Modeling Taxi Delay for Sustainable Aircraft Towing at Hub Airports using Queue-based Discrete Event Simulation

by

Yvor van den Beuken

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on the 29th of August 2024.

Student number:	4585887	
Project duration:	Jan 2022 – Aug 2024	
Thesis committee:	Chair:	A. Bombelli
	Supervisor:	Ir. P.C. Roling
	Examiner:	Dr. O.A. Sharpans'kykh

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

Acknowledgements

Almost nine years ago, when I visited the TU Delft campus to figure out where I wanted to study after high school, I immediately knew Delft was the place for me. Even though I have always been fascinated by aircraft and aviation, I decided to pursue a Bachelor in Mechanical Engineering instead of Aerospace Engineering. I figured Mechanical Engineering would give me a broader education, and I could always specialize in aerospace later by choosing a master in that field. Looking back, I think it was the right choice because I enjoyed every subject and continue to use the knowledge I gained about mechanics and the laws of nature in my daily life. This is driven by my enduring curiosity into how things work, especially behind the surface.

My love for aerospace never went away, though, so after finishing my bachelor, I started looking into the different Master tracks in Aerospace Engineering. Together with my good friend and current roommate, Thomas Hermans, we decided —after doing our Bachelor eind project (BEP) together— that we were both interested in the Air Transport & Operations track. The first year of the master program confirmed that aerospace was the right path for me because I really enjoyed all the courses. I liked it so much that I managed to finish all the coursework in the first year. Then came the second part of my master: my thesis.

Admittedly, working on my thesis was a lot more challenging than finishing the masters courses. Starting the process during the COVID-19 pandemic did not help either. The thesis, including the literature study, took longer than I had planned. At the same time, I was working part-time at a FinTech company called Yellowtail Conclusion, a job that eventually turned into a full-time position over the last year. Looking back, would I have taken on that job while working on my thesis? Probably not, but I am glad I earned some work experience already before graduating. I learned that doing research is tough when you only have evenings and weekends to work on it.

I was lucky to have my roommates, Lauren Vierhoven and Thomas Hermans, who are also two of my best friends, to support me. They were always there to brainstorm with me on different aspects of my thesis and, more importantly, helped me stay positive throughout the process. When progress is slow, it can be hard to stay confident about what you have achieved and what is next, but they really helped me keep going. I also want to thank my parents, Henk and Marjolein, for their frequent weekend trips from Venlo to Amsterdam, where I currently live. I was often too busy to visit them in Venlo, but they stood by me the whole time, even when I kept postponing my graduation deadline.

Finally, a huge thank you to my supervisor, Paul Roling, who was with me every step of the way during this thesis. Over the past two and a half years, we have had meetings almost every two weeks to discuss my progress. These meetings were supposed to be about the thesis, but we often ended up chatting about our shared interests like Formula 1, traveling, and other non-thesis related topics. I'm really grateful for all the knowledge Paul shared, the solutions he provided to the challenges I faced, and his patience at times when my progress was slow.

Looking back on my time in Delft, I have learned so much, not just in terms of knowledge gained from the courses that where taught to me, but also personally. I have figured out what subjects I am passionate about, what skills I am good at (and not so good at), and I have made a lot of great friends along the way. I am proudly taking all of this with me as I move on to the next phase of my life, which is starting my career.

Yvor van den Beuken
Delft, August 2024

Contents

List of Abbreviations 7

1 Introduction 9

I Scientific Paper 1

II Literature Study
previously graded under AE4020 23

III Supporting work 45

1 Appendix 1 - DES output data 47

2 Appendix 2 - Python Source Code 57

List of Abbreviations

$\#A$	Number of arrivals
$\#D$	Number of departures
$\#T$	Number of tow-trucks
μ_{Ts}	Mean service time for coupling an decoupling
σ_A	Input arrival frequency variance
σ_D	Input departure frequency variance
σ_{Ts}	Standard deviation service time for coupling and decoupling
τ^f_{dly}	Total taxi idle time of flight f
τ^f_{mov}	Total taxi moving time of flight f
τ^f_{ser}	Total taxi service time of flight f
τ^f_{tot}	Total taxi time of flight f
dly	Average total taxi idle time
mov	Average total taxi moving time
ser	Average total taxi service time
tot	Average total taxi time
CF	Taxi-speed corner factor
DRT	Departure runway throughput
f_A	Input arrival frequency
f_D	Input departure frequency
Lim_s	Parallel service limit
RD	Relative delay
RS	Runway separation
TR	Towing ratio
TS	Taxi separation
V^k_N	Straight line taxi speed (per vehicle type k)
V^k_T	Straight line towing speed (per vehicle type k)
ABM	Agent-Based Modelling
AGPS	Augmented Ground Propulsion System
APU	Auxilliary power unit
ATC	Air Traffic Control

DES	Discrete event simulation
EGTS	Electric Green Taxxing System
LTO	Landing and Take-off
OOP	Object oriented programming
TAT	Turn around time

1 Introduction

In front of you is a thesis report on a study into the effective implementation of operational towing at hub airports. This study focuses on identifying the challenges associated with using tow trucks to move aircraft to and from the runway during taxi operations. The study was conducted as part of obtaining a Master of Science degree in Aerospace Engineering at Delft University of Technology.

The study starts with a thorough literature review, looking into conventional aircraft taxiing methods and eco-friendly alternatives like operational towing. This review helped to pinpoint the key challenges that are slowing down the adoption of operational towing as a standard eco-friendly taxiing solution in the aviation industry. In the next phase, the research was guided by these challenges, aiming to provide insights that could help overcome some of them. The study includes a simulation method for modeling aircraft towing operations at hub airports and uses this simulation to understand how it affects the airport's operational efficiency.

This thesis report is divided into three main sections. Part I presents the scientific paper that was written. The paper starts with a summary of the literature review, outlines the research objectives, and explains the methodology, including the creation of a realistic simulation model. This model is then used to generate insights into the operational efficiency of towing. The paper wraps up with a discussion of the results, tying them back to the research objectives, and offers recommendations for future research.

Part II contains the literature review that was done before the research. It covers conventional taxiing operations and looks into different eco-friendly taxiing methods. It also discusses the challenges these new methods bring. The literature review also includes information on discrete event simulation and queuing theory as possible simulation approaches.

Lastly, Part III provides additional information in two appendices. Appendix A includes the full dataset of results from the simulation model, and Appendix B provides the Python source code used to build the simulation.

I

Scientific Paper

Modeling Taxi Delay for Sustainable Aircraft Towing at Hub Airports using Queue-based Discrete Event Simulation

Yvor van den Beuken,*

Delft University of Technology, Delft, The Netherlands

Abstract

This study addresses the operational challenges of sustainable aircraft towing at hub airports, particularly focusing on taxi delays. Using a queue-based discrete event simulation, airport taxi operations are modeled to evaluate the impact of tow-trucks on taxi delays and runway throughput. Key parameters include service times for coupling and decoupling, tow-truck availability, and parallel service capacity. The findings reveal that taxi delays and runway throughput are significantly affected by the mean and variability of service times. Allowing multiple parallel service stations in front of the runway can reduce delays but may require infrastructure modifications. The study concludes that while operational towing can reduce emissions, its successful implementation depends on careful management and optimized traffic strategies to avoid compromising airport efficiency.

1 Introduction

In recent years, the global demand for sustainability has intensified, reflecting a collective consciousness about the environmental impacts of various industries. This growing awareness has put significant pressure on the aviation industry to adopt greener practices, as it is a notable contributor to carbon emissions. Airlines and airports are now more committed than ever to finding innovative solutions to reduce their environmental footprint, aligning with global sustainability goals. This has led to the exploration of eco-friendly taxiing techniques, such as sustainable aircraft towing.

Despite the potential environmental benefits, the adoption of sustainable aircraft towing methods faces several operational challenges. The primary problem this paper addresses is understanding and managing taxi delays associated with operational towing, particularly at hub airports where traffic density is high. The introduction of tow-trucks for aircraft taxiing introduces new variables such as coupling and decoupling times, availability of tow-trucks, and potential congestion. This study aims to provide insights into these challenges by modeling and testing the implementation of operational towing systems during standard airport operations by using a queue-based discrete event simulation (DES) framework to model airport taxi operations. The simulation considers various parameters such as service times for coupling and decoupling, tow-truck availability, and available spaces for simultaneous coupling and decoupling. By simulating different scenarios bottlenecks are identified and the impact from operational towing on taxi delays and runway throughput are measured.

section 2 starts by summarizing existing research on eco-friendly taxiing solutions, their benefits, and the operational challenges they pose. The problem statement is then defined, highlighting the specific issue addressed in this research which is the need for insights into the operational efficiency of operational towing at hub airports. The methodology section details the simulation framework used, including the design of the simulation environment, the parameters tested, and the key performance indicators used to measure the performance. The results section presents the findings from the simulation, focusing on taxi delays, runway throughput, and the towing ratio under various scenarios. Finally, the conclusion discusses the implications of the results for the adoption of operational towing systems, the potential need for infrastructure modifications, and recommendations for future research.

2 Literature Review

This chapter summarizes the key take-aways that were found in existing literature on eco-friendly taxiing and concludes its results which have led to the definition of the problem statement in section 3 and methodology section 4 for this research.

*Msc Student, Air Transport and Operations, Faculty of Aerospace Engineering, Delft University of Technology

2.1 Introduction and motivation behind the study

Over the last decades, greenhouse gas emissions have been growing exponentially, significantly contributing to climate change its related negative effects on the environment. The aviation sector is responsible for a substantial share of these emissions and is therefore continuously looking to adopt more sustainable practices in all possible areas of aviation. One of these areas is aircraft taxiing. Conventional taxiing relies on the main aircraft engines operating at low speeds, resulting in significant fuel consumption as the aircraft engines are not optimized to operate at these speeds. The result is a lot of wasted energy and emissions. Eco-friendly taxi solutions aim to reduce or even mitigate these emissions from aircraft that are taxiing to and from the runway. These solutions are an attractive alternative to conventional aircraft taxiing, however a lot of uncertainties still exist that slow down the adoption in the aviation sector. This literature review aims to highlight existing challenges or uncertainties regarding eco-friendly taxiing that, once investigated properly, can contribute to the adoption if these solutions in the industry.

2.2 Background information on Conventional Aircraft Taxiing

Conventional taxiing involves the movement of aircraft on the ground using their main engines, a process that the engines are not optimized for. This method leads to high fuel consumption, increased wear on landing gear brakes and substantial energy waste. The typical landing and take-off (LTO) cycle includes two taxi phases: travelling to the gate after landing (taxi-in) and travelling from the gate to the departure runway (taxi-out).

Landing and Take-off Cycle
After landing, the aircraft exits the runway and taxis to the gate under its own power. The turnaround process begins at the gate, involving passenger disembarkation, cargo unloading, cleaning, refueling, and inspections. Once ready for departure, a push-back truck moves the aircraft from the gate to a position where it can taxi to the departure runway. During push-back, the pilots start the engines, which requires 3-5 minutes for warm-up. The aircraft then taxis to the departure runway, following a route assigned by Air Traffic Control (ATC).

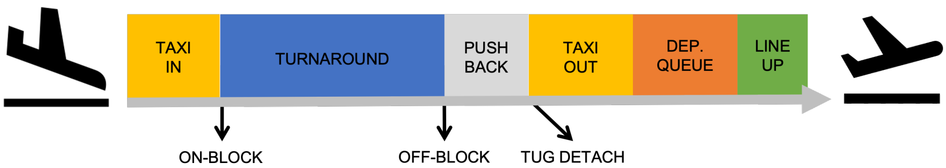


Figure 1: Conventional turnaround-process sequence

Separation Requirements
Safety regulations mandate minimum distances between aircraft on the ground and in the air. Taxi separation ensures aircraft maintain safe stopping distances, while airborne wake vortex separation prevents turbulence interference during landing and take-off. The FAA defines separation standards based on aircraft weight classes, which influence the intervals between departures and arrivals.

2.3 Eco-friendly Taxiing Solutions

The need for more efficient taxiing methods has led to the development of several eco-friendly taxiing solutions, classified into three main types: Single-engine taxiing, internal electric taxiing and external aircraft taxiing. The latter is also referred to as operational towing.

Single-engine Taxiing
This method uses only half of the aircraft's main engines, reducing fuel consumption and emissions. It also slightly increases taxi time due to slower speeds and required additional warm-up time for the unused engines.

Internal electric taxiing
Electric motors in the aircraft's landing gear, powered by the auxiliary power unit (APU), eliminate the need for main engines during taxiing. Examples include systems like Wheeltug and the Electric Green Taxiing System (EGTS). Challenges include managing the thermal behavior of the motors and the additional weight, which can offset fuel savings during flight.

External AGPS (operational towing)
Here tow-trucks are used to move aircraft between gates and runways, completely eliminating the use of the

main engines for taxiing. Systems like TaxiBot offer potential for zero-emission taxiing, depending on the fuel source for the tow-trucks. The main operational challenges are increased taxiway congestion and the complexity of handling multiple tows, especially during peak departure times.

2.4 Imposed operational challenges of operational towing

While Single-engine taxiing and internal electric taxiing both offer minimal operational challenges on the airport environment they also offer the least benefit in terms of limiting fuel emissions from taxiing as both require the use of the aircrafts main engines.[Re, 2012] Either via the single-engine taxiing or using the APU for internal electric taxiing. Implementing operational towing provides the highest ability in potential fuel savings, but introduces significant challenges to the effective handling of airport operations[Guo et al., 2014].

Availability of Tow Trucks
One primary concern is the availability of tow trucks. If the number of available tow trucks is insufficient, it can lead to increased delays as aircraft wait for a tow. This delay can be mitigated by optimizing the number of tow trucks in operation, ensuring there are enough to handle peak traffic periods without causing excessive idle time for the trucks.

Coupling and Decoupling Times
The process of coupling and decoupling tow trucks from aircraft is another important factor. These operations take time and, if not managed efficiently, can cause bottlenecks, especially in high-traffic areas. Implementing standardized procedures and training for ground staff can help minimize these times. Additionally, scheduling algorithms can be developed to optimize the timing of these operations, ensuring they occur in a manner that least disrupts overall traffic flow.

Traffic Flow Optimization
Optimizing traffic flow is crucial to manage the increased complexity introduced by tow trucks operating alongside traditional taxiing aircraft. This includes developing advanced routing algorithms that consider current traffic conditions, predicted traffic, and the availability of tow trucks. These algorithms can dynamically adjust routes to minimize congestion and ensure smooth traffic flow.

Queue Management at Departure Runways
At departure runways, managing the queue of aircraft waiting to take off is particularly challenging. The presence of tow trucks requires additional coordination to ensure they do not block taxiways and cause delays for other aircraft. Implementing holding areas where aircraft can wait for their tow trucks to decouple before proceeding to the runway can help manage this challenge. Additionally, precise scheduling and real-time adjustments can ensure that tow trucks and aircraft move in and out of these areas efficiently.

Safety and Separation Standards
Maintaining safety standards is paramount when integrating tow trucks into airport operations. Ensuring that tow trucks and aircraft maintain appropriate separation distances is critical to prevent accidents. This can be achieved through real-time monitoring systems and automated alerts that notify ground control of potential safety breaches.

Adaptations to airport infrastructure
In some cases, physical changes to airport infrastructure may be necessary to accommodate the increased use of tow trucks. This could include expanding taxiways, creating dedicated lanes for tow trucks, or building additional holding areas. Such adaptations would require careful planning and investment but could significantly improve the efficiency and safety of operations.

Environmental Impact Considerations
While the primary goal of operational towing is to reduce fuel consumption and emissions from aircraft, it is essential to consider the environmental impact of the tow-trucks themselves. Ensuring that these vehicles are powered by renewable energy sources and are as efficient as possible is critical to achieving overall sustainability goals.

Economic Feasibility
Finally, the economic feasibility of implementing operational towing on a large scale must be considered. This includes analyzing the costs of purchasing and maintaining tow trucks, training staff, and potentially modifying infrastructure. These costs must be weighed against the expected savings from reduced fuel consumption and emissions to ensure the strategy is viable for airports and airlines.

