup=FirstAirplane.Arrivaltime
            if Peakstartseconds <= self.pickup <= Peakendseconds:
                if self.CNrBags >= 25:
                    ShortestLine = 0
                    for i in range (1, NumberofColdTugs):
                        if ColdTug[i].MyQueue.length () < ColdTug[ShortestLine].MyQueue.length ():
                            ShortestLine = i
                    self.enter(ColdTug[ShortestLine].MyQueue)
                    if ColdTug[ShortestLine].ispassive():
                        ColdTug[ShortestLine].activate()

                    ShortestLine = 0
                    for i in range (1, NumberofTugs):
                        if Tug[i].MyQueue.length () < Tug[ShortestLine].MyQueue.length ():
                            ShortestLine = i
                    self.enter(Tug[ShortestLine].MyQueue)
                    if Tug[ShortestLine].ispassive():
                        Tug[ShortestLine].activate()
                    yield self.passivate
                else:
                    print('iets')
                    ShortestLine = 0
                    for i in range (1, NumberofTugs):
                        if Tug[i].MyQueue.length () < Tug[ShortestLine].MyQueue.length ():
                            ShortestLine = i
                    self.enter(Tug[ShortestLine].MyQueue)
                    if Tug[ShortestLine].ispassive():
                        Tug[ShortestLine].activate()
                    yield self.passivate
            else:
                print('time is above 10000')
                ShortestLine = 0
                for i in range (1, NumberofTugs):
                    if Tug[i].MyQueue.length () < Tug[ShortestLine].MyQueue.length ():
                        ShortestLine = i
                self.enter(Tug[ShortestLine].MyQueue)
                if Tug[ShortestLine].ispassive():
                    Tug[ShortestLine].activate()
                yield self.passivate
#==================================================================================================================
class TColdTug(sim.Component):
    def setup(self):
        self.MyQueue = sim.Queue (self.name () + '-WaitingLine')
    def process(self):
        while True:
            while self.MyQueue.length() == 0:
                yield self.passivate()
            FirstCart = self.MyQueue.pop()
            self.NrBags=FirstCart.CNrBags
            print('bags in cold tug', self.NrBags)
            self.EntryArea = int(FirstCart.EntryArea)
            self.ArrivalArea = int(FirstCart.ArrivalArea)
            yield self.hold(DrivingTimes.iloc[self.ArrivalArea,self.EntryArea])
            ShortestLine = 0
            for i in range(1, NumberofBuffers):
                if Buffer3[i].MyQueue.length () < Buffer3[ShortestLine].MyQueue.length ():
                     ShortestLine = i
            self.enter(Buffer3[ShortestLine].MyQueue)
            if Buffer3[ShortestLine].ispassive():
                Buffer3[ShortestLine].activate()
            yield self.passivate
            FirstCart.activate()
#===============================================================================================================
class TTug(sim.Component):
    def setup(self):
        self.MyQueue = sim.Queue (self.name () + '-WaitingLine')

    def process(self):
        while True:
            while self.MyQueue.length() == 0:
                yield self.passivate()
            FirstCart = self.MyQueue.pop()
            self.CNrBags=FirstCart.CNrBags
            self.HNrBags=FirstCart.HNrBags
            print('cold bags',self.CNrBags)
            print('hot bags',self.HNrBags)
            self.type=FirstCart.type
            self.pickup = FirstCart.pickup
            if Peakstartseconds <= self.pickup <= Peakendseconds:
                if self.type == 0:
                    if self.CNrBags >= 5:
                        self.NrBags = self.HNrBags
                    else:
                        self.NrBags = self.HNrBags+self.CNrBags
                if self.type == 1:
                    if self.CNrBags>= 25:
                        self.NrBags = self.HNrBags
                    else:
                        self.NrBags = self.HNrBags+self.CNrBags
            else:
                self.NrBags=self.HNrBags+self.CNrBags
            print('bags in tug', self.NrBags)
            self.EntryArea = int(FirstCart.EntryArea)
            self.ArrivalArea = int(FirstCart.ArrivalArea)
            print('entry',self.EntryArea,'arrival',self.ArrivalArea)
            yield self.hold(DrivingTimes.iloc[self.ArrivalArea,self.EntryArea])
            DrivingTime.append(DrivingTimes.iloc[self.ArrivalArea,self.EntryArea])
            if self.EntryArea == 0:
                ShortestLine = 0
                for i in range(1, InfeedsE):
                    if InfeedE[i].MyQueue.length () < InfeedE[ShortestLine].MyQueue.length ():
                        ShortestLine = i
                self.enter(InfeedE[ShortestLine].MyQueue)
                if InfeedE[ShortestLine].ispassive():
                    InfeedE[ShortestLine].activate()

                yield self.passivate
                FirstCart.activate()

            if self.EntryArea == 1:
                ShortestLine = 0
                for i in range(1, InfeedsD):
                    if InfeedD[i].MyQueue.length () < InfeedD[ShortestLine].MyQueue.length ():
                        ShortestLine = i
                self.enter(InfeedD[ShortestLine].MyQueue)
                if InfeedD[ShortestLine].ispassive():
                    InfeedD[ShortestLine].activate()
                yield self.passivate
                FirstCart.activate()

            if self.EntryArea == 2:
                ShortestLine = 0
                for i in range(1, InfeedsZ):
                    if InfeedZ[i].MyQueue.length () < InfeedZ[ShortestLine].MyQueue.length ():
                        ShortestLine = i

                self.enter(InfeedZ[ShortestLine].MyQueue)
                if InfeedZ[ShortestLine].ispassive():
                    InfeedZ[ShortestLine].activate()
                yield self.passivate
                FirstCart.activate()
                #self.leave(InfeedZ[ShortestLine].MyQueue)
            else:
                self.release()
```

```
#===================================================================================================
class TBuffer1(sim.Component):
    def setup(self):
        self.MyQueue = sim.Queue(self.name()+'-waitingline')
    def process(self):
        while True:
            while self.MyQueue.length() == 0:
                yield self.passivate()
            FirstAirplane = self.MyQueue.pop()
            self.NrBags=FirstAirplane.NrBags
            FirstAirplane.activate()
            yield self.hold(BufferTime)
            ShortestLine = 0
            for i in range(1, InfeedsE):
                if InfeedE[i].MyQueue.length () < InfeedE[ShortestLine].MyQueue.length ():
                    ShortestLine = i
            self.enter(InfeedE[ShortestLine].MyQueue)
            if InfeedE[ShortestLine].ispassive():
                InfeedE[ShortestLine].activate()