Soltani Et al.[Soltani et al., 2020] proposes a system with a hybrid towing solution. In this system, part of the taxi operations are handled by a tow-truck using renewable energy while the other part still does regular taxiing using their own engines. But Soltani et al. also mentions that the approach is deterministic and disregards factors such as decoupling and coupling times of the tow-trucks.[Soltani et al., 2020]. van Baaren [Baaren, 2019], studied the feasibility and performance of a fully electric operational towing system by making a proof of concept. This study assumes an ideal system in which all vehicles travel according to their schedules and disregards traffic interaction at busy taxiways and therefore delay and taxi congestion in both the scheduling solution as well as the routing problem are not considered. The process of coupling and decoupling tow-trucks from aircraft is a critical factor. These operations take time and, if not managed efficiently, can cause bottlenecks, especially in traffic dense areas. Delays can propagate and effect other flights as well.

2.5 Conclusion of the literature study

The literature study concludes that while external AGPS systems, such as operational towing, present a promising eco-friendly taxiing solution, their successful implementation requires addressing significant operational challenges. Further research and development of traffic management strategies are necessary to ensure these systems can be effectively integrated into airport operations, minimizing delays and maximizing fuel savings.

3 Problem statement

As described in the introduction, this research aims to accelerate the adoption of eco-friendly taxiing methods in the aviation industry, promoting a shift towards more sustainable practices. An extensive literature review was conducted to identify key topics and unresolved issues related to these eco-friendly taxiing methods. The primary objective of this research is to address at least one of these outstanding problems.
Three popular techniques regarding eco-friendly taxiing are often discussed in existing literature considering eco-friendly taxiing. These are single-engine taxiing, internally powered electric taxiing and externally powered electric taxiing. From these three options the latter seems to be the most promising in its ability to reduce emissions from aircraft taxiing. This option is referred to as sustainable aircraft towing because aircraft are towed by electric tow-trucks. The drawback of sustainable aircraft towing however, is its impact on the airports traffic-density. Given that taxi-time heavily relies on traffic density, the introduction of tow-trucks into the the airport environment raises concerns about potential taxi delays. Studies on sustainable aircraft towing are limited and the studies that do investigate this method of eco-friendly taxiing do often not consider key processes such as decoupling from an aircraft in front of the runway in detail. The impact of these processes on the resulting taxi-delay and runway throughput are still unknown. To comprehensively understand and mitigate these concerns, it is crucial to model and test the full implementation of operational towing during standard taxi operations.

Resulting from the literature study, it was found that there is not yet enough insight in the resulting airports operational efficiency after adopting fully integrated sustainable aircraft taxiing system. In an attempt to contribute to easier adoption of sustainable aircraft towing in the aviation industry the following objective is defined:

"***To provide insight into the impact of sustainable aircraft towing on the operational efficiency at hub airports.***"

The objective can be realized by creating a realistic simulation of airport taxi-operations at hub airports and analyze resulting taxi-delays. Therefore the simulation should be able to track taxi-delay and runway throughput. An emphasize here must be on modeling the characteristics of coupling and decoupling from the tow-truck as well as relocation and allocation between tow-trucks and flights. Simulating operational towing method in this way will will help identify the relevant bottlenecks and reflect on the best possible strategies to minimize the impact on taxi-delay and runway throughput which contributes to the research objective. Queue-Based Discrete event simulation can be used to empirically retrieve the underlying distributions of taxi-delay and runway-throughput as a result from different inputs. The conclusion of this research will reflect on the impact from different model parameters and how changing them influences the airports operational efficiency. A good understanding of conventional taxi operations as well as what changes when introducing sustainable aircraft towing is required while designing the simulation. The research starts with the design of the simulation environment and characteristic. This also includes defining the scope. After this, various important model parameters are defined should reflect the uncertain parameters from aircraft towing. These will be used to test different scenarios for sustainable taxiing. The conclusion of this research will reflect on and summarize the relation between the various parameters and the impact on the operational efficiency of the airport.

In summary, the problem addressed in this research is the need to understand and manage taxi delays resulting from the adoption of sustainable taxiing methods at hub airports. Addressing this issue is essential to ensure that the environmental benefits of sustainable taxiing do not come at the expense of airport efficiency and airline on-time performance.

4 Methodology

4.1 Object-Oriented Programming Framework

The general aim of this research is to evaluate the performance of an operational towing system, which requires analyzing various strategies and accounting for uncertain parameters. A fitting simulation method must be used in order to retrieve the required insights in the systems performance without over-analyzing it and thereby potentially increasing computing time unnecessarily. The operational taxiing problem can be defined by macro-specifications such as average decouple times, flight intensity, number of tow-trucks and more. These specifications apply to the overall system and do not directly specify the behaviour of individual actors in it. Resulting from the simulation is the performance of the operational taxi system, which can be assumed the output. This output, which can be brought down to a series of events plotted on a timeline for all actors in the system, is expected be influenced by these macro-specifications. This top-down approach fits well with Discrete event simulation (DES) which is why DES was chosen over other methods such as Agent-Based modeling (ABM) which use a bottom-up approach [Baldwin et al., 2015]. DES is also more computationally efficient. The DES model is realized using Salabim. Salabim is a discrete event library in Python specialized in object-oriented modeling and queue analysis.

The operational towing network considers two different components: moving components and non-moving components. Moving components can be seen as the users of the system. These components consume service at different service stations. This includes the aircraft and tow-trucks. The non-moving components are the handlers of the system. They are responsible for maintaining all model rules as well as providing service to the moving agents. These non-moving components include the runways, taxiway junctions as well as couple and decouple stations.

Every component type has a specific set of defined attributes and methods associated with them. The value of these attributes collectively determine the components state which changes over time. The methods represent functions to make calculations based an components state and based on define the attributes for the next event. The method of one component can change the attributes of itself and other components as well as schedule itself or other components to execute their methods in the future. The scheduling of one component by another is the basis of the discrete event simulation.

In the discrete event simulation, every event is driven by one single component. In that instance the component is current. All other components are passive at that time. When the component has finished its tasks for that event, the component turns passive. After which the next scheduled component becomes current again. In a repeating fashion, the discrete events are simulated one by one. Future events are always scheduled by the actions of current events. Only when there is no future event scheduled anymore, the discrete event simulation is finished. Figure 2 shows this principle schematically.

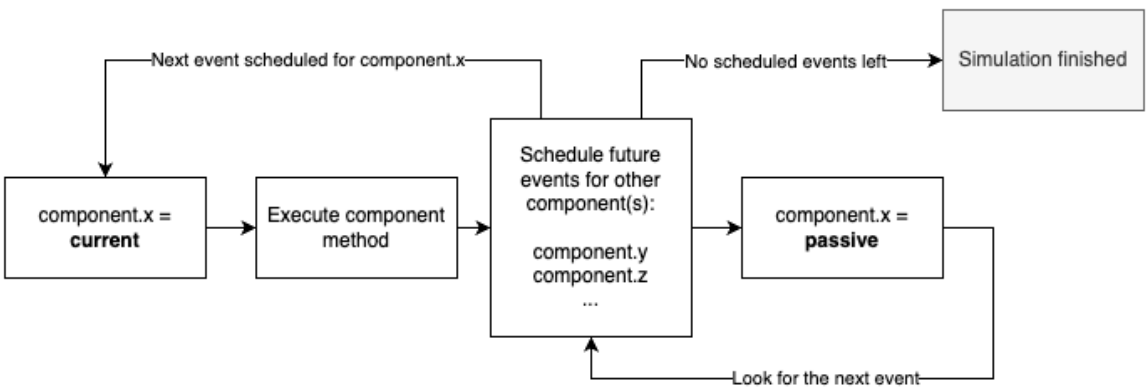


Figure 2: Single discrete event of the OOP framework

4.2 Environment

The basis for the discrete event simulation model is the environment that it is simulating. This includes the visual representation of the airport including runways, gates and taxiways as well as the vehicles. Because one of the goals of this research is to investigate a generic situation rather then a case study fitted to a specific airport, a fictitious airport situation is used. This section discusses the assumptions and design of the simulation environment.

4.2.1 General assumptions

Scope
When considering the turnaround cycle for an aircraft, all activities happening while the aircraft is at the gate have no direct impact on the taxi-delay of an aircraft and are therefore disregarded in the simulation. Departures enter the environment apron, ready to travel to the runway and leave the environment after runway entry. For Arrivals this is the other way around. They enter the system when they leave the runway and leave the system when they arrive at the apron.

Hub airport
Operational towing is especially effective at airports with long taxi-routes to the runway. This characterizes mostly hub airports that have multiple runways and longer distances in general [EUROCONTROL, 2023]. As these type of airports are most likely to introduce operational towing, the environment in this research matches the layout characteristics of multiple hub airports. This includes long taxi-routes between gate and runway, multiple runways for separate departures and arrivals and multiple taxiway connections between the runway and the gates.

Parallel runways
Two parallel runways are used, one for departures and one for arrivals. This way incoming and outgoing flights can enter and leave the system independently from each other as they are not joining the same runway queue.

4.2.2 Assumptions related to operational towing

Out of the many challenges that operational towing brings when implemented throughout an airport, two have been identified to be investigated by this research. Firstly the challenge of returning tow-trucks after delivering aircraft either to the runway or gates. The second challenge is decoupling and coupling of tow-trucks near the runway. The model environment is designed to service both these challenges. For this a couple of assumptions are made.

Returning tow-trucks
When considering for instance departing aircraft, the decoupled tow-truck that is returning after having delivered an aircraft to the runway forms a challenge as this truck needs to return and leave the runway site. Ideally, decoupling the tow-truck is done as close to the runway as possible in order to maximize the effectiveness of operational towing. Returning in opposite direction on the same runway entry road is not possible when there are aircraft behind it as it cannot pas these aircraft safely. One possibility is to have the tug wait in the runway queue and leave via the runway by taking via another exit. This method however is not desired as this will significantly limit the runway throughput. Ruling out this option as well as the option to return on the same road, another solution was chosen. A separate service road next to the runway entry taxi road that can process returning tow-trucks from the runway. This road leads to the first junction of the original runway entry road it came from. A visualization of this situation is shown in Figure 3.
The same solution works for departing tow-trucks that can enter the runway exit freely and couple with an arriving aircraft. At the apron this challenge does not hold because it is assumed that there is enough room at the apron for service roads.
At the taxiways in between the runway and the gates, the returning tow-trucks form less of a challenge. Assuming there is more than one taxiway connection between runway and gate and that these different routes are dedicated for arriving or departing traffic the tow-trucks can take one of these routes that matches its direction. Still, having tow-trucks travel back and forth between the gate and runway while not pulling an aircraft is not desired as this limits the total utilization of the tow-trucks. In order to maximize the tow-trucks utilization, tugs preferably combine towing-out departing aircraft and towing-in arriving aircraft on their way back. After delivering an aircraft to the runway tugs will travel parallel to the runway towards to the other end and wait for an arrival to bring it back in. This system is explained visually in Figure 4. The airside and landside refer to the runway and gate areas respectively. Also buffers are represent at the pick-up locations where tow-trucks can wait and potentially charge. Given the scope of this research however, the charging and energy consumption of electric tow-trucks is not considered here. When looking at this strategy from a flow balancing perspective

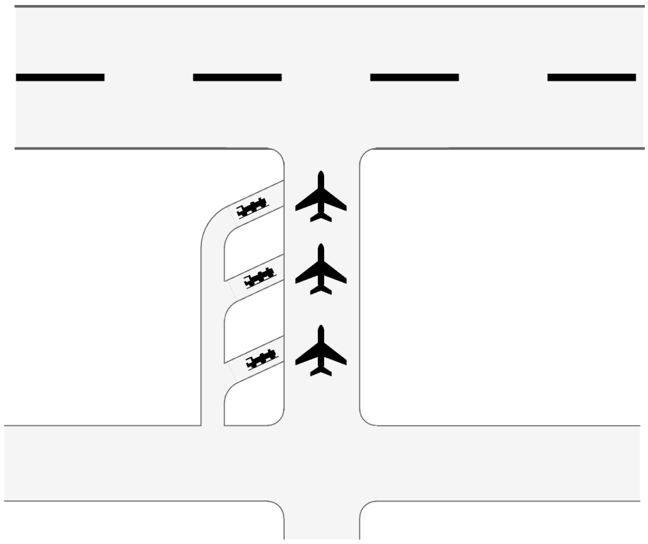


Figure 3: Tow-truck service road near runway entry

it also makes sense because on average the number of incoming a outgoing aircraft at any given airport must be the same. The buffers should capture any momentarily differences between arrival and departure rates that happen during the day. When these differences are becoming too big for the buffers, a tow-truck can always travel back without an aircraft along the same route. Two important assumptions however have been made to validate this method. It is assumed that there is a parallel taxiway next to the runway that the tugs can use in order to travel from one end to the other end of the runway. This assumptions is often true at hub airports. It is also important to note that even though the flow balance between incoming and outgoing flights must hold for the net traffic of the entire airport, it does not necessarily hold when considering a smaller portion of the airport as described here. These local imbalances could be covered by the tow-truck buffers.

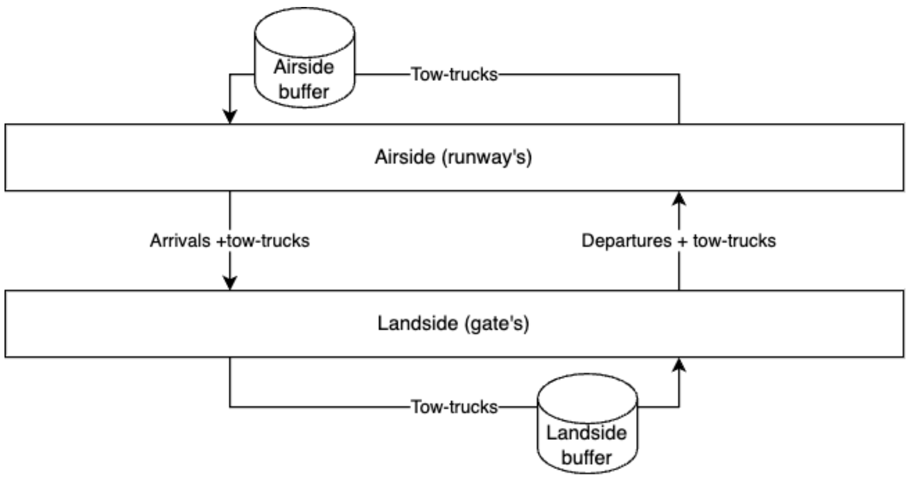


Figure 4: Circular traffic in airport environment

Queue behaviour for decoupling and coupling of aircraft near the runway

In the literature study it was found the coupling and decoupling of aircraft in front of the runway has not been investigated in detail in previous studies, but is expected to have a major influence on the taxi delay [Guo et al., 2014, Lukic et al., 2018, Lukic et al., 2019]. Coupling and decoupling of the aircraft takes some time during which they cannot move and block other aircraft block other traffic as well. Given the expected influence on taxi-times of aircraft, simulating this process in detail is important. Unless specific couple or service spots are available, it is assumed that the aircraft attach or detach from the tow-truck on the taxiway itself. As aircraft cannot pass each other here, aircraft that are in service block all traffic behind them. When considering the departure runway queue, conventionally all aircraft in the queue move up one place as soon as the runway is available for the aircraft at the head of the queue. However, now only aircraft in the queue can only move up one place when the aircraft in front of them is finished with service. This induces a lot of lost time and space. When an aircraft arrives at the tail of an existing runway queue and starts getting service at that position immediately, it essentially creates a new queue with its head at that position. This way the runway queue could propagate to the back indefinitely. At some point aircraft that enter the tail of the queue should not start decoupling immediately but instead wait for their predecessors to finish, move to the front and then start decoupling in order to reset the queue to its original head position. Having dedicated parallel decoupling spots can help overcome this issue. *However, except for de-icing pads, most airports do not have the required spaces available.*