#===================================================================================================
class TBuffer2(sim.Component):
    def setup(self):
        self.MyQueue = sim.Queue(self.name()+'-waitingline')
    def process(self):
        while True:
            while self.MyQueue.length() == 0:
                yield self.passivate()
            FirstAirplane = self.MyQueue.pop()
            self.NrBags=FirstAirplane.NrBags
            FirstAirplane.activate()
            yield self.hold(BufferTime)
            ShortestLine = 0
            for i in range(1, InfeedsD):
                if InfeedD[i].MyQueue.length () < InfeedD[ShortestLine].MyQueue.length ():
                    ShortestLine = i
            self.enter(InfeedD[ShortestLine].MyQueue)
            if InfeedD[ShortestLine].ispassive():
                InfeedD[ShortestLine].activate()

#===================================================================================================
class TBuffer3(sim.Component):
    def setup(self):
        self.MyQueue = sim.Queue(self.name()+'-waitingline')
    def process(self):
        while True:
            while self.MyQueue.length() == 0:
                yield self.passivate()
            FirstAirplane = self.MyQueue.pop()
            self.NrBags=FirstAirplane.NrBags
            FirstAirplane.activate()
            yield self.hold(BufferTime)
            ShortestLine = 0
            for i in range(1, InfeedsZ):
                if InfeedZ[i].MyQueue.length () < InfeedZ[ShortestLine].MyQueue.length ():
                    ShortestLine = i
            self.enter(InfeedZ[ShortestLine].MyQueue)
            if InfeedZ[ShortestLine].ispassive():
                InfeedZ[ShortestLine].activate()

#===================================================================================================
class TInfeedE(sim.Component):
    def setup(self):
        self.MyQueue = sim.Queue(self.name()+'-waitingline')
    def process(self):
        while True:
            while self.MyQueue.length() == 0:
                yield self.passivate()
            FirstAirplane = self.MyQueue.pop()
            InfeedTimeE.append(env.now())
            Sum2.append(env.now())
            BagsE.append(FirstAirplane.NrBags)
            print('nr of bags at E',FirstAirplane.NrBags)
            yield self.hold(Loadingtime*FirstAirplane.NrBags)
            yield FirstAirplane.hold(300)
            FirstAirplane.activate()
#=====================================================================================================================
class TInfeedD(sim.Component):
    def setup(self):
        self.MyQueue = sim.Queue(self.name()+'-waitingline')
    def process(self):
        while True:
            while self.MyQueue.length() == 0:
                yield self.passivate()
            FirstAirplane = self.MyQueue.pop()
            InfeedTimeD.append(env.now())
            Sum2.append(env.now())
            BagsD.append(FirstAirplane.NrBags)
            print('nr of bags D',FirstAirplane.NrBags)
            yield self.hold(Loadingtime*FirstAirplane.NrBags)
            yield FirstAirplane.hold(300)
            FirstAirplane.activate()
#=====================================================================================================================
class TInfeedZ(sim.Component):
    def setup(self):
        self.MyQueue = sim.Queue(self.name()+'-waitingline')
    def process(self):
        while True:
            while self.MyQueue.length() == 0:
                yield self.passivate()
            FirstAirplane = self.MyQueue.pop()
            Sum2.append(env.now())
            InfeedTimeZ.append(env.now())
            BagsZ.append(FirstAirplane.NrBags)
            print('nr of bags Z',FirstAirplane.NrBags)
            yield self.hold(Loadingtime*FirstAirplane.NrBags)
            yield FirstAirplane.hold(300)
            FirstAirplane.activate()
#=====================================================================================================================




env = sim.Environment(trace=True)
AirplaneGenerator = TAirplaneGenerator(LB = BagLB, UB = BagUB, T1LB=T1LB, T1UB=T1UB)
AllAirplanes = sim.Queue('all-airplanes')
Buffer1={}
for i in J:
    Buffer1[i]=TBuffer1()
Buffer2={}
for i in J:
    Buffer2[i]=TBuffer2()
Buffer3={}
for i in J:
    Buffer3[i]=TBuffer3()
InfeedE ={}
for i in range(0, InfeedsE):
    InfeedE[i]=TInfeedE()

InfeedD ={}
for i in range(0, InfeedsD):
    InfeedD[i]=TInfeedD()

InfeedZ ={}
for i in range(0, InfeedsZ):
    InfeedZ[i]=TInfeedZ()

Carts = {}
for i in M:
    Carts[i] = TCarts(CartLB=CartLB, CartUB=CartUB, CartMode=CartMode, T2NALB=T2NALB, T2NAUB=T2NAUB)

Containers = {}
for i in N:
    Containers[i] = TContainers(ContLB=ContLB, ContUB=ContUB, ContMode=ContMode,
    ↪  T2WLB=T2WLB,T2WUB=T2WUB,T2WMode=T2WMode)

Tug ={}
for i in J:
    Tug[i]=TTug()
```

```
ColdTug ={}
for i in J:
    ColdTug[i]=TColdTug()


env.run(till=86400)

print('\n')





#print('\n')
#AllAirplanes.print_statistics()

print('\nREADY')


# COMMAND ----------

for i in range(0,20):
    Tug[i].MyQueue.print_statistics()

# COMMAND ----------

for i in range(0,20):
    ColdTug[i].MyQueue.print_statistics()

# COMMAND ----------

for i in range(21,40):
    ColdTug[i].MyQueue.print_statistics()

# COMMAND ----------

for i in range(41,60):
    ColdTug[i].MyQueue.print_statistics()

# COMMAND ----------

ResultE={}                                                         #Create dataframe for data of
↪  infeedpoint E
ResultE['TB']=BagsE                                                #Add the total amount of bags
↪  brought to E (from model)
ResultE['time']=InfeedTimeE                                        #Add the time the bags are
↪  delivered at E (from model)
ResultE=pd.DataFrame(ResultE)                                      #turn dataframe in to pandas
↪  format
KoffersE1=[]                                                       #Create dataframe to append data
↪  into in the for loop
for i in range(0,len(BagsE)):                                      #Create for loop with length of
↪  the total amount of entries
    for j in range(0,int(ResultE['TB'][i])):                       #Create loop in loop with the
    ↪  total amount of bags that are entered at that moment
        KoffersE1.append((j+1)*LoadingtimeE+ResultE['time'][i])    #Create individual infeed times
        ↪  for each bag based in arrivaltime and loadingtime
KoffersE1 = pd.DataFrame(KoffersE1)                                 #turn dataframe in to pandas
↪  format

ResultD={}                                                         #Same steps as above for E but
↪  then D
ResultD['TB']=BagsD
ResultD['time']=InfeedTimeD
ResultD=pd.DataFrame(ResultD)
KoffersD1=[]
for i in range(0,len(BagsD)):
    for j in range(0,int(ResultD['TB'][i])):
        KoffersD1.append((j+1)*LoadingtimeD+ResultD['time'][i])
KoffersD1 = pd.DataFrame(KoffersD1)

ResultZ={}                                                         #Same steps as above for E but
↪  then South
ResultZ['TB']=BagsZ
ResultZ['time']=InfeedTimeZ
```

```
ResultZ=pd.DataFrame(ResultZ)
KoffersZ1=[]
for i in range(0,len(BagsZ)):
    for j in range(0,int(ResultZ['TB'][i])):
        KoffersZ1.append((j+1)*LoadingtimeZ+ResultZ['time'][i])
KoffersZ1 = pd.DataFrame(KoffersZ1)
print(data['ENTRY_HANDLING_AREA'].value_counts())

print('E',len(KoffersE1),'D',len(KoffersD1),'Z',len(KoffersZ1))
print('E',sum(BagsE),'D',sum(BagsD),'Z',sum(BagsZ))

KoffersAll = [KoffersE1,KoffersD1,KoffersZ1]
KoffersAll =  pd.concat(KoffersAll)
print(KoffersAll)
KoffersAll['time']=KoffersAll
#KoffersAll= pd.DataFrame(KoffersAll)
KoffersAll['time']= pd.to_datetime(KoffersAll['time'], unit='s' )
#print(Koffers)



print('uitkomst',KoffersAll)
df13buffer = KoffersAll['time'].value_counts(bins = dts5).sort_index()

ax= plt.figure()                                                  #Create plot of counted frame
plt.plot(dts6,df13buffer,label='model infeedtimes')
#plt.plot(dts6,df10,label='model')
plt.plot(dts2,df3, label='excel data',linestyle='dashed')
#plt.plot(dts2,df3, label='excel data',linestyle='dashed')
plt.xlabel("Time")                                                 #Add X label
plt.ylabel("BAX")                                                  #Add Y label
plt.xticks(rotation = 90)                                       #Turn X data ticks
plt.xticks(np.arange(3, 92, 4))                                    #Reduce ticks
plt.title("BAX infeed per quarter hour")                           #add title
plt.legend(loc="upper left")                                       #Add legend
ax.set_figwidth(20)
ax.set_figheight(8)
plt.show()
df3=pd.DataFrame(df3)
aantal =df3['TIME_OF_ENTRY_LT'].astype(bool).sum(axis=0)
print('aantal=',aantal)
Gem=df3['TIME_OF_ENTRY_LT'].sum()/aantal

print('Gem=',Gem)
diff=[None] * len(df13buffer)
reldif=[None] * len(df13buffer)
reldif2=[None] * len(df13buffer)
for i in range(0, len(df13buffer)):
    diff[i]=df3['TIME_OF_ENTRY_LT'][i]-df13buffer[i]
    reldif[i]=((diff[i]/df3['TIME_OF_ENTRY_LT'][i])**2)**0.5
    reldif2[i]= ((diff[i]/Gem)**2)**0.5
reldif=pd.DataFrame(reldif)
reldif2=pd.DataFrame(reldif2)
print(reldif)
from numpy import inf
reldif[reldif==inf]=0
reldif=reldif.fillna(0)
reldif2[reldif2==inf]=0
reldif2=reldif2.fillna(0)
res =  [abs(ele) for ele in diff]
x=reldif2.sum()
x1=reldif.sum()
print('average zonder 0',x/len(reldif))
print('average',x1/len(reldif))
print('average with minus and plus',sum(diff)/len(diff))
print('abs. average',sum(res)/len(diff))
print('peak at', max(df13buffer))
print('model peak at', max(df3))
df13buffer=pd.DataFrame(df13buffer)
display(df13buffer)

# COMMAND ----------

df14buffer = KoffersAll['time'].value_counts(bins = dts5_1).sort_index()
```

```
df11 = df['TIME_OF_ENTRY_LT'].value_counts(bins = dts_1).sort_index()  #create dataframe that has the
↪  counts of the dataset per quarter hour
ax= plt.figure()                                                    #Create plot of counted frame
plt.plot(dts6_1,df14buffer,label='model infeed')
plt.plot(dts2_1,df11,label='excel data',linestyle='dashed')
↪  #Add data
#plt.plot(dts6_1,df12,label='model')
plt.xlabel("Time")                                                #Add X label
plt.ylabel("BAX")                                                 #Add Y label
plt.xticks(rotation = 90)                                         #Turn X data ticks
plt.xticks(np.arange(14, 276, 12))                                #Reduce ticks
plt.title("BAX infeed per 5 minutes")                             #add title
plt.legend(loc="upper left")
ax.set_figwidth(20)
ax.set_figheight(8)
plt.show()

diff=[None] * len(df14buffer)
reldif=[None] * len(df14buffer)
for i in range(0, len(df14buffer)):
    diff[i]=df11[i]-df14buffer[i]
    reldif[i]=diff[i]/df11[i]
res =  [abs(ele) for ele in diff]

print('average with minus and plus',sum(diff)/len(diff))
print('abs. average',sum(res)/len(diff))
print('peak at', max(df11))
print('model peak at', max(df14buffer))