4.2.3 Visual representation

The airport layout is derived from the Northern part of Paris Charles de Gaulle airport and considers runways 09R/27L and 09L/27R (shown in Figure 5. Runway 27L is used for departures and runway 27R is used for arrivals. Arrivals have to cross the active runway 27L before getting to their coupling station. Vehicles travel from one node to the other, displayed by the white dots. In between nodes, the taxi-speed of vehicles is assumed to be for constant for aircraft of the same type. The distances between the nodes are measured using google maps. This layout should fit all the requirement and assumptions explained in section 4.2.1

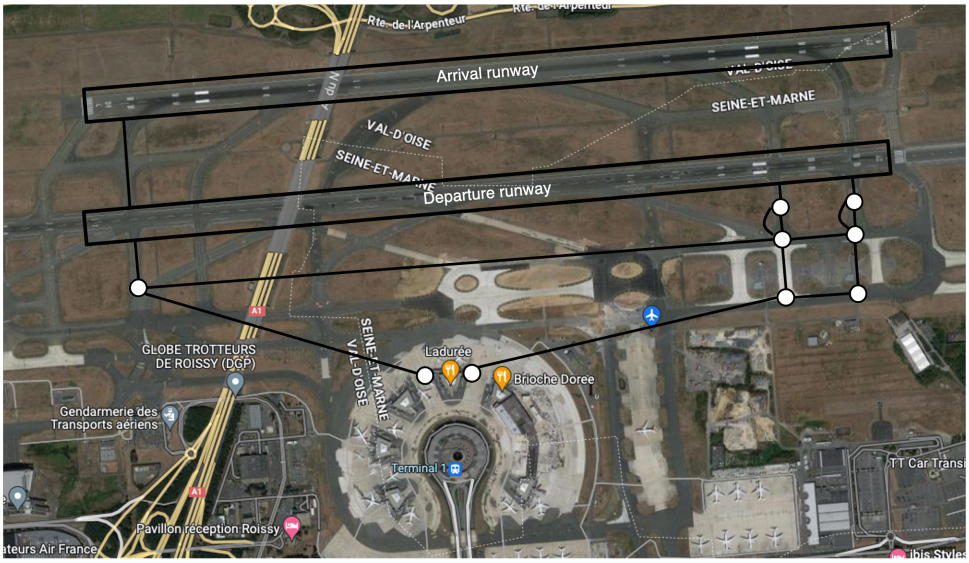


Figure 5: Satellite shot of Charles de Gaulle Airport including runways 09R/27L and 09L/27R

4.2.4 Schematic representation

The network environment consists of a set of 10 nodes. Nodes can either represent an junction, tow-truck couple or decouple station or the departure runway. Every node maintains its own distinct queue and after that delivers service. All vehicles enter its queue before being processed by that node so that the node is able to maintain the right amount of vehicles passing through it. Since arrivals only enter the environment after they have landed and exited the runway, no node exists for the arrival runway. Also see Figure 6, in the picture only the departure runway (7) is present. The first interaction between arrivals and the environment is crossing the active departure runway (7). Hence, arrivals enter the queue from node 7 upon arrival.

Junctions
Nodes 1, 2, 3 and 5 are all junctions. The junction nodes make sure that only one vehicle at a time passes through it. It also maintains the taxi separation between vehicles travelling in the same direction. Given that the discrete event simulation does not track the position of vehicles when they are travelling between nodes, vehicles with different speeds could close in on each other. The next node can reset this separation once again.

Couple stations
Pairing aircraft with tow-trucks is done at node 0 for departures and at node 8 for arrivals. These nodes are linked to their unique tug-truck buffer where available tow-trucks are waiting to be paired. The couple station nodes have two responsibilities: making sure that an aircraft is paired with an available tow-truck from its buffer, and providing the service, that is the actual coupling of the aircraft to a tow-truck. When the buffer is empty, aircraft in the queue of the respective node must either wait for a tow-truck to become available or continue taxiing without on their own engines. Because departing aircraft can be coupled to a tow-truck while

at the gate and given the departures only enter the simulation environment after they have left the gate, it is assumed that at node 0, no service-time is added for aircraft that are being towed. This makes the simultaneous couple capacity for this node effectively unlimited, which given that departures are coupled at their gate, is true. In reality checking the availability of a tow-truck after it should have been coupled already is not possible. However, for the purpose of this simulations this makes no difference. The capacity of the amount of tow-trucks that are available at the apron is the only limiting factor here. It is required to check the capacity for available tow-trucks for every departing flight at least once. As long as instance when the capacity is checked is the same for all departures of the simulation, the effect will be the same no matter at when in the cycle that is.

Decouple stations
The decouple stations work in the same way as the couple stations. However, checking capacity for an available tow-truck is not needed at the decoupling station. The decouple nodes are 4, 6 and 9. The decouple nodes check whether an incoming aircraft is towed or not. When it is, decouple the flight from the truck. Once the process is finished the tow-truck moves to the appropriate tow-truck buffer and the aircraft continues its path. Flights that are not towed have service-time 0, however should still wait in the queue because they are not able to pass other flights in front of them. In the same way as was the case for coupling at node 0, the decoupling at node 9 is assumed to be done at the gate. Since this will not cost any effective time because the gate activities can start simultaneously, the service-time for aircraft decoupling at node 9 is always 0. For the tow-trucks that are decoupled at node 9 however, this is not the case. The need to wait for decoupling to become available again so the service-time here is applied for the tow-trucks.

Runways
It was already mentioned quickly in the introduction of this paragraph that there is no need for an arrival runway node. No queue needs to be maintained here as the arrival time of incoming flights are assumed to be an input to the simulation and are not affected by the results or behaviour of the simulation itself. After aircraft have landed at the arrival runway, the need to cross the active departure runway. Here a queue management is needed. The departure runway is identified by node 7 in the schematic representation of Figure 6. The runway node is used to manage the users of that runway. In the simulation this can either be the departing flights, for which standard separation intervals have been defined (Table 1), or the arrivals that need to cross the runway on the other side. The runway handles its queue on a first-come-first-serve (FCFS) basis which means that departing flights do not have priority over arrivals needing to cross the runway.

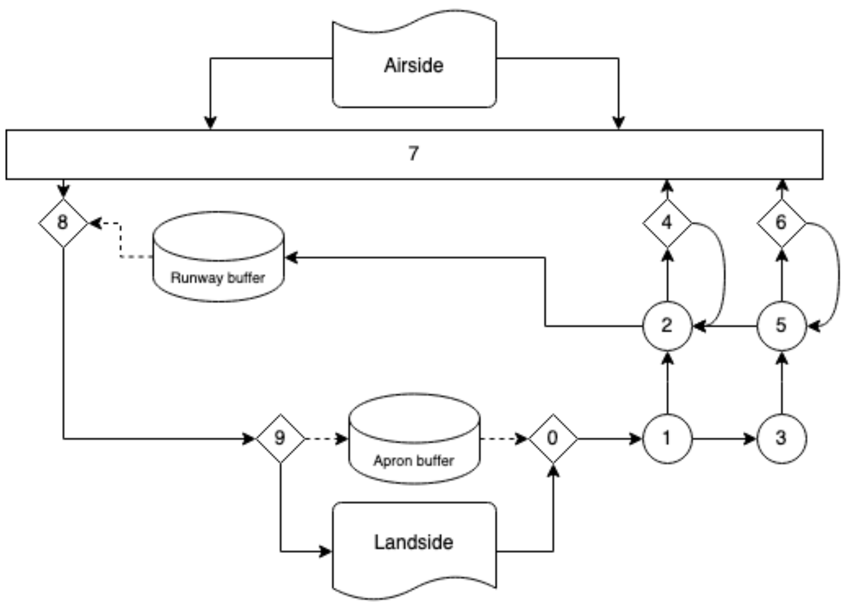


Figure 6: Schematic representation of airport environment. The nodes are numbered with their unique ID

4.3 Model parameters

The characteristics of the simulation are defined by its parameters. This section describes all determined parameters and inputs per subject.

4.3.1 Input & constants

The input & constants section describes all parameters that remain the same throughout the simulation and its multiple iterations. An overview of these values can be found in Table 2.

Flight intensity
The simulation starts when arrivals and departures enter the system at the first node. The input of traffic is defined by the frequency at which arrivals and departures enter the system on average: f_D and f_A respectively, as well as a standard deviation to this factor σ_D, σ_A to include some randomness. The goal is to test the effect of operational towing on the maximum runway throughut. Therefore it was chosen to have the input frequency of arrivals and departures match the effective maximum runway capacity as closely as possible. For a departure runway this is mostly dependent on the mix of aircraft using the runway and their respective separation due to wake turbulence. A value of 35 arrivals/departures an hour was chosen, which is a good estimate for achieved optimal departure throughput based on empirical data from CDG airport [DECDEA, EUROCONTROL, 2021]. The standard deviation is equal 60% of this value for departures and 80% for arrivals.
One simulation iteration handles 300 consecutive flights for both departures and arrivals. Given that these flights arrive at a rate of 35 flights per hour, the simulation will roughly take $300/35 \approx 8.5$ hours. In total there will be 300 departures and 300 arrivals and thus 600 flights. Because runway separation and taxi-speeds can vary with the weight of the aircraft, a mix of three distinct aircraft weight classes is used. These three weight classes are based on the weight classes defined by the FAA for runway separation and are described in the section below. The mix is defined as 20% light aircraft, 50% medium and 30% heavy.

Taxi speeds
The average taxi speed for aircraft, as established by [Salihu et al., 2021, Guimarans et al., 2017], is 7 m/s. This was derived from a Boeing 737 NG flight crew training manual. This manual specifies a standard taxi speed of 20 knots (10.3 m/s), a straight-line maximum speed of 30 knots (15.4 m/s), and a reduced speed of 10 knots (5.1 m/s) during turns. It is widely accepted in aviation that these speeds apply uniformly for all aircraft. Because the model does not include acceleration an deceleration of the aircraft, a slightly lower taxi speed is used for heavy aircraft. The corner factor of 50% of the straight line taxi speed is assumed for all types of aircraft. The above discussed speeds hold for aircraft travelling under their own engine power.
Determining the taxi-speeds while being towed is a bit more challenging given the limited empirical data. The manufacturer of TaxiBot specifies that the TaxiBot can reach a nominal straight line taxi speed of 23 knots (11.8 m/s) [TaxiBot, 2023] for most aircraft.
Next to the three aircraft weight classes, a fourth category is added for determining taxi speeds which are the speeds of a tow-truck travelling without a paired aircraft. It is assumed that the tow-truck can drive the same speeds as it does when coupled to an aircraft. Table 2 shows the straight line speeds and towing speeds for the three aircraft types and the tow-truck.

Separation
Separation should be maintained both at the runway and during taxiing. Because of wake vortex turbulence the separation at the runway depends on the weight class of trailing and leading aircraft respectively. As the departure runway is used for departures only, only departure-departure separation intervals are considered. The intervals are defined in Table 1 as defined by the FAA [Federal Aviation Administration, 2014].
At the taxiways there is no wake turbulence but separation is still required for safe stopping. [Yang et al., 2017] defines an average taxi separation for straight line taxiing aircraft between 100 and 300 meters. Assuming an average distance of 200 meter and a nominal straight line average speed of 10 m/s, a 20 second taxi separation is defined and should be maintained between all vehicles. The junction nodes are responsible for maintaining this separation.

Leading aircraft cat.	Trailing Aircraft Cat.		
	Light	**Medium**	**Heavy**
Light	90	120	120
Medium	60	60	60
Heavy	45	45	45

Table 1: Departure-Departure separation in s

Description	Symbol	Unit	Value
Number of departures	$\#D$	1	300
Departure frequency	f_D	$^1/_{hour}$	35
Departure frequency variance	σ_D	$^1/_{hour}$	21
Number of arrivals	$\#A$	300	1
Arrival frequency	f_A	$^1/_{hour}$	35
Arrival frequency variance	σ_A	$^1/_{hour}$	28
Straight line taxi speed (per vehicle type k)	V_N^k	$^m/_s$	[15, 15, 10, 12]
Straight line towing speed (per vehicle type k)	V_T^k	$^m/_s$	[12, 12, 10, NA]
Taxi-speed corner factor	CF	1	0.5
Taxi separation	TS	s	20
Runway separation	RS	s	See Table 1

Table 2: Model constants

4.3.2 Parameters

In order to test different strategies and scenarios a set of parameters have been determined that can vary throughout different scenarios in order to investigate their expected impact on the performance of the operational towing system. These variables are defined by the an interval between which they can vary. Also as default value is defined that is used when the specific input variable is not the variable of interest and can remain steady throughout the multiple iterations. One of these factors is couple and decouple service times. The model considers a parameter for both the average time for coupling and decoupling (μ_{Ts}) as well as a standard deviation (σ_{Ts}) so that the impact on the variance of can be investigated. These values can vary for different iterations however are assumed to be equal between coupling and decoupling tasks.
Another interesting aspect is the amount of aircraft that getting parallel service when decoupling and coupling Lim_s. This can be represented by a limit. When the limit is equal to 1, this effectively means that only the aircraft that is in the front of the queue is getting service. Flights behind this flight will have to wait until they are first in line. Increasing this number to n will let aircraft start coupling or decoupling when they are the n-th in line. This means that simultaneous service is allowed which can shorten delays, however it not limiting n might result in an unbounded propagation of the queue to its back because aircraft that are finished service still have to wait for their predecessor to leave queue as they cannot pass each other. Increasing n is therefore also expected to lower the average utilization of the individual service spots in the queue.
Lastly the number of available tow-trucks ($\#T$) is an interesting factor. Because aircraft are only towed when a tow-truck is available, the resulting total share of flights that is towed will decrease when limiting the amount of tow-trucks. Setting $\#T = 0$ means that no aircraft are towed (conventional situation) while setting it to $\#T = \infty$ means that all aircraft will be towed.
An overview of the parameters and there respective interval and default values is shown in Table 3

Description	Symbol	Unit	Value range	Default value
Mean service time	μ_{Ts}	s	10-360	120
Standard deviation service time	σ_{Ts}	s	10-60	30
Parallel service limit	Lim_S	1	1-8	3
Number of tow-trucks	$\#T$	1	0-20, ∞	12

Table 3: Input parameters

4.4 Output

This section describes the output from the simulation, the key performance indicators (KPIs) that will be used in the results and the conclusion and how the output is used to calculate the KPIs.

4.4.1 Output parameters

One iteration of the simulation tracks the timelines of all 600 flights. A flights timeline is defined by tracking the following instances: The moment a flight arrives at the queue of node n: $tq^+_{f,n}$, the moment it enters service at node n: $ts^+_{f,n}$ and the moment it leaves node n: $ts^-_{f,n}$. Tracking this for all nodes n on the route of the flight forms the timeline. The route depends on flight f being a departure or arrival. From these timelines the interval times between important steps of taxiing can be retrieved. Table 4 shows the outputs per flight f. The averages for all departures and arrivals of these values are also calculated and will later be used to evaluate the results.

Description	**Symbol**	**Unit**
Total taxi time	τ^f_{tot}	s
Taxi moving time	τ^f_{mov}	s
Service time	τ^f_{ser}	s

Table 4: Simulation output per flight ($f \in F$) for one iteration

4.4.2 Key performance indicators

Delay

It is important to define the used interpretation of taxi-delay when calculating the resulting delay incurred by operational towing. When considering departures, the total taxi-time is often measured as the time between gate push-back and wheel-off time. Since the simulation model at hand does not consider the actual take-off procedure, the moment of runway entry is used as reference instead of the wheel-off moment for the total taxi-time. When predicting the taxi-out time, the total taxi-time can be divided in three separate values: the unimpeded taxi-time, any taxi-impedance due to e.g. congestion or bad weather at the airport and the remaining departure queue delay. Using the total taxi-out time output from the simulation the following expression can be written:

$$\tau_{tot} = \tau_{unimpeded} + \tau_{impedance} + \tau_{dly} \tag{1}$$

Here $\tau_{unimpeded} + \tau_{impedance}$ combined equals the total resulting travelling time for the aircraft between gate and runway queue. This value can be derived directly from the flights timeline τ_{mov} (Table 4). Also, Equation 1 refers to a standard taxiing scenario. An extra factor can be added to the equation when considering operational towing, which is the tow-truck decouple time τ^d_{ser}. Adding this value to Equation 1 and using τ^d_{mov} for the total travel time the expression can be rewritten as:

$$\tau_{tot} = \tau_{mov} + \tau_{ser} + \tau_{dly} \tag{2}$$

This leads to the following expression for the taxi-delay incurred by operational towing:

$$\tau_{dly} = \tau_{tot} - \tau_{mov} - \tau_{ser} \tag{3}$$

This will also be referred as the idle time as this value includes the moment that the aircraft is waiting in the departure queue while remaining idle, and thus is not being decoupled. It is important to note that when comparing operational towing taxi-times in relation to the conventional taxiing scenario, the service time τ_{ser} is in fact an extra time factor adding up to the total taxi-time and should be treated as such. However, in this study it is not defined as delay because these service-times are constrained as they cannot be influenced by the system itself. Also, the purpose of this research is to analyze the impact of operational towing on the departure queue delay rather then the total taxi-time.