# COMMAND ----------

KoffersZ1['time']=KoffersZ1
KoffersZ1['time']= pd.to_datetime(KoffersZ1['time'], unit='s' )
dfZmodbuf = KoffersZ1['time'].value_counts(bins = dts5).sort_index()

ax1= plt.figure()                                                        #Create plot of counted frame
plt.plot(dts6,dfZmodbuf)
plt.plot(dts2,dfz3,linestyle='dashed')                                           #Add south data
plt.xlabel("Time")                                                       #Add X label
plt.ylabel("BAX")                                                        #Add Y label
plt.xticks(rotation = 90,fontsize=10)                                    #Turn X data ticks
plt.yticks(fontsize=10)                                                  #Change fontsize ticks
axes = plt.gca()
axes.set_ylim([0,850])
plt.xticks(np.arange(3, 92, 4))                                          #Reduce ticks
plt.title("BAX infeed South per quarter hour (15-07-2019)" , fontsize=20)      #add title
#plt.legend(loc="upper left",fontsize=10)                                 #Add legend
ax1.set_figwidth(10)                                                     #set width dimention of plot
ax1.set_figheight(3.5)
plt.show()
                                          #Add E data
KoffersE1['time']=KoffersE1
KoffersE1['time']= pd.to_datetime(KoffersE1['time'], unit='s' )
dfEmodbuf = KoffersE1['time'].value_counts(bins = dts5).sort_index()
ax2= plt.figure()
plt.plot(dts6,dfEmodbuf)
plt.plot(dts2,dfe3,linestyle='dashed')                                           #Add south data
plt.xlabel("Time")                                                       #Add X label
plt.ylabel("BAX")                                                        #Add Y label
plt.xticks(rotation = 90,fontsize=10)                                    #Turn X data ticks
plt.yticks(fontsize=10)                                                  #Change fontsize ticks
axes = plt.gca()
axes.set_ylim([0,850])
plt.xticks(np.arange(3, 92, 4))                                          #Reduce ticks
plt.title("BAX infeed E per quarter hour (15-07-2019)" , fontsize=20)      #add title
#plt.legend(loc="upper left",fontsize=10)                                 #Add legend
ax2.set_figwidth(10)                                                     #set width dimention of plot
ax2.set_figheight(3.5)
plt.show()

KoffersD1['time']=KoffersD1
KoffersD1['time']= pd.to_datetime(KoffersD1['time'], unit='s' )
dfDmodbuf = KoffersD1['time'].value_counts(bins = dts5).sort_index()
ax3= plt.figure()                                                       #Create plot of counted frame
plt.plot(dts6,dfDmodbuf)
plt.plot(dts2,dfd3,linestyle='dashed')                                          #Add south data
plt.xlabel("Time")                                                      #Add X label
plt.ylabel("BAX")                                                       #Add Y label
plt.xticks(rotation = 90,fontsize=10)                                   #Turn X data ticks
plt.yticks(fontsize=10)                                                 #Change fontsize ticks
axes = plt.gca()
axes.set_ylim([0,850])
plt.xticks(np.arange(3, 92, 4))                                         #Reduce ticks
plt.title("BAX infeed D per quarter hour (15-07-2019)" , fontsize=20)     #add title
#plt.legend(loc="upper left",fontsize=10)                                #Add legend
ax3.set_figwidth(10)                                                    #set width dimention of plot
ax3.set_figheight(3.5)
plt.show()

# COMMAND ----------

import salabim as sim

#airplanes
#IAT = 300      #mean Interarivaltimes between airplanes
T1LB = 120      #Time between In block time and first bag/container removed lowerbound
T1UB = 180      #Time between In block time and first bag/container removed upperboundbound
T2NALB = 6      #Lowerbound
T2NAUB = 7.5
T2WLB= 55
T2WUB= 115
T2WMode= 75
BagLB = 1       #Lowerbound Amount of bags in airplane
BagUB = 80      #Upperbound Amount of bags in airplane


#Infeedstations
InfeedsE = 4
CapacityE = 900

InfeedsD = 5
CapacityD = 900

InfeedsZ = 2
CapacityZ = 1000

Loadingtime = 3.53
LoadingtimeD = 4
LoadingtimeE = 4
LoadingtimeZ = 3.6

Drivingtime = 30

#Carts
NumberofCarts = NRNABO
CartLB = 20
CartUB = 40
CartMode = 30

#Tugs
NumberofTugs = 100
#Containers
NumberofContainers = NRWIBO
ContLB = 30
ContUB = 42
ContMode = 38

#output
WaitingTime=[]
UnloadingTime=[]
Cartss = []
Containerss=[]
DrivingTime=[]
InfeedTimeE=[]
BagsE=[]
InfeedTimeD=[]
BagsD=[]
InfeedTimeZ=[]
BagsZ=[]
Eentryarea=[]
```

```
Entry=[]
Bagsinplane=[]
Bagsinplane2=[]
Licenseplatemodel =[]
Arrivaltime=[]
Sum2=[]
Inter=[]
Inter2=[]
Inter3=[]
#=============================================================================================================

class TAirplaneGenerator(sim.Component):
    def setup(self, LB, UB, T1LB, T1UB ):
        self.ArrivalTime = np.zeros(NRplanes-1)
        self.ArrivalArea = np.zeros(NRplanes-1)
        #self.BagsPerPlane = sim.Uniform(LB, UB+1)
        self.BagsPerPlane =np.zeros(NRplanes-1)
        self.T1 = sim.Uniform(T1LB, T1UB)
        self.BodyType = np.zeros(NRplanes-1)
        self.Temp = np.zeros(NRplanes-1)
        self.EntryArea = np.zeros(NRplanes-1)
        self.LicencePlate = np.zeros(NRplanes-1)
    def process(self):
        for i in I:
            InterarrivalTime = np.zeros(NRplanes-1)
            InterarrivalTime[i] = data['AIBT_S'][i+1]-data['AIBT_S'][i]
            #Inter.append(data['AIBT_S'][i+1]-data['AIBT_S'][i])
            #Inter2.append(self.InterarrivalTime[i])
            myIAT = InterarrivalTime[i]
            self.ArrivalTime[i] = data['AIBT_S'][i+1]

            self.ArrivalArea[i]=data['GATE_AREA'][i+1]
            self.EntryArea[i]=data['ENTRY_HANDLING_AREA'][i+1]
            self.LicencePlate[i]= data['LICENSE_PLATE_CODE'][i+1]
            LicensePlate=self.LicencePlate[i]
            EntryArea = self.EntryArea[i]
            ArrivalArea = self.ArrivalArea[i]
            ArrivalTime= self.ArrivalTime[i]
            self.Temp[i]= data['temp'][i+1]
            Temp= self.Temp[i]
            Temp_output = "cold" if  Temp==0 else "hot"
            print('Temp=',Temp_output)
            print('gebied nummer =',self.ArrivalArea[i])
            print('Entry Area =',self.EntryArea[i])
            self.BodyType[i]=data['BO'][i+1]


            self.BagsPerPlane[i] =data['total'][i+1]
            NrB = self.BagsPerPlane[i]
            Bagsinplane2.append(NrB)
            #NrB = int(self.BagsPerPlane.sample())
            T1= self.T1.sample()

            BO= self.BodyType[i]
            BO_output = "WIBO" if  BO==1 else "NABO"
            print('BodyType=',BO_output)
            yield self.hold(till=data['AIBT_S'][i+1])
            NewAirplane = TAirplane(NrBag = NrB,T10 = T1, BoTy=BO, temp=Temp, EA=EntryArea, AA =
            ↪  ArrivalArea, LP=LicensePlate, AT=ArrivalTime)
            NewAirplane.enter(AllAirplanes)



 ↪  #=============================================================================================================
class TAirplane(sim.Component):
    def setup(self, NrBag, T10, BoTy, temp,EA, AA, LP, AT):
        self.BodyType1 = BoTy
        self.NrBags = NrBag
        self.T100 = T10

        self.Temperature = temp
        self.Entry_Area = EA
        self.Arrival_Area = AA
        self.LicensePlates = LP
        self.Arrivaltime = AT
    def process(self):
        FirstAirplane = AllAirplanes.pop()
        Inter3.append(env.now())
        yield self.hold(self.T100)
        WaitingTime.append(self.T100)

        #UnloadingTime.append(self.NrBags*self.T200N)
        Licenseplatemodel.append(self.LicensePlates)
        Bagsinplane.append(self.NrBags)
        Entry.append(self.Entry_Area)
        Arrivaltime.append(self.Arrivaltime)
        #D = pd.DataFrame(D)
        print('BoTy=',self.BodyType1)
        print('temp=',self.Temperature)
        if self.BodyType1 == 0:
            ShortestLine = 0
            for i in range (1, NumberofCarts):
                if Carts[i].MyQueue.length () < Carts[ShortestLine].MyQueue.length ():
                    ShortestLine = i
            self.enter(Carts[ShortestLine].MyQueue)
            if Carts[ShortestLine].ispassive():
                Carts[ShortestLine].activate()
        if self.BodyType1 == 1:
            #yield self.hold(self.NrBags*self.T200N)
            #UnloadingTime.append(self.NrBags*self.T200N)
            ShortestLine = 0
            for i in range (1, NumberofContainers):
                if Containers[i].MyQueue.length () < Containers[ShortestLine].MyQueue.length ():
                    ShortestLine = i
            self.enter(Containers[ShortestLine].MyQueue)
            if Containers[ShortestLine].ispassive():
                Containers[ShortestLine].activate()
#==============================================================================================================
class TCarts(sim.Component):
    def setup(self, CartLB, CartUB, CartMode, T2NALB, T2NAUB):
        self.BagsonCart = sim.Triangular(CartLB,CartUB,CartMode)
        self.MyQueue = sim.Queue (self.name () + '-WaitingLine')
        self.T2N = sim.Uniform(T2NALB,T2NAUB)
    def process(self):
        while True:
            BagsonCart = int(self.BagsonCart.sample())
            T2N= self.T2N.sample()
            while self.MyQueue.length() == 0:
                yield self.passivate()
            FirstAirplane = self.MyQueue.pop()
            self.NrBags=FirstAirplane.NrBags
            yield self.hold(self.NrBags*T2N)
            self.EntryArea = FirstAirplane.Entry_Area
            self.ArrivalArea = FirstAirplane.Arrival_Area
            self.tempcart = FirstAirplane.Temperature
            print('temp cart=',self.tempcart)
            NRofCarts= math.ceil(FirstAirplane.NrBags/BagsonCart)
            Cartss.append(NRofCarts)
            print('number of bags=',FirstAirplane.NrBags,'Bags fit on this cart',BagsonCart,' so Carts
            ↪  needed=',NRofCarts)
            ShortestLine = 0
            for i in range (1, NumberofTugs):
                if Tug[i].MyQueue.length () < Tug[ShortestLine].MyQueue.length ():
                    ShortestLine = i
            self.enter(Tug[ShortestLine].MyQueue)
            if Tug[ShortestLine].ispassive():
                Tug[ShortestLine].activate()
            yield self.passivate
#==============================================================================================================
class TContainers(sim.Component):
    def setup(self, ContLB, ContUB, ContMode, T2WLB,T2WUB,T2WMode):
        self.BagsonCont = sim.Triangular(ContLB,ContUB,ContMode)
        self.T2W = sim.Triangular(T2WLB,T2WUB,T2WMode)
        self.MyQueue = sim.Queue (self.name () + '-WaitingLine')
    def process(self):
        while True:
            BagsonCont = int(self.BagsonCont.sample())
            T2W= self.T2W.sample()
            while self.MyQueue.length() == 0:
                    yield self.passivate()
            FirstAirplane = self.MyQueue.pop()
            self.NrBags=FirstAirplane.NrBags
            self.EntryArea = FirstAirplane.Entry_Area
            self.ArrivalArea = FirstAirplane.Arrival_Area
            self.tempcont = FirstAirplane.Temperature
            print('temp container=',self.tempcont)
            NRofConts= math.ceil(FirstAirplane.NrBags/BagsonCont)
            yield self.hold(NRofConts*T2W)
            Containerss.append(math.ceil(FirstAirplane.NrBags/BagsonCont))
            print('number of bags=',FirstAirplane.NrBags,'Bags fit in this container',BagsonCont,' so
            ↪  Containers needed=',NRofConts)
            ShortestLine = 0
            for i in range (1, NumberofTugs):
                if Tug[i].MyQueue.length () < Tug[ShortestLine].MyQueue.length ():
                    ShortestLine = i
            self.enter(Tug[ShortestLine].MyQueue)
            if Tug[ShortestLine].ispassive():
                Tug[ShortestLine].activate()
            yield self.passivate
#=====================================================================================================================
class TTug(sim.Component):
    def setup(self):
        self.MyQueue = sim.Queue (self.name () + '-WaitingLine')

    def process(self):
        while True:
            while self.MyQueue.length() == 0:
                yield self.passivate()
            FirstCart = self.MyQueue.pop()
            self.NrBags=FirstCart.NrBags
            self.EntryArea = int(FirstCart.EntryArea)
            self.ArrivalArea = int(FirstCart.ArrivalArea)
            print('Entry area =' ,self.EntryArea,'arrival =', self.ArrivalArea)
            yield self.hold(DrivingTimes.iloc[self.ArrivalArea,self.EntryArea])
            print('driving time=', DrivingTimes.iloc[self.ArrivalArea,self.EntryArea])
            DrivingTime.append(DrivingTimes.iloc[self.ArrivalArea,self.EntryArea])
            if self.EntryArea == 0:
                ShortestLine = 0
                for i in range(1, InfeedsE):
                    if InfeedE[i].MyQueue.length () < InfeedE[ShortestLine].MyQueue.length ():
                        ShortestLine = i
                self.enter(InfeedE[ShortestLine].MyQueue)
                if InfeedE[ShortestLine].ispassive():
                    InfeedE[ShortestLine].activate()
                FirstCart.activate()
                yield self.passivate