For arrivals a similar derivation for total taxi-delay is used. The standard "departure" queue delay that is present for departures in the conventional way is not present for arrivals, nor is there an arrival queue after touchdown. However, with operational towing, an arrival-queue will form for coupling with tow-trucks near the runway. The same expression still holds for arrivals.

The average delay for all departures and arrivals can then be retrieved by taking the mean from all flights. Using the average delay for departures and arrivals the average relative delay is also calculated. The relative delay is defined as the total delay share compared to the total taxi time:

$$RD = \frac{\tau_{dly}}{\tau_{tot}} \tag{4}$$

The relative delay is used as one of the performance indicators.

Towing-ratio

Another important aspect when analyzing the performance is the number of aircraft that are towed by an electric tow-truck. The simulation does not constrain the amount of flights to be towed. Instead it lets aircraft be towed only when a tow-truck is available. An output from the model is thus the realized share of flights that are towed compared to the total number of flights.

$$TR = \frac{\#\text{flights towed}}{\#\text{total flights}} \tag{5}$$

Runway throughput
The effect on the max departure runway throughput is measured as well. The model uses an input traffic rate (per runway) of 35 flights per hour, which is meant to match a standard maximum runway throughput. The average number of departures/hour leaving the environment is considered the third KPI. This value reflects the resulting runway throughput. Given that the simulation always handles 300 consecutive flights, this value can be received by dividing the number of flights by the total time interval between the first and last flight leaving the departure runway.

$$DRT = \frac{300}{(ts^{-}_{(299,7)} - ts^{-}_{(0,7)}} \times 3600 \quad (flights/hour) \tag{6}$$

4.4.3 Determining the number of iterations

The above KPIs are the result from a single iteration of the simulation. As the simulation involves a lot of stochastic processes, running the simulation a n number of times is needed to acquire more accurate results. The central limit theorem (CLT) can be used for determining the required number of iterations in order to attain the desired precision. The CLT is applied only once for one of the output KPIs from section 4.4.2 and the resulting number of iterations is used in evaluating the other kpis as well. The total average departure delay $\bar{\tau}^{d}_{dly} = T^{D}_{dly}$ is used. Assuming that the different results for T^{D}_{dly} follow an unknown distribution with known mean μ and σ, then the the the CLT states that the the distribution of the random variable,

$$Z_n = \frac{\bar{T}^{D}_{dly} - \mu}{\sigma/\sqrt{n}} \tag{7}$$

follows the standard normal distribution. The margin of error $(\bar{T}^{D}_{dly} - \mu)$ is equal to:

$$\bar{T}^{D}_{dly} - \mu = E = \frac{z\sigma}{\sqrt{n}} \tag{8}$$

Here z represents the error margin for the standard normal distribution. For a 95% confidence interval, $z = 1.96$, μ and σ are the mean and standard deviation of output T^{D}_{dly} and n is the required number of samples. The values for σ is found by running the simulation an initial 500 times which led to a value of 305 seconds. Weighing in both precision and computational efficiency, an error margin of 30 seconds is allowed for a 95% confidence interval. Solving Equation 8 for n results in the following required sample size:

$$n = \left(\frac{z\sigma}{E}\right)^2 = \left(\frac{1.96 \cdot 305}{30}\right)^2 \approx 397 \tag{9}$$

As a result a sample size of $n = 400$ will be used to retrieve results from the simulation.

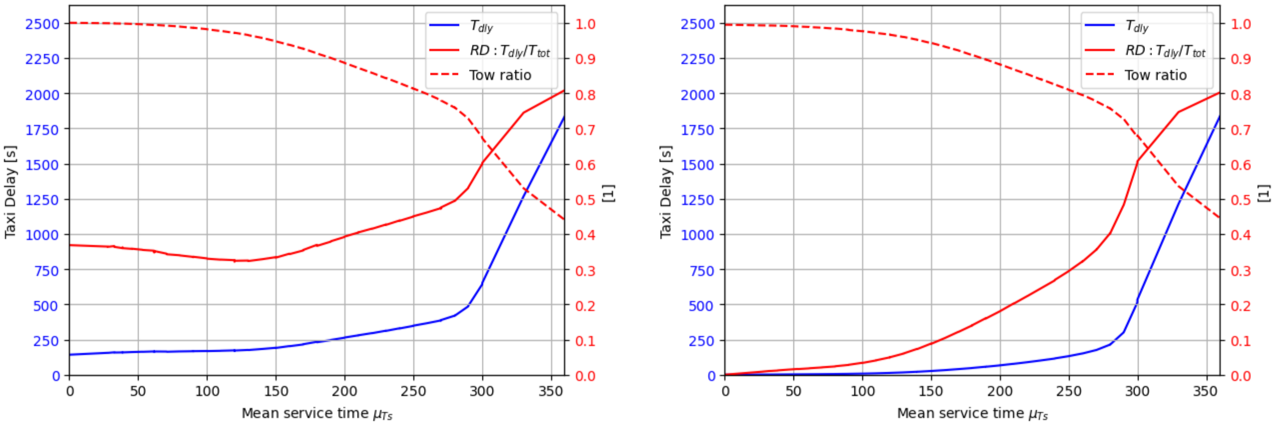
5 Results

The following chapter assess the performance of the operational towing system by discussing the output from the discrete event simulation. The performance of the system is measured using four different KPIs which have been defined in section 4.4.2.

5.1 Taxi-delay

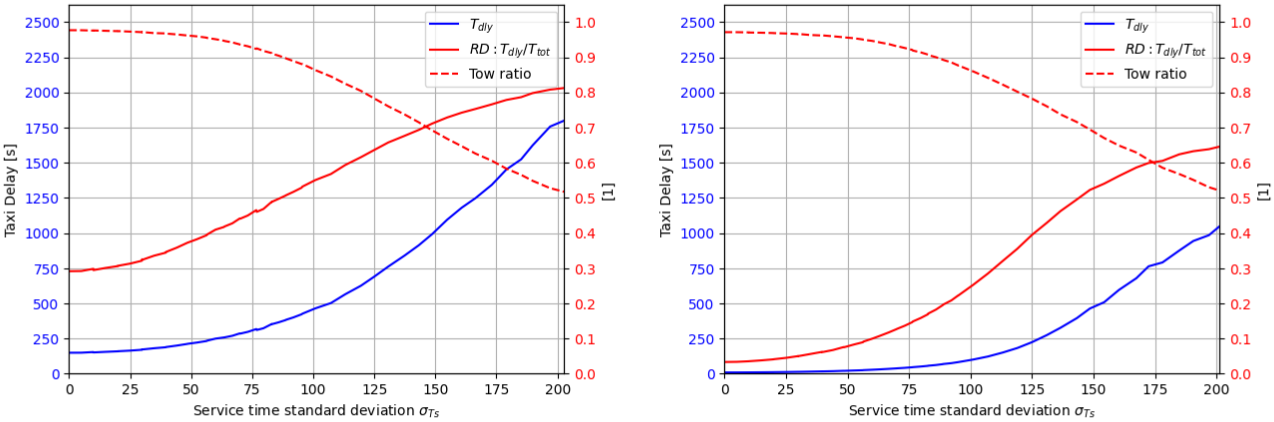
Figure 7 shows the resulting taxi-delay (absolute: T_{dly} and relative: RD) against the mean service time for decoupling departures (left) and coupling arrivals (right) in front of the runway. The input variable μ_{Ts} which represents the mean service time for coupling and decoupling in front of the runway, is simulated in a range from 10 seconds to 360 seconds (6 minutes) using discrete steps of 10 seconds. Every data point is the average result of $n = 400$ iterations done by the simulation for that specific input variable. The variability of the service time, expressed in standard deviation σ_{Ts} is kept constant at 30 seconds. The number of available tow-trucks is in all cases 12 and there are 3 parallel service-stations that can be used for the departures and arrivals individually. The same is done for for the variability of the service time. Figure 8 shows the resulting taxi-delay against a range of different values for the service time standard deviation σ_{Ts} while keeping the average service time constant at $\mu_{Ts} = 120s$.

For all situations the resulting towing-ratio (TR) which is the share of total flights that are towed is shown as well. Flights are only towed by a tow-truck when a truck is available when the aircraft arrives. It can be seen that the taxi-delay for departures aircraft does not approach zero when towing service times are 0 (no towing). This is expected because of the departure queue in front of the runway. As the taxi-delay T_{dly} was defined as



(a) Departures (b) Arrivals

Figure 7: Taxi-delay vs Mean service time, $\sigma_{Ts} = 30s$, $Lim_s = 3$, $\#T = 12$



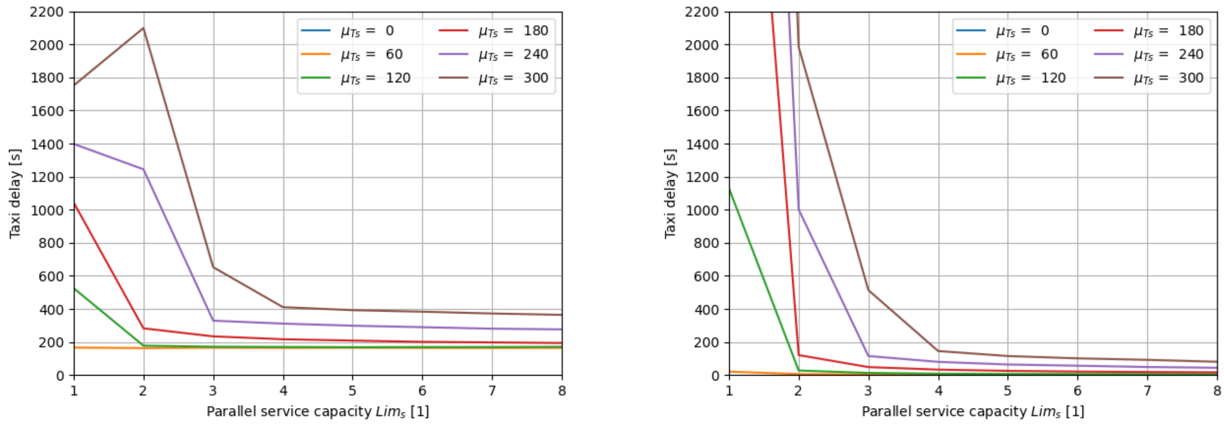
(a) Departures (b) Arrivals

Figure 8: Taxi-delay vs Service time standard deviation, $\mu_{Ts} = 120s$, $Lim_s = 3$, $\#T = 12$

being total idle time of the aircraft, it includes any other form of waiting time of the aircraft. Using 7(a) the effective delay of the conventional situation, when no flights are being towed, can be derived. It was found that this value is equal to $T^0_{dly} \approx 142s$.

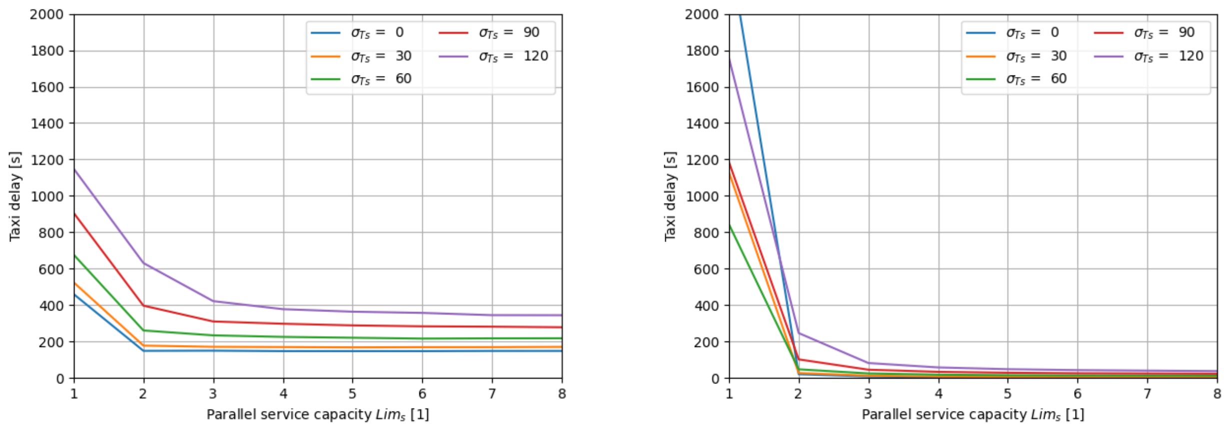
Next to distribution parameters of the service time for aircraft coupling and decoupling, the impact of the parallel service limit is investigated as well. Figure 9 and Figure 10 show the relation between taxi-delay and the parallel service limit. Multiple lines are shown for various values of μ_{Ts} and σ_{Ts} to show how the relations change for different service time distributions. It can be seen that the taxi-delay is acceptably bounded in most scenarios when allowing at least a parallel service limit $Lim_s \geq 3$. It can be noted however that especially arrivals suffer from very high taxi-delays if $Lim_s < 3$. 9(b) and 10(b). Arrivals cannot pass each other if another arrival in front of them is being coupled to a tug, while it is waiting the chances increases that the apron runway buffer grow and thus a tow-truck becomes available before the arrival is front in line. In that case this arrival will also be paired with a tug and decoupled. This means that the demand from arrivals for service in front of the runway is spread evenly. This is not the case for departures. Departures couple to their tug at the gate simultaneously. This means that the buffer empties relative quickly at once if enough tow-trucks. Because departures arrive at the runway either being towed or not, their waiting time does not influence their required service. They either need to be decoupled or not. Therefore the service demand for departures at the runway station is not evenly spread but comes in waves. It turns out that is more efficient then evenly spread demand as is the case for arrivals.

Another remarkable finding is the fact that taxi-delay increases while changing from one to two parallel service capacity. This is only the case for relative high average service times μ_{Ts} as can be seen in 9(a). A clear explanation for why this is happens is not yet found. The data in Table 5 also shows that when $Lim_s = 2$ the



(a) Departures (b) Arrivals

Figure 9: Taxi-delay vs Parallel service limit trends for various values for μ_{Ts}, $\sigma_{Ts} = 30$, $\#T = 12$



(a) Departures (b) Arrivals

Figure 10: Taxi-delay vs Parallel service limit trends for various values of σ_{Ts}, $\mu_{Ts} = 120$, $\#T = 12$

towing-ratio increases compared to when $Lim_s = 1$ which might indicate the increase in taxi-delay. However this increase in towing-ratio happens for all values of μ_{Ts}. Still, from μ_{Ts} values of 260s and higher the expected decrease in taxi-delay when increasing Lim_s from 1 to 2 changes in an increase in taxi-delay instead.