            if self.EntryArea == 1:
                ShortestLine = 0
                for i in range(1, InfeedsD):
                    if InfeedD[i].MyQueue.length () < InfeedD[ShortestLine].MyQueue.length ():
                        ShortestLine = i
                self.enter(InfeedD[ShortestLine].MyQueue)
                if InfeedD[ShortestLine].ispassive():
                    InfeedD[ShortestLine].activate()
                FirstCart.activate()
                yield self.passivate

            if self.EntryArea == 2:
                ShortestLine = 0
                for i in range(1, InfeedsZ):
                    if InfeedZ[i].MyQueue.length () < InfeedZ[ShortestLine].MyQueue.length ():
                        ShortestLine = i

                self.enter(InfeedZ[ShortestLine].MyQueue)
                if InfeedZ[ShortestLine].ispassive():
                    InfeedZ[ShortestLine].activate()
                FirstCart.activate()
                yield self.passivate
                #self.leave(InfeedZ[ShortestLine].MyQueue)
            else:
                self.release()

#=====================================================================================================================
class TBuffer(sim.Component):
    def setup(self):
        self.MyQueue = sim.Queue(self.name()+'-waitingline')
    def process(self):
        while True:
            while self.MyQueue.length() == 0:
                yield self.passivate()
#==================================================================================================================
class TInfeedE(sim.Component):
    def setup(self):
        self.MyQueue = sim.Queue(self.name()+'-waitingline')
    def process(self):
        while True:
            while self.MyQueue.length() == 0:
                yield self.passivate()
            FirstAirplane = self.MyQueue.pop()
            InfeedTimeE.append(env.now())
            Sum2.append(env.now())
            BagsE.append(FirstAirplane.NrBags)
            Eentryarea.append(FirstAirplane.EntryArea)
            yield self.hold(Loadingtime*FirstAirplane.NrBags)
            FirstAirplane.activate()
#==================================================================================================================
class TInfeedD(sim.Component):
    def setup(self):
        self.MyQueue = sim.Queue(self.name()+'-waitingline')
    def process(self):
        while True:
            while self.MyQueue.length() == 0:
                yield self.passivate()
            FirstAirplane = self.MyQueue.pop()
            InfeedTimeD.append(env.now())
            Sum2.append(env.now())
            BagsD.append(FirstAirplane.NrBags)
            yield self.hold(Loadingtime*FirstAirplane.NrBags)
            FirstAirplane.activate()
#==================================================================================================================
class TInfeedZ(sim.Component):
    def setup(self):
        self.MyQueue = sim.Queue(self.name()+'-waitingline')
    def process(self):
        while True:
            while self.MyQueue.length() == 0:
                yield self.passivate()
            FirstAirplane = self.MyQueue.pop()
            Sum2.append(env.now())
            InfeedTimeZ.append(env.now())
            BagsZ.append(FirstAirplane.NrBags)
            yield self.hold(Loadingtime*FirstAirplane.NrBags)
            FirstAirplane.activate()
#==================================================================================================================




env = sim.Environment(trace=True)
AirplaneGenerator = TAirplaneGenerator(LB = BagLB, UB = BagUB, T1LB=T1LB, T1UB=T1UB)
AllAirplanes = sim.Queue('all-airplanes')
processingtime = sim.Monitor('processingtime')
InfeedE ={}
for i in range(0, InfeedsE):
    InfeedE[i]=TInfeedE()

InfeedD ={}
for i in range(0, InfeedsD):
    InfeedD[i]=TInfeedD()

InfeedZ ={}
for i in range(0, InfeedsZ):
    InfeedZ[i]=TInfeedZ()

Carts = {}
for i in M:
    Carts[i] = TCarts(CartLB=CartLB, CartUB=CartUB, CartMode=CartMode, T2NALB=T2NALB, T2NAUB=T2NAUB)
```

```
Containers = {}
for i in N:
    Containers[i] = TContainers(ContLB=ContLB, ContUB=ContUB, ContMode=ContMode,
    ↪  T2WLB=T2WLB,T2WUB=T2WUB,T2WMode=T2WMode)

Tug ={}
for i in J:
    Tug[i]=TTug()


env.run(till=86400)

print('\n')
for i in range(0, InfeedsE):
    InfeedE[i].MyQueue.print_statistics()



#print('\n')
#AllAirplanes.print_statistics()

print('\nREADY')

# COMMAND ----------

print('\n')
tuug= {}
for i in range(0,20):
    Tug[i].MyQueue.print_statistics()
for i in range(0,20):
    ColdTug[i].MyQueue.print_statistics()




# COMMAND ----------

print(tuug)

# COMMAND ----------

ResultE={}                                                          #Create dataframe for data of
↪  infeedpoint E
ResultE['TB']=BagsE                                                 #Add the total amount of bags
↪  brought to E (from model)
ResultE['time']=InfeedTimeE                                         #Add the time the bags are
↪  delivered at E (from model)
ResultE=pd.DataFrame(ResultE)                                       #turn dataframe in to pandas
↪  format
KoffersE1=[]                                                        #Create dataframe to append data
↪  into in the for loop
for i in range(0,len(BagsE)):                                       #Create for loop with length of
↪  the total amount of entries
    for j in range(0,int(ResultE['TB'][i])):                        #Create loop in loop with the
    ↪  total amount of bags that are entered at that moment
        KoffersE1.append((j+1)*LoadingtimeE+ResultE['time'][i])     #Create individual infeed times
        ↪  for each bag based in arrivaltime and loadingtime
KoffersE1 = pd.DataFrame(KoffersE1)                                  #turn dataframe in to pandas
↪  format

ResultD={}                                                          #Same steps as above for E but
↪  then D
ResultD['TB']=BagsD
ResultD['time']=InfeedTimeD
ResultD=pd.DataFrame(ResultD)
KoffersD1=[]
for i in range(0,len(BagsD)):
    for j in range(0,int(ResultD['TB'][i])):
        KoffersD1.append((j+1)*LoadingtimeD+ResultD['time'][i])
KoffersD1 = pd.DataFrame(KoffersD1)

ResultZ={}                                                          #Same steps as above for E but
↪  then South
ResultZ['TB']=BagsZ
ResultZ['time']=InfeedTimeZ
```

```
ResultZ=pd.DataFrame(ResultZ)
KoffersZ1=[]
for i in range(0,len(BagsZ)):
    for j in range(0,int(ResultZ['TB'][i])):
        KoffersZ1.append((j+1)*LoadingtimeZ+ResultZ['time'][i])
KoffersZ1 = pd.DataFrame(KoffersZ1)
print(data['ENTRY_HANDLING_AREA'].value_counts())

KoffersAll = [KoffersE1,KoffersD1,KoffersZ1]
KoffersAll =  pd.concat(KoffersAll)
print(KoffersAll)
KoffersAll['time']=KoffersAll
#KoffersAll= pd.DataFrame(KoffersAll)
KoffersAll['time']= pd.to_datetime(KoffersAll['time'], unit='s' )
#print(Koffers)



print('uitkomst',KoffersAll)
df13 = KoffersAll['time'].value_counts(bins = dts5).sort_index()
print(df13)
ax= plt.figure()                                                  #Create plot of counted frame
plt.plot(dts6,df13,label='model infeedtimes')
#plt.plot(dts6,df10,label='model')
plt.plot(dts2,df3, label='excel data',linestyle='dashed')
#plt.plot(dts2,df3, label='excel data',linestyle='dashed')
plt.xlabel("Time")                                                 #Add X label
plt.ylabel("BAX")                                                  #Add Y label
plt.xticks(rotation = 90)                                       #Turn X data ticks
plt.xticks(np.arange(3, 92, 4))                                    #Reduce ticks
plt.title("BAX infeed per quarter hour")                           #add title
plt.legend(loc="upper left")                                       #Add legend
ax.set_figwidth(20)
ax.set_figheight(8)
plt.show()
df3=pd.DataFrame(df3)
aantal =df3['TIME_OF_ENTRY_LT'].astype(bool).sum(axis=0)
print('aantal=',aantal)
Gem=df3['TIME_OF_ENTRY_LT'].sum()/aantal

print('Gem=',Gem)
diff=[None] * len(df13)
reldif=[None] * len(df13)
reldif2=[None] * len(df13)
for i in range(0, len(df13)):
    diff[i]=df3['TIME_OF_ENTRY_LT'][i]-df13[i]
    reldif[i]=((diff[i]/df3['TIME_OF_ENTRY_LT'][i])**2)**0.5
    reldif2[i]= ((diff[i]/Gem)**2)**0.5
reldif=pd.DataFrame(reldif)
reldif2=pd.DataFrame(reldif2)
print(reldif)
from numpy import inf
reldif[reldif==inf]=0
reldif=reldif.fillna(0)
reldif2[reldif2==inf]=0
reldif2=reldif2.fillna(0)
res =  [abs(ele) for ele in diff]
x=reldif2.sum()
x1=reldif.sum()
print('average zonder 0',x/len(reldif))
print('average',x1/len(reldif))
print('average with minus and plus',sum(diff)/len(diff))
print('abs. average',sum(res)/len(diff))
print('peak at', max(df13))
print('model peak at', max(df3))
df13=pd.DataFrame(df13)
display(df13)

# COMMAND ----------

df14 = KoffersAll['time'].value_counts(bins = dts5_1).sort_index()
df11 = df['TIME_OF_ENTRY_LT'].value_counts(bins = dts_1).sort_index()  #create dataframe that has the
↪  counts of the dataset per quarter hour
ax= plt.figure()                                                  #Create plot of counted frame
plt.plot(dts6_1,df14,label='model infeed')
plt.plot(dts2_1,df11,label='excel data',linestyle='dashed')
↪  #Add data
#plt.plot(dts6_1,df12,label='model')
plt.xlabel("Time")                                                  #Add X label
plt.ylabel("BAX")                                                   #Add Y label
plt.xticks(rotation = 90)                                           #Turn X data ticks
plt.xticks(np.arange(14, 276, 12))                                  #Reduce ticks
plt.title("BAX infeed per 5 minutes")                               #add title
plt.legend(loc="upper left")
ax.set_figwidth(20)
ax.set_figheight(8)
plt.show()

diff=[None] * len(df14)
reldif=[None] * len(df14)
for i in range(0, len(df14)):
    diff[i]=df11[i]-df14[i]
    reldif[i]=diff[i]/df11[i]
res =  [abs(ele) for ele in diff]

print('average with minus and plus',sum(diff)/len(diff))
print('abs. average',sum(res)/len(diff))
print('peak at', max(df11))
print('model peak at', max(df14))

# COMMAND ----------

KoffersZ1['time']=KoffersZ1
KoffersZ1['time']= pd.to_datetime(KoffersZ1['time'], unit='s' )
dfZmod = KoffersZ1['time'].value_counts(bins = dts5).sort_index()

ax1= plt.figure()                                                          #Create plot of counted frame
plt.plot(dts6,dfZmod)
plt.plot(dts2,dfz3,linestyle='dashed')                                             #Add south data
plt.xlabel("Time")                                                         #Add X label
plt.ylabel("BAX")                                                          #Add Y label
plt.xticks(rotation = 90,fontsize=10)                                      #Turn X data ticks
plt.yticks(fontsize=10)                                                    #Change fontsize ticks
axes = plt.gca()
axes.set_ylim([0,850])
plt.xticks(np.arange(3, 92, 4))                                            #Reduce ticks
plt.title("BAX infeed South per quarter hour (15-07-2019)" , fontsize=20)      #add title
#plt.legend(loc="upper left",fontsize=10)                                   #Add legend
ax1.set_figwidth(10)                                                       #set width dimention of plot
ax1.set_figheight(3.5)
plt.show()
                                          #Add E data
KoffersE1['time']=KoffersE1
KoffersE1['time']= pd.to_datetime(KoffersE1['time'], unit='s' )
dfEmod = KoffersE1['time'].value_counts(bins = dts5).sort_index()
ax2= plt.figure()
plt.plot(dts6,dfEmod)
plt.plot(dts2,dfe3,linestyle='dashed')                                             #Add south data
plt.xlabel("Time")                                                         #Add X label
plt.ylabel("BAX")                                                          #Add Y label
plt.xticks(rotation = 90,fontsize=10)                                      #Turn X data ticks
plt.yticks(fontsize=10)                                                    #Change fontsize ticks
axes = plt.gca()
axes.set_ylim([0,850])
plt.xticks(np.arange(3, 92, 4))                                            #Reduce ticks
plt.title("BAX infeed E per quarter hour (15-07-2019)" , fontsize=20)      #add title
#plt.legend(loc="upper left",fontsize=10)                                   #Add legend
ax2.set_figwidth(10)                                                       #set width dimention of plot
ax2.set_figheight(3.5)
plt.show()