5.2 Runway-Throughput

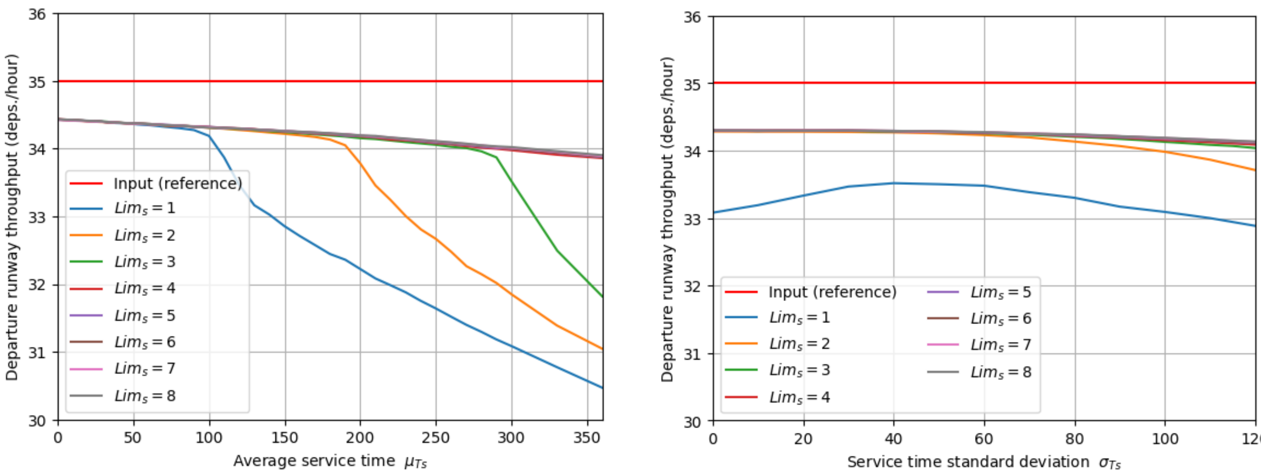
Another aspect of the operational efficiency is the realized runway throughput. The simulation processes 300 departures which enter the model at a rate 35 flights per hour. The resulting throughput of the entire system is equal to the average rate of departures leaving the system (take-off). In order to identify the throughput drop caused by the system, the average taxi separation between flights at the runway should match the input rate of the departures. This is is equal to $\frac{60}{35} \times 60 \approx 103$ seconds.
Figure 11 displays the resulting throughput rate in dep/hour from the departure runway as a function of the mean (left) and standard deviation (right) of the service time. It shows this behaviour for multiple service capacity limits. The results indicate that the throughput starts to drop rapidly when the average service time becomes more then 100 seconds per available service capacity, which is similar to the average interval time between departures at a rate of 35 per hour.

The impact of the standard deviation and thus the variability of decouple service times impact the runway throughput to a lesser extent. In the case that only one aircraft is served at a time, the departure throughput even increases for a short time as the standard deviation grows. However assuming parallel service limit of three aircraft at a time, standard deviations of up to 120 seconds seem to make no significant impact on runway

μ_{Ts} [s]	σ_{Ts} [s]	Lim_s [-]	$\#T$ [-]	T_{dly} [s]	TR [-]
240	30	1	12	1408,271	0,443
240	30	2	12	1244,192	0,583
250	30	1	12	1468,909	0,426
250	30	2	12	1387,437	0,547
260	30	1	12	1516,153	0,411
260	30	2	12	1554,065	0,514
270	30	1	12	1584,063	0,396
270	30	2	12	1700,541	0,486
280	30	1	12	1627,166	0,383
280	30	2	12	1838,550	0,463
290	30	1	12	1694,877	0,370
290	30	2	12	1981,873	0,441
300	30	1	12	1734,013	0,359
300	30	2	12	2098,372	0,423
360	30	1	12	2091,401	0,302
360	30	2	12	2746,207	0,347

Table 5: Towing-ratio & taxi-delay values for high average service time inputs

throughput.



(a) Average service time μ_{Ts}, $\sigma_{Ts} = 30$

(b) Service time standard deviation σ_{Ts}, $\mu_{Ts} = 120$

Figure 11: Resulting departure runway throughput vs service time distribution for various values of Lim_s, $\#T = 12$

Figure 11 highlights the fact that when average service times equal zero, meaning no operational towing is happening, the resulting departure runway throughput still does not reach the reference throughput value of 35 deps/hour. This can be explained by the variance of inter-arrival times between different departures that enter the system. Departures are entering the system at an average rate of 35 deps/hour but the inter-arrival times are still random samples from a probability distribution. On the other hand the departure-departure interval times are assumed fixed and thus are all the same, matching the 35 deps/hour. Given that the departure runway queue is empty at times, the resulting time between two departures leaving the system can be higher then the fixed minimum interval time. Because of this the average resulting runway throughput will be lower then the reference value.

5.3 Towing-ratio

The towing-ratio is the last measurement that was used to test the performance of the operational towing system. As the purpose of this research is analyze the adoption of operational towing, the level of adoption is import to analyze. In the results discussed above, the simulation does not enforce a minimal amount of flights to be towed instead the achieved towing-ratio is an output from the model. section 5.1 shows that the towing-rate stays above 80% as long as the average service time remains below 250 seconds, and starts dropping more quickly after this threshold is passed. The relation between the towing-ratio and the standard deviation of the service time is more linear. An 80% towing-ratio is maintained below values of 120s for σ_{Ts}. Figure 12 shows the resulting towing-ratio for different amounts of tow-trucks.

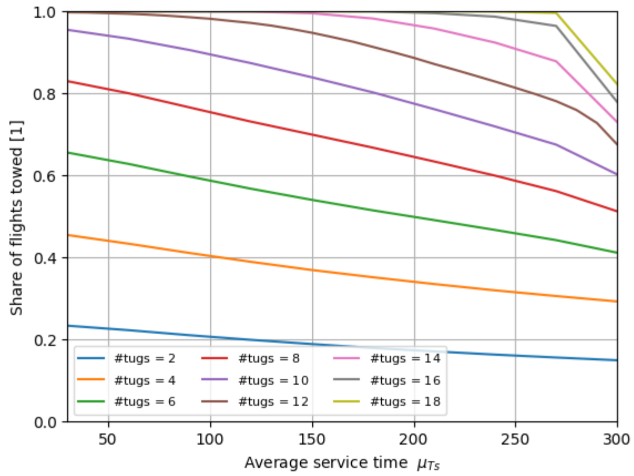


Figure 12: Resulting towing ratio vs average service time for different amount of tow-trucks used, $\sigma_{Ts} = 30s$, $Lim_s = 3$

6 Conclusion

This study aimed to provide insights into the operational efficiency of sustainable aircraft towing at hub airports through a detailed simulation model. The research explored the impact of operational towing on taxi delays and runway throughput, focusing on the coupling and decoupling times of tow trucks and their availability. The simulation results indicate that taxi delays for both departures and arrivals are significantly influenced by the mean and standard deviation of service times for coupling and decoupling tow-trucks. It was found that taxi delays increase with longer mean service times and higher variability. For instance, a mean service time beyond 250 seconds substantially increases taxi delays, while a standard deviation should not be higher then 50 seconds in order to maintain acceptable delays below 250 seconds.

Even though taxi-delay and runway throughput are typically correlated, taxi-delay is more important for on-time performance of individual flights, whereas runway throughput determines the capacity of an airport. The throughput seems to remain considerably stable, until the mean service time exceeds 100 seconds, upon which the throughput decreases sharply. 100 seconds aligns with the average interval time between departures that was used as input for the model. This indicates that the average service time should not be any higher than the average arrival rate of departures. Allowing for more serial service areas in front of the runway can help overcome this. However every increment of added capacity will be less effective as long as the aircraft cannot pass each other. This actually provides an expression for calculating the the minimum required parallel decouple spaces a runway needs in order to maintain its desired runway throughput RT_{des}. The desired runway throughput is assumed to be the runway throughput that is achieved without operational towing. The following relation between RT_{des} and Lim_s roughly holds:

$$Lim_s \geq \frac{3600}{\mu_{Ts}} \times \frac{1}{RT_{des}} \tag{10}$$

The expression holds especially for low values of μ_{Ts} : 0-300 seconds). High variability in service times (standard deviations up to 120 seconds) do not significantly affect throughput.

The objective to which this research is executed was to provide insight in the relations and behaviour of operational towing processes at hub airports by analyzing the resulting operational efficiency of the airport. In order to achieve this objective intermediate steps are defined that all should contribute to achieving this goal. Even though this research does not consider the environmental and economic impact from adopting operational towing, it can be assumed that for the best economical performance the taxi-delay must be limited and for optimal environmental impact the towing-ratio should be maximized. Using 10 tow-trucks for departures and arrivals combined, the resulting tow-ratio remains above 90%, when average service times are 120 seconds. The resulting taxi-delay will remain in this case below 250 seconds on average for departures and even almost 0 for arrivals. This indicates that efficient towing operations are feasible with well-managed service times and adequate tow truck availability.

Key operational challenges related to operational towing were identified in this research. It turns out that these challenges heavenly impact the resulting operational efficiency of an airport. The research has provided a simulation model that can be used to create valuable insight in some of the concerns of operational towing and their resulting behaviour. Even though the resulting performance of sustainable in this research is analyzed in absolute values, measuring taxi-delay and runway throughput, the value of the outcome of this research lies more in the fact that this research has identified and confirmed that some of the uncertainties that where determined in the literature study influence taxi-delay and runway throughput. The research provides ranges for certain parameters in which they must lie in order to support a feasible operational towing.

Effective implementation of operational towing requires addressing operational challenges, particularly in managing the coupling and decoupling from aircraft near the runway. Modifying airport infrastructure to support efficient tow truck operations is potentially needed as well. The economical and environmental aspects have not been treated in this research, however the results from this research can be used in future research to explore economic and environmental feasibility of operational towing on a larger scale. Additionally, exploring the integration of tow-truck charging and assessing the long-term benefits of reduced fuel consumption and emissions from these electrics tow-trucks provide a more comprehensive understanding of the viability and effectiveness of operational towing systems.

In conclusion, while sustainable aircraft towing presents a promising solution for reducing greenhouse gas emissions from airport operations, its successful implementation depends on addressing the operational complexities identified in this study. By optimizing service times, managing departure queues, and ensuring sufficient tow-truck availability, airports can achieve environmental sustainability while maintaining operational efficiency. This study has proposed a solution using object-oriented discrete event simulation, which has proven its usability in assessing the logistical effects of operational towing. The model can be used as solid basis in future research on this topic.

7 Recommendations

Although this research has provided valuable insights into the effective management and behavior of operational towing at hub airports, several areas require further exploration to obtain more accurate and reliable results. Given the many unknowns at this stage regarding the actual performance of tow trucks, and considering the initial aim of this research to generate a generic scenario for assessment, numerous assumptions were made to make the study feasible. Additionally, certain aspects of operational towing were not addressed.

Firstly, this research did not account for the charging of tow trucks and their battery levels. While the relationship between the number of available tow trucks, towing ratio, and delay was investigated, the assumption that tow trucks are electric implies they need to be charged, which reduces their availability compared to the scenario presented in this study. The buffers introduced in this research could potentially serve as charging stations for the tow trucks. Future research could explore this aspect further, analyzing the usability of tow trucks in greater depth. In addition to charging, other aspects of tow truck usability could be examined, such as the percentage of time a tow truck is actually towing an aircraft or the ratio between trips with and without an aircraft.

This research extensively investigated the characteristics and impact of serial aircraft coupling and decoupling near the runway. However, an alternative approach would involve parallel services, allowing aircraft in the service queue to pass others that have not yet completed their service. While parallel serving could positively impact taxi delays and runway throughput, it presents practical challenges, such as the need for dedicated service spots near the runway. Airports would need to allocate space for these spots. Future research could investigate whether investing in such parallel service spaces would be beneficial compared to the performance gains achieved.

The research aimed to analyze a typical day of operations by simulating a fixed number of flights in each iteration, assuming a constant arrival rate for flights (departures and arrivals) throughout the simulation. The chosen rate in this study matched the assumed maximum runway capacity. In reality, flight arrival rates fluctuate throughout the day, only reaching maximum capacity at specific times. The method used in this research, where all flights arrive at the same maximum rate over time, is useful for analyzing the system's steady-state behavior. However, achieving steady-state more quickly would require adding initial non-empty queues to the system, which was not done in this study. Another approach could involve simulating a larger number of flights and analyzing taxi times only for the last portion of these flights to obtain steady-state results. These are two recommendations for future research to better analyze the system's steady-state behavior. Conversely, if the goal is to analyze the performance over a single day, limiting the number of flights and starting with empty queues is more accurate. However, in this case, a dynamic arrival rate, with peak rates only at specific times, should be employed.

To achieve more reliable results, specific case studies could be conducted using the framework provided in this research. The framework could be adapted to reflect the physical representation of real airport environments and used with actual airport data. This would ultimately yield more practical results that could directly inform airport management decisions on implementing operational towing.

References

[Baaren, 2019] Baaren, E. V. M. V. (2019). The feasibility of a fully electric aircraft towing system master thesis.

[Baldwin et al., 2015] Baldwin, W. C., Sauser, B., and Cloutier, R. (2015). Simulation approaches for system of systems: Events-based versus agent based modeling. *Procedia Computer Science*, 44:363–372. 2015 Conference on Systems Engineering Research.

[DECDEA, EUROCONTROL, 2021] DECDEA, EUROCONTROL (2021). "comparison of operational air navigation system performance in brazil and europes". Part B - Initial Comparison Report Brazil/Europe.

[EUROCONTROL, 2023] EUROCONTROL (2023). "eurocontrol data snapshot 40 on taxi times". Data Snapshot 40.

[Federal Aviation Administration, 2014] Federal Aviation Administration (2014). "advisor circular - aircraft wake turbulence". AC No: 90-23G.

[Guimarans et al., 2017] Guimarans, D., Antonio, A., Juan, A., Fonseca i Casas, P., and Calvet, L. (2017). Using simulation to estimate critical paths and survival functions in aircraft turnaround processes.

[Guo et al., 2014] Guo, R., Zhang, Y., and Wang, Q. (2014). Comparison of emerging ground propulsion systems for electrified aircraft taxi operations. *Transportation Research Part C 44(2014):98-109*.

[Lukic et al., 2018] Lukic, M., Hebala, A., Giangrande, P., Klumpner, C., Nuzzo, S., Chen, G., Gerada, C., Eastwick, C., and Galea, M. (2018). State of the art of electric taxiing systems; state of the art of electric taxiing systems. *2018 IEEE International Conference on Electrical Systems for Aircraft, Railway, Ship Propulsion and Road Vehicles & International Transportation Electrification Conference (ESARS-ITEC)*.

[Lukic et al., 2019] Lukic, M., Member, S., Giangrande, P., Member, S., Hebala, A., Nuzzo, S., and Galea, M. (2019). Review, challenges, and future developments of electric taxiing systems. *IEEE TRANSACTIONS ON TRANSPORTATION ELECTRIFICATION*, 5.

[Re, 2012] Re, F. (2012). Assessing environmental benefits of electric aircraft taxiing through object-oriented simulation. *SAE International Journal of Aerospace*, 5:503–512.

[Salihu et al., 2021] Salihu, A. L., Lloyd, S. M., and Akgunduz, A. (2021). Electrification of airport taxiway operations: A simulation framework for analyzing congestion and cost. *Transportation Research Part D: Transport and Environment*, 97:102962.

[Soltani et al., 2020] Soltani, M., Ahmadi, S., Akgunduz, A., and Bhuiyan, N. (2020). An eco-friendly aircraft taxiing approach with collision and conflict avoidance. *Transportation Research Part C: Emerging Technologies*, 121:102872.

[TaxiBot, 2023] TaxiBot (2023). TaxiBot - Concept.

[Yang et al., 2017] Yang, L., Yin, S., Han, K., Haddad, J., and Hu, M. (2017). Fundamental diagrams of airport surface traffic: Models and applications. *Transportation Research Part B Methodological*, 106.

II

Literature Study

previously graded under AE4020

Literature Study

Implementing external aircraft taxiing with minimal flight delay for sustainable aviation

Yvor van den Beuken

Technische Universiteit Delft

TU Delft

Literature Study

Implementing external aircraft taxiing with minimal flight delay for sustainable aviation

by

Yvor van den Beuken

in partial fulfilment of the requirements for the degree of

Master of Science
in Aerospace Engineering

at the Delft University of Technology,
Control and Operations - Air Transport Operations

Student Number:	4585887
Date of publishment:	January 2, 2023
Thesis supervisor:	ir. Paul Roling

Delft
University of
Technology

Contents

Nomenclature **iii**

1 Introduction **1**
1.1 Reducing emissions from taxiing . . . 1
1.2 Report Structure . . . 2

2 Aircraft taxiing **3**
2.1 Conventional aircraft taxiing . . . 3
2.1.1 Landing and take-off cycle . . . 3
2.1.2 Separation requirements . . . 3
2.2 Operational aircraft towing . . . 5

3 Eco-friendly taxiing systems **6**
3.1 Emerging eco-friendly taxiing solutions . . . 6
3.2 External aircraft ground propulsion systems . . . 8
3.2.1 Operational impact of dispatch towing. . . . 8
3.2.2 Adjusted traffic management for dispatch towing . . . 9

4 Simulation modelling **10**
4.1 Queueing theory . . . 10
4.2 Discrete event simulation . . . 11

5 Problem statement and research questions **12**
5.1 Research objective . . . 12
5.2 Research questions . . . 13
5.2.1 Sub questions . . . 13

Bibliography **14**

Nomenclature

Abbreviations

AGPS aircraft ground propulsion systems
APU auxiliary power unit
ATC air traffic control
DES discrete event simulation
EASA European Union Aviation Safety Agency
EGTS Electric Green Taxiing System
FAA Federal Aviation Administration
ICAO International Civil Aviation Organisation
LTO landing and take-off cycle
LUAW line up and wait
LVNL Luchtverkeersleiding Nederland
MILP mixed integer linear problem
MTOW maximum take-off weight
NLG nose landing gear
TAT turnaround time

1

Introduction

Over the last decades, greenhouse gas emissions have been growing exponentially. This subject has become a more and more prominent topic in politics because of the negative effects these emissions have on climate change due to global warming. When the emission of these greenhouse gasses, or at least its constant growth rate, is not limited, the negative effects of global warming will be irreversible. One of the measures that substantiates this concern is the United Nations Paris Agreement, in which the signing parties declare to take global action to limit global temperature rise by 2030. To achieve this goal, all companies and governments must adapt their way of operations and policies in order to limit their greenhouse gas emissions and eventually even become climate neutral.

Currently, the transport industry accounts for roughly 14% of the global greenhouse gas emissions [24]. The transport industry is mostly powered by fossil fuels. The relative high energy density of fossil fuels make it the ideal energy source for transportable energy. But the downside of these fossil fuels are the pollutants that come from it when burned. Various transport modes such as the car industry have been introducing alternatives to fossil fuels in order to make transportation less pollutant. However flying has always been a mode of transport with a very high fuel burn rate per passenger. Unlike the car industry, current technology is not promising short-term breakthroughs that making a shift to sustainable energy sources for aircraft. Newer, more efficient aircraft, such as the narrow-body Airbus A320neo and the Boeing 737 Max or the wide-body Airbus A350 and the Boeing 777X have been introduced in the last decade that are all claimed to be 10-20% more fuel efficient then their predecessors, due to the introduction of newer engines and lighter fuselage materials[1, 3, 2, 20]. Still, these improvements are not enough when considering the vast growth of the aviation sector [14]. These efficiency improvements of newer aircraft and engines are simply outpaced by the growth rate of the industry itself and thus more solutions in order to reduce fuel burn in aviation are required.
Next to the environmental interest, using less fuel will also bring an economical improvement for airlines. Currently, airlines spend roughly 32% of their operating budget on fuel, which is the second largest expense after labor [12]. Since airlines operate with a very small profit margin, they remain concerned about the volatility of fuel prizes because of the direct impact they have on the airlines' profitability. Therefore, becoming less dependent on fuel is desirable both from an environmental and economical perspective.

1.1. Reducing emissions from taxiing

Aircraft use significant amounts of fuel during their landing and take-off cycle (LTO). A popular narrow-body aircraft such as the Airbus A320 uses 5 to 10% of its fuel during its LTO [5, 17] which includes landing, take-off and taxiing. With conventional taxiing, an aircraft uses its own engines to move forward. Different solutions have been proposed to minimize or make aircraft ground operations more fuel-efficient. These types of sustainable novel solutions in order to replace the conventional taxiing propulsion, are collectively called eco-friendly taxiing solutions. Among these novel solutions are internal aircraft ground propulsion systems (AGPS), which incorporates an electronic driving system into the gears of the aircraft to drive it. Proposed systems are EGTS and Wheeltug [25]. Other studies investigate using only a single engine for taxiing (single-engine taxiing). A third way to limit taxiing emissions from aircraft is having them tugged by an external electric tow-truck. Which can be called dispatch towing. The challenge here is mainly the implementation of a full functioning system in the current airport infrastructure as dispatch towing introduces new operational challenges for airports. External tow-trucks lead to larger amounts of traffic in often already very busy airport environments. If not managed properly, this can lead to taxiway congestion resulting in flight delays. This study aims to reveal the challenges and bottlenecks of operating an aircraft towing system with respect to taxiway congestion. By doing so it hopes to find both design requirements for tow-truck design as well as strategic recommendations for future implementations in an airports' infrastructure .

1.2. Report Structure

This literature study report is build with the main purpose of defining the problem setup and research direction of the corresponding master thesis by reviewing literature on the topic. For this to be achieved, the report consists of a five chapters. chapter 2 gives insight into the process of conventional taxiing as it is now as well as an introduction into the proposed method for dispatch towing. Both processes are compared. chapter 3 reviews the existing researches on eco-friendly taxiing solutions. Next, in chapter 4 discrete event simulation and queuing theory are introduced as the methods used for developing a taxi operations simulation. The report ends with the research plan in chapter 5. Here, the research questions as well as the proposed method are described in a concluding fashion.

2
Aircraft taxiing

In order to study the management of aircraft taxiing, a basic understanding of the taxi process and its characteristics is needed. In this chapter the conventional taxi process is outlined including aspects such as standard procedures and regulations that are important for managing the aircraft taxi movement. The second part of the chapter is devoted to describing a basic design of the new taxiing process of dispatch towing and the different challenges that come with it compared to conventional taxing.

2.1. Conventional aircraft taxiing

The International Civil Aviation Organisation (ICAO) refers to the taxi phase of an aircraft as: ”*The phase of flight in which movement of an aircraft over the surface of an aerodrome under its own power occurs, excluding take-off and landing*”[13]. Taxiing happens thus only between the ramp area which lies in the airports non-movement area and the runways which are a part of the airports movement area. During taxiing aircraft may use the taxiways which are also part of the movement area. This means that during a standard LTO cycle, the aircraft has two taxi phases. The first towards the gate and the second from the gate back to the departure runway.

2.1.1. Landing and take-off cycle

A standard LTO cycle looks as follows: After landing, the aircraft leaves the runway through an assigned runway exit. It follows its journey over the taxiways under its own power towards its assigned gate where it comes to a complete stop, this moment is called the ”on-block time”. Now the crew and staff will start the turnaround process which includes (dis)embarking of passengers, cargo loading, cleaning the aircraft, refueling and aircraft inspections. When these activities are all finished the plane is ready to depart again. A push-back truck connects to the nose-wheel gear of the aircraft and starts pushing it backwards from the gate. This instance is referred to as the ”off-block time”. The total time between the on-block time and the off-block time is called the turnaround time (TAT). The TAT for an aircraft typically depends on the type of aircraft, number of passengers and the quantity of cargo. It ranges from 30 minutes for narrow body aircraft like the Boeing 737 to 90 minutes for wide body aircraft like the Airbus A380 [11, 23]. While the aircraft is pushed back the pilots typically start the engines for warm-up which takes about 3-5 minutes [22]. When the aircraft is pushed back it can continue taxiing under its own power to its destination runway through an assigned route by air traffic control (ATC). Figure 2.1 shows the complete sequence schematically.

2.1.2. Separation requirements

In order to guarantee safety, aircraft should maintain a minimal distance from one another both in the air and on the ground. ATC carries out these rules in the airports movement area. Two types of separation standards are of interest for managing the aircraft during their LTO. Taxi separation handles the horizontal separation distance between two aircraft taxiing, whereas the airborne wake-vortex separation handles the separation of two consecutive aircraft that are either about to land, take off or a combination and are using the same runway. Taxi separations guarantee that in any conditions

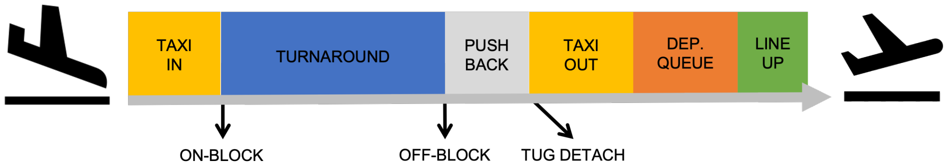


Figure 2.1: Conventional turnaround-process sequence

(e.g. wet runways) aircraft are able to make a safe sudden stop. For taxi separation, no clear definition by either the Federal Aviation Administration (FAA), European Union Aviation Safety Agency (EASA) or the Luchtverkeersleiding Nederland (LVNL) are defined however. The Australian Civil Aviation Order 20.9 indicates that a minimum distance of 15 meter separation should be held for turbo-prop aircraft and 30 meters for turbo-jet engines assuming the aircraft travel at, or below standard taxiing speeds.[22]. Airborne wake vortex separation is there to make sure that trailing aircraft do not get disturbed by the wake vortex turbulence of their predecessor during landing or take-off. These separations are defined for both landing as well as departing aircraft. For various regulation purposes, the FAA has defined six different classes which group aircraft according to their maximum take-off weight (MTOW) and wingspan.[7]

Cat	Wake Turbulence Class	MTOW (lb)	Wingspan (ft)
A	Super Heavy	$>$ 300.000	$>$ 245
B	Upper Heavy	$>$ 300.000	$>$ 175 & $\leq$ 245
C	Lower Heavy	$>$ 300.000	$>$ 125 & $\leq$ 175
D	Upper Medium	$\geq$ 41.000 & $<$ 300.000	$>$ 90 & $\leq$ 175
E	Lower Medium	$\geq$ 41.000 & $<$ 300.000	$>$ 65 & $\leq$ 90
F	Light	$<$ 41.000	wingspan $<$ 125 feet

Table 2.1: FAA wake turbulance categories.[7]

These categories are also used when defining separation standards between the possible pairings representing a leading and trailing aircraft pair of either departing or landing aircraft. This can thus either be a departure-departure, departure-arrival, arrival-departure or arrival-arrival combination. For clarity reasons, when talking about an arrival-departure combination the arrival is assumed to be preceding the departure.

Departure-Departure
Separation between departure-departure pairs has been defined for aircraft that are both planning on using either the same runway or two parallel runways within 2500 feet of each other.[8]. For departing aircraft there is no need to define a distinct interval time for all possible combinations of the six aforementioned categories. In practice, only three categories are used for consecutive departures on the same runway.[8, 21] These FAA separation standards are defined as interval times between the two aircraft and can be found in Table 2.2.

Arrival-Departure
For Arrival-Departure combinations, no wake vortex separation is needed. In order to mitigate collisions, the preceding landing aircraft must be clear of the runway before the departing is allowed to start their take-off roll.[9]

Arrival/Departure-Arrival
Incoming arriving aircraft that are preceded either by a departure or another arrival should maintain a minimal separation distance between one and another depending on their respective aircraft weight class. At the instance that the leading aircraft crosses the runway threshold, the trailing aircraft is not allowed to be closer then the stated separation distances.

Leading aircraft cat.	Trailing Aircraft Cat.		
	A,B,C	**D,E**	**F**
A,B,C	90	120	120
D,E	60	60	60
F	45	45	45

(a) Departure-Departure separation in s

Leading aircraft cat.	Trailing Aircraft Cat.		
	A,B,C	**D,E**	**F**
A,B,C	4	5	6
D,E	2.5	2.5	4
F	2.5	2.5	2.5

(b) Arrival-Arrival/Departure separation in nm

Table 2.2: FAA separation intervals for arrivals and departures on the same runway.[8, 21]

2.2. Operational aircraft towing

As described in the introduction of this chapter, operational towing refers to a system in which aircraft are towed towards and from the runway and gates instead of using their own engines to travel. The major differences are the additional waiting times for aircraft in order to attach and detach the tugs and the increased traffic due to tugs now travelling in the airports' movement area. The sequence of the conventional LTO cycle from Figure 2.1 must be adapted in order to represent the new system. The visualization of the new LTO cycle sequence is displayed in Figure 2.2.

After landing, arriving aircraft now wait at the first taxiway intersection to be picked up by an available tow-truck. The tow-truck attaches to the aircraft and starts towing it towards the gate. When the tugged aircraft arrives at the gate, detachment of the tug and the turn around procedure can both start. After turnaround has passed and push-back clearance is given again an available tug will attach to the aircraft and push it back from the gate and immediately continue with towing the aircraft towards its assigned runway. When arriving at the runway departure queue, the tug will be detached. The aircraft must now wait for line up and wait (LUAW) clearance. Challenges will mainly arise in the last part of the sequence during the departure queue. Depending on the length of the queue and the time to detach, decisions must be made in order to minimize delay. Assuming aircraft cannot pass each other, leading aircraft can hold up other aircraft when they are being detached from their tug.

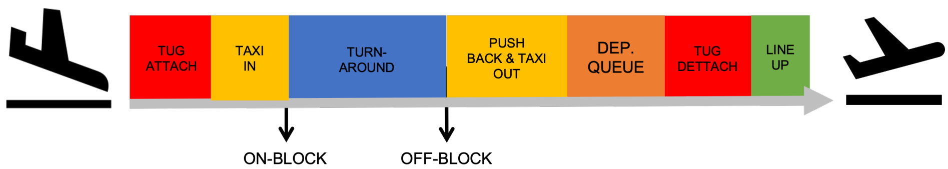


Figure 2.2: Turnaround-process sequence with dispatch towing

3

Eco-friendly taxiing systems

The following chapter investigates the research and development into novel eco-friendly taxiing solutions and shall provide insight into the overall performance potential of various AGPS (introduced in chapter 1) and the related challenges that remain. The chapter will start with an overview of the three emerging types of solutions and follow up with a more detailed analysis of one of the three types, namely external AGPS since these systems are the subject of interest for this study.

3.1. Emerging eco-friendly taxiing solutions

Conventional taxiing requires that aircraft use their main engines at relative low speeds and power modes, for which the engines are not optimized. Also, during this phase landing gear brakes are used a in order to control the speed of the aircraft which results in wear and a significant amount of energy waste.[12] Conventional taxiing therefore, is a rather in-efficient process. More eco-friendly alternatives and strategies for taxiing could achieve lower fuel emissions at airports and in aviation in general. AGPS covers all emerging eco-friendly taxiing solutions for aircraft. These can generally be classified into three different types: single engine taxiing, on-board AGPS and external AGPS systems. Various types of these solutions exist that may or may not include an electric motor. According to Re [19], AGPS have three functional requirements:

- Perform gate pushback
- Moving the aircraft from standstill with a sufficient acceleration
- Driving the aircraft along its taxi route

The remainder of this section briefly introduces the emerging eco-friendly taxiing strategies found in the literature in recent years and finishes with a comparison between the pros and cons of each strategy.

Single-engine taxiing

Single-engine taxiing is operationally the most attractive as it is relative easy to implement. With this system only half of the aircraft's main engines are used for taxiing. The engines that are used, deliver more thrust compared to conventional taxiing. Various studies have shown the potential fuel savings and emissions reduction following from single-engine taxiing while also improving engine life economy. [12]

One of the disadvantages of this solution however, is the increase in taxi-time. Aircraft have slower taxi speeds when powered by only half their engines due to lower acceleration and increased difficulty when taking sharp turns or taxiing in bad weather conditions. Additionally, aircraft need extra time before take-off to warm up the other unused engines which leads to longer departure queues. Especially at busy airports this can decrease on-time flight performance.