KoffersD1['time']=KoffersD1
KoffersD1['time']= pd.to_datetime(KoffersD1['time'], unit='s' )
dfDmod = KoffersD1['time'].value_counts(bins = dts5).sort_index()
ax3= plt.figure()                                                         #Create plot of counted frame
plt.plot(dts6,dfDmod)
plt.plot(dts2,dfd3,linestyle='dashed')                                             #Add south data
plt.xlabel("Time")                                                         #Add X label
plt.ylabel("BAX")                                                          #Add Y label
plt.xticks(rotation = 90,fontsize=10)                                      #Turn X data ticks
plt.yticks(fontsize=10)                                                 #Change fontsize ticks
axes = plt.gca()
axes.set_ylim([0,850])
plt.xticks(np.arange(3, 92, 4))                                         #Reduce ticks
plt.title("BAX infeed D per quarter hour (15-07-2019)" , fontsize=20)     #add title
#plt.legend(loc="upper left",fontsize=10)                                #Add legend
ax3.set_figwidth(10)                                                    #set width dimention of plot
ax3.set_figheight(3.5)
plt.show()

# COMMAND ----------

output1 = df13buffer
print(df13buffer)
display(output1)

output2 = df13
display(output2)

ax= plt.figure()                                                   #Create plot of counted frame
plt.plot(dts6,df13buffer,label='model output with buffer', linewidth=5)
#plt.plot(dts6,df10,label='model')
plt.plot(dts6,df13, label='modeloutput without buffer',linestyle='dashed', linewidth=5)
#plt.plot(dts2,df3, label='excel data',linestyle='dashed')
plt.xlabel("Time", fontsize=25)                                                  #Add X label
plt.ylabel("BAX", fontsize=25)                                                   #Add Y label
plt.xticks(rotation = 90,fontsize=20)                                        #Turn X data ticks
plt.yticks(fontsize=20)
plt.xticks(np.arange(3, 92, 4))
↪  #Reduce ticks
plt.title("BAX infeed per quarter hour", fontsize=35)                            #add title
plt.legend(loc="upper left", fontsize=20)                                        #Add legend
ax.set_figwidth(20)
ax.set_figheight(8)
plt.show()

ax= plt.figure()                                                   #Create plot of counted frame
plt.plot(dts6_1,df14buffer, label='modeloutput with buffer', linewidth=4)
plt.plot(dts6_1,df14, label='modeloutput without buffer',linestyle='dashed', linewidth=4)
↪  #Add data
#plt.plot(dts6_1,df12,label='model')
plt.xlabel("Time")                                               #Add X label
plt.ylabel("BAX")                                                #Add Y label
plt.xticks(rotation = 90)                                        #Turn X data ticks
plt.xticks(np.arange(14, 276, 12))                               #Reduce ticks
plt.title("BAX infeed per 5 minutes")                            #add title
plt.legend(loc="upper left")
ax.set_figwidth(20)
ax.set_figheight(8)
plt.show()

ax1= plt.figure()                                                       #Create plot of counted frame
plt.plot(dts6,dfZmodbuf, label='modeloutput with buffer', linewidth=4)
plt.plot(dts6,dfZmod,linestyle='dashed', label='modeloutput without buffer', linewidth=4)
↪  #Add south data
plt.axhline(y = 500, color = 'r', linestyle = '-',label='max capcity', linewidth=4)
plt.xlabel("Time")                                                      #Add X label
plt.ylabel("BAX")                                                       #Add Y label
plt.xticks(rotation = 90,fontsize=10)                                   #Turn X data ticks
plt.yticks(fontsize=10)                                                 #Change fontsize ticks
axes = plt.gca()
axes.set_ylim([0,1000])
plt.xticks(np.arange(3, 92, 4))                                         #Reduce ticks
plt.title("BAX infeed South per quarter hour (15-07-2019)" , fontsize=20)     #add title
plt.legend(loc="upper left",fontsize=10)                                #Add legend
ax1.set_figwidth(10)                                                    #set width dimention of plot
ax1.set_figheight(3.5)
plt.show()

ax2= plt.figure()
plt.plot(dts6,dfEmodbuf, label='modeloutput with buffer', linewidth=4)
plt.plot(dts6,dfEmod,linestyle='dashed', label='modeloutput without buffer', linewidth=4)
↪  #Add south data
plt.axhline(y = 900, color = 'r', linestyle = '-',label='max capcity', linewidth=4)
plt.xlabel("Time")                                                      #Add X label
plt.ylabel("BAX")                                                   #Add Y label
plt.xticks(rotation = 90,fontsize=10)                               #Turn X data ticks
plt.yticks(fontsize=10)                                             #Change fontsize ticks
axes = plt.gca()
axes.set_ylim([0,1000])
plt.xticks(np.arange(3, 92, 4))                                     #Reduce ticks
plt.title("BAX infeed E per quarter hour (15-07-2019)" , fontsize=20)   #add title
plt.legend(loc="upper left",fontsize=10)                            #Add legend
ax2.set_figwidth(10)                                                #set width dimention of plot
ax2.set_figheight(3.5)
plt.show()


ax3= plt.figure()                                                  #Create plot of counted frame
plt.plot(dts6,dfDmodbuf, label='modeloutput with buffer', linewidth=4)
plt.plot(dts6,dfDmod,linestyle='dashed', label='modeloutput without buffer', linewidth=4)
↪  #Add south data
plt.axhline(y = 875, color = 'r', linestyle = '-',label='max capcity', linewidth=4)
plt.xlabel("Time")                                                  #Add X label
plt.ylabel("BAX")                                                   #Add Y label
plt.xticks(rotation = 90,fontsize=10)                               #Turn X data ticks
plt.yticks(fontsize=10)                                             #Change fontsize ticks
axes = plt.gca()
axes.set_ylim([0,1000])
plt.xticks(np.arange(3, 92, 4))                                     #Reduce ticks
plt.title("BAX infeed D per quarter hour (15-07-2019)" , fontsize=20)   #add title
plt.legend(loc="upper left",fontsize=10)                            #Add legend
ax3.set_figwidth(10)                                                #set width dimention of plot
ax3.set_figheight(3.5)
plt.show()
```