Internal AGPS

Internal AGPS requires an electric motor in the aircrafts' nose or main gear that powers the aircraft. This method thus eliminates the use of the aircraft's main engines during taxiing completely. Electricity

that drives the motor is provided through the auxiliary power unit (APU) of the aircraft.These kind of systems have a quite high fuel saving potential however this method still is not emission free as the APU still uses fuel.[12] The APU from the aircraft in combination with the electric motor is more fuel efficient then the aircrafts' main engines for driving the aircraft at low speeds. The required technological development in order to implement such a system is mostly the design of a capable internal electric motor that fits the landing gear of an aircraft. Various initiatives have developed an internal AGPS such as Wheeltug and Electric Green Taxiing System (EGTS). Wheeltug is a fully integrated AGPS system integrated in the nose landing gear (NLG) of aircraft. The NLG was chosen because of the absence of the brakes, which leaves more room for the electric motor. It has been tested on medium sized aircraft such as the boeing 767 and the boeing 737-800 more recently.[12]. A picture of Wheeltug can be found in 3.1a. EGTS on the other hand is an internal AGPS motor developed for integration in the aircraft's main gears.
However, these systems also have some challenges. Thermal behaviour of the electric motor could damage the breaks and other parts of the landing gear. This is especially important for the EGTS solution. Next to this, the main concern for these types of engine-less taxiing, is the extra weight the electric motor adds to the aircraft. This extra weight is carried during the entire flight and thus are savings from engine-less taxiing partially nullified by the extra weight of the aircraft. Most studies have only considered savings from taxiing only. Some studies have considered the actual trade-off between savings from taxiing and the increase fuel consumption due to the extra weight. Re 2012[18], found an overall potential fuel saving of 2.6% for mid-sized aircraft.

External AGPS
Just as with internal AGPS, external systems eliminate the need of the aircraft's main engines for taxiing. A separate truck is attached to the aircraft that carries the aircraft from gate to runway and back. This includes push-back as well. The trucks that are used for external AGPS are different from standard push-back trucks used only for gate push back. This means that new trucks should be developed for this purpose. TaxiBot, developed by Israeli Aerospace Industries is one of these developments and has been tested in practice already at various airports (3.1b)[12, 15]. The external AGPS taxiing procedure is also called dispatch towing [19, 12]. The advantage of such systems is that the choice fuel source is much more flexible when compared to the other methods. Electric trucks can be powered by renewable energy and this allows for zero emission taxiing.
However, the main concern of this method is the increased complexity of handling aircraft during taxi. Especially before take-off, in the departure queue. As already mentioned in chapter 2, time is needed to detach the aircraft from the tug. For optimal fuel savings, this should happen as close to the runway as possible. If multiple aircraft in the departure queue need to detach their tugs, it can lead to difficulties in the departure flow. Not only will taxi time increase per aircraft, also unattached tow trucks now travel at the airport, making the taxiways at already busy airports even busier. When considering a minimal separation between travelling vehicles for safety guarantees, this can lead to even more congestion and as product larger delays.

(a) Wheeltug system implementend in NLG, *source: Wheeltug*

(b) KLM Boeing 737 being towed by TaxiBot, *source: KLM*

Previous literature has discussed and compared both types [12, 15, 16]. Both with unique advantages

and disadvantages over the other. The larger part of studies focuses on onboard systems because these systems are more easily adoptable on the short term and still have very promising fuel saving potential. [12] investigates the total emissions from taxiing for three different scenarios and compares this to the conventional method. The study includes both on-board as well as external AGPS as a scenario. The best performance improvement in general fuel consumption and emissions was found for an onboard AGPS powered by the aircraft's APU. This as opposed to external towing of aircraft by a diesel powered truck and single-engine taxiing. However, the extra mass of the on-board AGPS was neglected since only taxi operations where considered here. In reality the extra weight will counteract the fuel efficiency of the entire flight. [15] Argues that for narrow-body aircraft, which have many flight cycles with relative short routes, an on-board AGPS would be the better choice while for wide-body an external tow-truck is more beneficial. Wide-body have a high flight-time/taxi-time ratio. A possible scenario would therefore be a mixture of the two.
Concluding from these findings, the three eco-friendly taxiing techniques are graded relative to each other based on difficulty and fuel saving potential. The results can be found in Table **??**.

Eco-friendly Taxi strategy	Fuel saving potential	Difficulty of implementation
Single-engine taxiing	Small	Easy
Onboard AGPS	Medium	Medium
External AGPS	High	Hard

Table 3.1: Relative performance on various emerging eco-friendly taxiing strategies

3.2. External aircraft ground propulsion systems

The previous section concludes that external AGPS provide a good potential for zero-emmission taxiing. This makes this strategy attractive for further research such that eventually the strategy can be adopted by the aviation industry. While the fuel saving potential is high, implementation is still challenging due to the discussed issues before. The research aims to investigate the resulting operational impact when adopting external taxiing (dispatch towing), and how to handle it thereafter. In this section firstly other literature is assessed. When assessing, two important question will be answered: "What is the operational impact of dispatch towing?" and "How does the study manage traffic in order to limit the operational impact of dispatch towing?". With the term "*operational impact of dispatch towing*", the resulting delay of dispatch towing due to taxiway congestion is meant. Traffic management regards the strategy used in order to minimize or even mitigate the resulting taxiway congestion effects. The results will be concluded at the end.

3.2.1. Operational impact of dispatch towing

In order to investigate the resulting operational impact from dispatch towing, a representative model that can realistically simulate the process of operational towing is needed. This means that studies regarded in this section should include a modeling approach. So far, limited studies have researched dispatch towing by actually modelling a realistic situation. Soltani Et al.[26] proposes a system with a hybrid towing solution. In this system, part of the taxi operations are handled by a tow-truck using renewable energy while the other part still does regular taxiing using their own engines. Using a mixed integer linear problem (MILP), the study seeks to minimize fuel cost and delay by selecting the optimal choice of route and towing strategy for a given aircraft in a given schedule. Results include decision variables include amongst multiple the assigned taxi route per aircraft and whether or not the aircraft is towed or not. These decision variables represent the optimal traffic management strategies. Depending on penalties for fuel cost and delay cost, the model can prioritize taxi delay over fuel saving or vice versa. The study uses a network model reflecting the layout of Montreal's Pierre Elliot Trudeau International Airport (YUL) which is a medium sized hub airport containing three runways. When a hard constraint was given that all aircraft must be towed by a tow-truck, the delay seemed to increase rapidly when limiting the amount of available tow-trucks in the system. When more and more tow-trucks where introduced in the system, delay decreased in the best scenario to only three minutes. From this it can be concluded that the a first cause of delay due to dispatch towing would be the unavailability of tow-trucks. But Soltani et al. also mentions that the approach is a deterministic one and disregards various important factors such as decoupling and coupling times of the tow-trucks.[26] These factors can influence the performance of the system delay just as much and thus should be investigated further. When retrieving a more economically focused conclusion, Soltani et al found that 205 aircraft in one day ideally are handled by 12 tow-trucks in a hybrid towing scenario. With this configuration the lowest total operating cost is achieved. Here, operating cost included extra costs for delay, fuel cost and total operating costs of tow-trucks. Meaning that a trade-off has been made to sacrifice some extra delay such that lesser tow-trucks need to be operated.

van Baaren [4], studied the feasibility and performance of a fully electric dispatch towing system by making a proof of concept. This is done in two case studies, one at Rotterdam-The Hague airport and one at Amsterdam Schiphol Airport. Similarly to Soltani et al, van Baaren uses a MILP based on a network of nodes and links to realize the most optimal vehicle routes and planning. Different however is the fact that van Baaren uses a detailed approach of aircraft en tow-truck kinematics in order to determine and substantiate the taxi performance metrics such as speeds and acceleration. These metrics are also used to determine the energy and fuel consumption of the vehicles. This provides the possibility of considering charging of the electric tow trucks and a more accurate estimate of fuel consumption. Van Baaren seeks to find an optimal solution that states which aircraft from the schedule are towed and which are not, such to achieve minimal fuel consumption. Van Baaren assumes an ideal system in which all vehicles travel according to their schedules. It disregards traffic interaction at busy taxiways and therefore delay and taxi congestion in both the scheduling solution as well as the routing problem are not considered here.[4]

3.2.2. Adjusted traffic management for dispatch towing

From various studies it was found that dispatch towing brings new challenges for a realistic implementation. Van Baaren [4] covers the availability of tow trucks in a detailed manner by regarding the battery capacity of electric trucks and the fact that they need to be charged. These results can be very usefull for the technical design of these trucks. [26] covers with a case study conflict and collision avoidance between aircraft and tow-trucks. Both models propose a MILP and therefore it should be noted that these results opt the best possible strategy for handling dispatch towing in the exact scenario's that have been established in the studies, respectively. For both studies the output is an ideal number of tow-trucks for the given scenario. However both studies have not regarded the micro management during coupling and decoupling of aircraft that are being towed. This is especially important when service times for various tow-truck actions are not deterministic and thus can vary. Plannings wise this can be a real challenge. Such cases have not been discusses explicitly in the current literature. These cases are important in improving modeling efforts because they allow, amongst other things, for more accurate departure queue modelling. Traffic management strategies regarding the departure queue then can follow from it, providing a more complete picture for implementation of dispatch towing.

4

Simulation modelling

In chapter 3 it was found that previous literature that include a realistic modelling simulation of dispatch towing is limited. Soltani et al. [26] assessed dispatch towing mainly from an economical perspective realizing a strategy for minimizing operational costs by reducing fuel consumption and emissions. Van Baaren[4] focused mostly on the technical feasibility of the system focusing on the performance of tow-trucks and providing an optimal strategy for maximal fuel savings.

The goal of this study is to contribute a new modelling study within the subject of electric aircraft taxiing that gives insight in a third aspect of the feasibility of dispatch towing, namely the operational feasibility rather then economical or technical feasibility. For operational feasibility the emphasis lies on taxiway congestion and delay. Lukic M[15] states that introduction of dispatch towing will result in an increased level of taxiway congestion and therefore suggests adaptations to existing airport infrastructure should be made to overcome this. In order to retrieve well substantiated recommendations a more detailed simulation method is necessary that allows for stochastic service times. Queuing analysis can be used for this. Using a discrete even simulation, development of queues at the airport can be monitored live. Monte carlo simulations can be used to analyze the steady state behaviour of the traffic flow of all vehicles and give insight in performance indicators such as average taxi-time or delay. With this modelling technique, multiple different traffic management strategies can be tested for returning the best results. From these results the substantiated recommendations to the airport infrastructure can be deduced. The following sections treat the subject of queuing theory and discrete event simulation in order to get a better understanding of the method and how this can be used to create the desired simulation for this research by elaborating firstly on queuing theory and later on discrete event simulation.

4.1. Queueing theory

Queueing theory is often used for mathematical modelling of real life processes. The general process is formed by a queue in which customers arrive. Usually arrival is at random times following a probability distribution. The queue acts as a buffer in which customers wait to be served by the server. Every customer is handled one by one by the server when its their turn in the queue. The queueing theory is used to analyze the behaviour of the queue over time. Various characteristics can be retrieved, such as average waiting time per customer. This is valuable information for the design of real-life queueing related processes. Traffic flow in a road network is one of these real-life processes that is often described as a queueing network.

A queueing process is mathematically described using six basic characteristics that should be defined in order to adequately analyze the behaviour of the queue: (1) Arrival pattern of the customers involving an arrival rate λ, (2)Service pattern of the customers involving a service rate μ, (3)Queue discipline, regarding the order the customers are served, (4)System capacity, (5)Number of service channels, (6)Number of service stages.[10] Using these characteristics, queueing networks of multiple queues and service channels can be created. The key functionality of queueing theory is thus to model the behaviour of stochastic processes in the cases that arrival pattern and or service pattern are often non-deterministic.

Every stage of a queuing network can be analyzed individually. Such a stage exists of a queue and one or more servers. Behaviour analysis of these stages is pretty straightforward and probability distributions of resulting performance indicators (for instance the average taxi-time for an aircraft) can be derived analytically. However, when considering a more complex network of of multiple serial and parallel queues and service stages, the behaviour of the system becomes more advanced and harder to describe analytically. The representation of an airports taxiway network is an example of such a complex system. The behaviour of various vehicles and queues within the system is hard to compute analytically now. However by including the right logic, discrete event simulation can allow in this case for accurate step by step modelling of all actions the network undergoes. In this way, the state of the queues and vehicles in the network over time can be analyzed. Running a discrete event simulation for a large number of times will provide probability distributions of the relevant performance indicators. This will provide insight in an empirical way; without the need of deriving results from the complex underlying individual probability distributions from the different parts of the system. The system basically is a black box in which traffic flows in at a given rate and will flow out at another rate resulting from the logic of the simulation.

4.2. Discrete event simulation

A simulation is the computation of a relation between the state of a system and time. Every simulation has a time horizon in which the state of the problem is defined by a state vector. For continues simulations, the state-vector $\vec{S_t}$ containing all state values, is a continues function of time. This means that the state vector is known at every time point t in the time horizon. Discrete event simulation however, does not consider the entire time horizon but only discrete time points within it. More specifically, discrete event simulation only regards the state of the system at time points where the state of the system changes. The change of the state vector is called an 'event'. Discrete event simulation is therefore a well defined sequence of events at certain points in time. This requires relatively low computational effort. At every state $\vec{S_\tau}$, the next event is determined including the time at which this event occurs. The time in between events is disregarded and from a computational perspective unknown. The underlying assumption here is that the state of the system in between events is of no interest to the observer, which is often the case. At every moment in time, regarding the state at that point and possibly other previous states, the following event can be determined, which leads to simulation to the next state. Discrete event simulation is an attractive method when the analytical derivation of state variables is hard to derive. Since the method is far less computationally intensive then other forms of simulation, monte carlo simulation can often be used to derive the desired probability distributions of performance indicators. Monte carlo simulation makes use of the law of large numbers. This law states that if a simulation is ran enough times, the resulting empirical distribution of state variables from that simulation accurately represent the real underlying probability distribution. discrete event simulation (DES) therefore seems to be a good method for analyzing complex queueing networks and should allow for retrieving the required results for this study.

5 Problem statement and research questions

In the following chapter the problem and research questions of the thesis are described. Resulting from the background given in chapter 1 and the discussed literature through the rest of this paper, a research problem can be stated. From the problem a set of objectives can be described. The chapter will finish with the statement of the research questions.

5.1. Research objective

As described in the introduction of this paper, the aviation sector is one of the fastest growing sources of greenhouse gas emissions, and its impact on climate change is of increasing concern. The sector is responsible for 3.8% of the total CO_2 and creates roughly 13.9% of all emissions from transport, making it the second biggest contributor after road transport [6]. It was concluded that more new technological developments that help limit the emissions of greenhouse gasses in the air transport industry are very much needed. One of these developments include eco-friendly taxiing solutions such to limit or mitigate the emissions from taxi operations at airports.

Currently, three popular techniques regarding eco-friendly taxiing are tested with and discussed in previous research. These include single-engine taxiing, internal- and external AGPS. Of which the third one is most promising from an emissions reduction perspective, but also the most challenging one for full implementation in current airport infrastructure. Studies considering external AGPS, also called dispatch towing, are limited. And the studies that do investigate the implementation of this system at an operational level do not consider macro management of tow-trucks such as coupling and decoupling from aircraft or handling collision and conflict avoidance. These factors are important when testing operational feasibility of dispatch towing in current airport infrastructure.
Therefore this literature study concludes that in order to contribute to the development of dispatch towing and its successful adoption by the industry, research that focuses on macro management of tow-trucks and their interactions with other vehicles and aircraft is needed. This will allow for operational feasibility statements. Economical and environmental effects have been studied too a larger extend already.

For this to be achieved a simulation must be created that can handle various airport layouts and situations, all focused on the LTO cycle of aircraft. This is the only way in order to effectively study the micro management of the vehicles in real scenarios. Discrete event simulation in combination with monte carlo will be used to emperically retrieve the underlying probability distributions of the important performance indicators. By adapting the simulations logic, multiple strategies can be tested resulting in different results such that optimal strategies can be found. This will include management strategies such as queue discipline but also variations of airport layouts. Adaptations to taxiways are probably necessary for optimal performance. The goal of this research is to investigate what these adaptations will look like and how traffic is handled most effectively.

Following from the problem stated above the general objective for this research is determined:

"To provide insight in the operational feasibility of dispatch towing by identifying challenges and providing corresponding resolutions related to both effective traffic management of ground vehicles as well as adaptations to airport infrastructure"

5.2. Research questions

Corresponding to the problem statement and research objective, a set of questions have been established that will guide this thesis. The main question asks the following:

"How should current airport infrastructure be adapted in order to handle dispatch towing as an alternative to conventional taxiing without aircraft on-time performance loss?"

5.2.1. Sub questions

As such, the research will focus on answering the following questions:

1. What are the elements of conventional aircraft taxiing and how is this regulated?
2. What are the typical elements in an airports layout and how can we generalize these in an environment suited for the simulation model?
3. What are the performance characteristics of aircraft and tow-truck and how can these be processed by the simulation model?
4. How can real life flight demand be modeled accurately and brought as input to the simulation model?
5. Is the model resulting from this research valid?
6. Which aspects of dispatch towing result in an increase in flight delay?
7. What are the most optimal circumstances such to minimize the total delay?

Bibliography

[1] Airbus. *A320neo: Creating higher customer value*. June 2021. URL: https://aircraft.airbus.com/en/aircraft/a320/a320neo-creating-higher-customer-value.

[2] Airbus. *A350 less weight. less fuel. more sustainable.* June 2021. URL: https://aircraft.airbus.com/en/aircraft/a350/a350-less-weight-less-fuel-more-sustainable.

[3] AirInsightGroup. *Boeing improves performance estimates for 737 MAX*. Apr. 2021. URL: https://airinsight.com/boeing-improves-performance-estimates-for-737-max/.

[4] E V M Van Baaren. *The feasibility of a fully electric aircraft towing system Master Thesis*. 2019. URL: http://repository.tudelft.nl/..

[5] Indira Deonandan and Hamsa Balakrishnan. "Evaluation of Strategies for Reducing Taxi-out Emissions at Airports". In: *Proceedings of the 10th AIAA Aviation, Technology, Integrations, and Operations Conference (ATIO 2010)* (2010). DOI: 10.2514/6.2010-9370.

[6] European Commission - Climate Action. *"Reducing emissions from aviation"*. "2021". URL: https://ec.europa.eu/clima/eu-action/transport-emissions/reducing-emissions-aviation_en.

[7] Federal Aviation Administration. *"Advisor Circular - Aircraft Wake Turbulence"*. AC No: 90-23G. Feb. 2014. URL: https://www.faa.gov/documentLibrary/media/Advisory_Circular/AC_90-23G.pdf.

[8] Federal Aviation Administration. *"Air Traffic Organization Policy - Consolidated Wake Turbulence (CWT) Separation Standards"*. Order JO 7110,126A. Sept. 2019. URL: https://www.faa.gov/documentLibrary/media/Order/JO_7110.126A.pdf.

[9] Federal Aviation Administration. *"FAA Order JO 7110.65Z - Air Traffic Control"*. Sept. 2019. URL: https://www.faa.gov/documentLibrary/media/Order/JO_7110.126A.pdf.

[10] Donald Gross et al. *Fundamentals of queueing theory*. en. 4th ed. Wiley Series in Probability and Statistics. Chichester, England: Wiley-Blackwell, July 2008.

[11] Daniel Guimarans et al. "Using simulation to estimate critical paths and survival functions in aircraft turnaround processes". In: Dec. 2017. DOI: 10.1109/WSC.2017.8248055.

[12] Rui Guo, Yu Zhang, and Qing Wang. "Comparison of emerging ground propulsion systems for electrified aircraft taxi operations". In: *Transportation Research Part C 44(2014):98-109* (2014). DOI: 10.1016/j.trc.2014.03.006.

[13] ICAO. *Attribute Values*. URL: https://www.icao.int/safety/airnavigation/AIG/Documents/ADREP%5C%20Taxonomy/ECCAIRS%5C%20Aviation%5C%201.3.0.12%5C%20(VL%5C%20for%5C%20AttrID%5C%20%5C%20391%5C%20-%5C%20Event%5C%20Phases).pdf.

[14] International Civil Aviation Organization. *Post-COVID-19 Forecasts Scenarios*. Tech. rep. June 2021. URL: https://www.icao.int/sustainability/Pages/Post-Covid-Forecasts-Scenarios.aspx.

[15] M Lukic et al. "State of the Art of Electric Taxiing Systems; State of the Art of Electric Taxiing Systems". In: *2018 IEEE International Conference on Electrical Systems for Aircraft, Railway, Ship Propulsion and Road Vehicles & International Transportation Electrification Conference (ESARS-ITEC)* (2018).

[16] Milos Lukic et al. "Review, Challenges, and Future Developments of Electric Taxiing Systems". In: *IEEE TRANSACTIONS ON TRANSPORTATION ELECTRIFICATION* 5 (4 2019). DOI: 10.1109/TTE.2019.2956862. URL: http://www.ieee.org/publications_standards/publications/rights/index.html.

[17] Jeff Overton. “The Growth in Greenhouse Gas Emissions from Commercial Aviation, Part 1 of a Series on Airlines and Climate Change”. In: *Environmental and Energy Study Institute* (2019).

[18] Fabrizio Re. “Assessing Environmental Benefits of Electric Aircraft Taxiing through Object-Oriented Simulation”. In: *SAE International Journal of Aerospace* 5 (2 Oct. 2012), pp. 503–512. ISSN: 19463855. DOI: 10.4271/2012-01-2218.

[19] Fabrizio Re. “Viability and state of the art of environmentally friendly aircraft taxiing systems”. In: (2012), pp. 1–6. DOI: 10.1109/ESARS.2012.6387462.

[20] KDC Resource. *The Boeing 777X and the future of Efficient Air Travel*. URL: https://www.kdcresource.com/insights/the-boeing-777x-and-the-future-of-efficient-air-travel/.

[21] Paul Roling. *AE444 Airport Design and Operation: Lecture 3 - Airside capacity*. Feb. 2022.

[22] Abdulrazaq Lemu Salihu, Shannon M. Lloyd, and Ali Akgunduz. “Electrification of airport taxiway operations: A simulation framework for analyzing congestion and cost”. In: *Transportation Research Part D: Transport and Environment* 97 (2021), p. 102962. ISSN: 1361-9209. DOI: https://doi.org/10.1016/j.trd.2021.102962. URL: https://www.sciencedirect.com/science/article/pii/S1361920921002601.

[23] Michael Schultz et al. “Dynamic turnaround management in a highly automated airport environment”. In: *28th Congress of the International Council of the Aeronautical Sciences 2012, ICAS 2012* 6 (Jan. 2012), pp. 4362–4371.

[24] SLOCAT. *Tracking Trends in a Time of Change: The Need for Radical Action Towards Sustainable Transport Decarbonisation, Transport and Climate Change Global Status Report – 2nd edition*. Tech. rep. 2021. URL: www.tcc-gsr.com.

[25] Sabine Soepnel et al. “Impact of Electric Taxi Systems on Airport Apron Operations and Gate Congestion”. In: *American Institute of Aeronautics and Astronautics* (2017). DOI: 10.2514/6.2017-4391.

[26] Mojdeh Soltani et al. “An eco-friendly aircraft taxiing approach with collision and conflict avoidance”. In: *Transportation Research Part C: Emerging Technologies* 121 (2020), p. 102872. ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2020.102872.

III

Supporting work

1

Appendix 1 - DES output data

Input μ_{Ts}	Input σ_{Ts}	Result μ_{Ts}	Result σ_{Ts}	Lim_s	$\#T$	T_{tot}	T_{mov}	T_{dly}	RD	TR	DTR
0	0	0.000	0.000	1	12	370.326	226.211	144.116	0.371	1.000	34.429
0	0	0.000	0.000	2	12	370.002	226.263	143.739	0.371	1.000	34.431
0	0	0.000	0.000	3	12	368.314	226.195	142.119	0.368	1.000	34.432
0	0	0.000	0.000	3	12	368.822	226.175	142.647	0.369	1.000	34.427
0	0	0.000	0.000	4	12	369.984	226.229	143.755	0.371	1.000	34.434
0	0	0.000	0.000	5	12	369.630	226.178	143.453	0.370	1.000	34.432
0	0	0.000	0.000	6	12	370.220	226.185	144.035	0.371	1.000	34.430
0	0	0.000	0.000	7	12	370.918	226.230	144.688	0.372	1.000	34.423
0	0	0.000	0.000	8	12	370.197	226.228	143.970	0.371	1.000	34.434
10	30	27.887	19.848	1	12	408.602	226.075	154.681	0.363	0.999	34.402
10	30	27.973	19.899	2	12	408.260	226.075	154.248	0.362	0.999	34.393
10	30	27.960	19.907	3	12	409.562	226.193	155.450	0.364	0.999	34.400
10	30	27.989	19.827	3	12	409.305	226.231	155.126	0.363	0.998	34.399
10	30	27.934	19.854	4	12	410.153	226.158	156.099	0.365	0.999	34.398
10	30	27.971	19.864	5	12	409.698	226.225	155.540	0.364	0.999	34.398
10	30	28.056	19.947	6	12	408.761	226.075	154.668	0.363	0.999	34.400
10	30	27.919	19.840	7	12	409.897	226.140	155.875	0.365	0.999	34.396
10	30	27.962	19.986	8	12	409.782	226.200	155.658	0.364	0.999	34.403
20	30	32.802	21.795	1	12	416.549	226.123	157.684	0.363	0.998	34.388
20	30	32.789	21.782	2	12	416.561	226.151	157.671	0.363	0.998	34.392
20	30	32.748	21.807	3	12	416.530	226.256	157.597	0.363	0.998	34.397
20	30	32.703	21.799	3	12	417.893	226.079	159.165	0.366	0.998	34.395
20	30	33.008	21.875	4	12	415.853	226.182	156.730	0.362	0.998	34.390
20	30	32.733	21.787	5	12	415.572	226.154	156.737	0.362	0.998	34.394
20	30	32.862	21.749	6	12	414.791	226.119	155.869	0.360	0.998	34.388
20	30	32.850	21.904	7	12	415.712	226.161	156.765	0.362	0.998	34.392
20	30	32.745	21.706	8	12	416.254	226.127	157.440	0.363	0.998	34.394
30	30	38.616	23.778	1	12	425.737	226.093	161.121	0.364	0.998	34.378
30	30	38.616	23.758	2	12	422.805	226.035	158.255	0.360	0.997	34.382
30	30	0.000	0.000	3	0	347.962	191.465	156.497	0.432	0.000	34.573
30	30	38.851	23.602	3	2	390.156	199.591	181.528	0.451	0.233	34.440
30	30	38.481	23.662	3	4	407.174	207.068	182.641	0.435	0.454	34.394
30	30	38.630	23.738	3	6	418.851	214.272	179.272	0.414	0.655	34.389
30	30	38.498	23.740	3	8	420.258	220.280	168.049	0.385	0.829	34.393
30	30	38.507	23.718	3	10	423.555	224.544	162.249	0.368	0.955	34.389
30	30	38.661	23.771	3	12	423.111	226.132	158.416	0.360	0.997	34.386
30	30	38.693	23.746	3	12	424.832	226.085	160.145	0.362	0.998	34.387
30	30	38.788	23.832	3	12	423.320	226.185	158.438	0.360	0.998	34.389
30	30	38.616	23.703	3	14	422.838	226.209	158.012	0.359	1.000	34.385
30	30	38.673	23.809	3	16	423.968	226.172	159.122	0.360	1.000	34.386
30	30	38.595	23.692	3	18	422.431	226.200	157.635	0.358	1.000	34.392
30	30	38.631	23.800	3	20	424.237	226.195	159.411	0.361	1.000	34.385
30	30	38.610	23.716	4	12	424.781	226.175	160.085	0.362	0.998	34.381
30	30	38.545	23.725	5	12	425.438	226.129	160.846	0.363	0.998	34.383

Input μ_{Ts}	Input σ_{Ts}	Result μ_{Ts}	Result σ_{Ts}	Lim_s	$\#T$	T_{tot}	T_{mov}	T_{dly}	RD	TR	DTR
30	30	38.705	23.731	6	12	425.667	226.171	160.890	0.363	0.997	34.387
30	30	38.622	23.762	7	12	424.635	226.101	160.005	0.362	0.998	34.385
30	30	38.618	23.705	8	12	421.863	226.091	157.243	0.358	0.998	34.384
40	30	45.332	25.563	1	12	432.411	226.080	161.143	0.359	0.997	34.374
40	30	45.351	25.436	2	12	433.779	226.064	162.513	0.360	0.997	34.375
40	30	45.427	25.618	3	12	432.315	226.048	160.986	0.358	0.997	34.375
40	30	45.408	25.528	3	12	432.266	226.112	160.899	0.358	0.997	34.375
40	30	45.422	25.471	4	12	433.937	226.061	162.605	0.360	0.997	34.374
40	30	45.463	25.555	5	12	434.225	226.100	162.825	0.360	0.996	34.377
40	30	45.405	25.533	6	12	433.147	226.078	161.825	0.359	0.996	34.378
40	30	45.319	25.389	7	12	433.110	226.116	161.830	0.359	0.997	34.373
40	30	45.336	25.518	8	12	432.713	226.195	161.331	0.358	0.997	34.376
50	30	52.988	27.028	1	12	443.001	226.038	164.213	0.357	0.996	34.356
50	30	53.130	26.969	2	12	442.499	225.973	163.621	0.356	0.996	34.367
50	30	53.283	27.108	3	12	441.745	226.040	162.674	0.354	0.995	34.370
50	30	53.044	27.033	3	12	441.806	225.995	163.008	0.355	0.995	34.367
50	30	53.130	27.077	4	12	442.539	226.073	163.575	0.356	0.995	34.370
50	30	53.017	26.889	5	12	442.718	226.043	163.905	0.356	0.995	34.370
50	30	53.237	27.019	6	12	442.088	226.020	163.087	0.355	0.995	34.367
50	30	53.148	27.016	7	12	442.334	226.104	163.312	0.355	0.996	34.364
50	30	53.053	27.135	8	12	442.682	226.034	163.835	0.356	0.995	34.371
60	0	60.000	0.000	1	12	429.973	226.098	144.055	0.321	0.997	34.357
60	0	60.000	0.000	2	12	431.895	226.059	146.080	0.324	0.996	34.362
60	0	60.000	0.000	3	12	431.565	226.071	145.731	0.324	0.996	34.365
60	0	60.000	0.000	4	12	429.202	226.058	143.389	0.320	0.996	34.366
60	0	60.000	0.000	5	12	429.937	226.042	144.132	0.321	0.996	34.369
60	0	60.000	0.000	6	12	430.827	226.088	145.002	0.323	0.996	34.366
60	0	60.000	0.000	7	12	430.770	226.056	144.955	0.323	0.996	34.367
60	0	60.000	0.000	8	12	428.351	225.993	142.603	0.319	0.996	34.362
60	30	61.596	28.181	1	12	453.462	226.040	166.202	0.353	0.994	34.344
60	30	61.461	28.130	2	12	450.071	226.015	162.956	0.349	0.994	34.360
60	30	0.000	0.000	3	0	346.626	191.453	155.173	0.430	0.000	34.575
60	30	61.583	27.875	3	2	414.199	199.074	201.476	0.474	0.222	34.411
60	30	61.660	28.146	3	4	433.679	206.335	200.667	0.452	0.433	34.375
60	30	61.780	28.165	3	6	444.170	213.143	192.265	0.422	0.627	34.374
60	30	61.659	28.101	3	8	450.142	219.344	181.477	0.390	0.800	34.362
60	30	61.576	28.161	3	10	450.543	223.857	169.227	0.362	0.933	34.358
60	30	61.611	28.278	3	12	452.416	225.841	165.371	0.352	0.993	34.361
60	30	61.810	28.116	3	12	452.510	225.994	165.083	0.351	0.994	34.362
60	30	61.707	28.132	3	12	449.378	225.954	162.100	0.347	0.994	34.359
60	30	61.738	28.169	3	14	453.469	226.217	165.514	0.351	1.000	34.354
60	30	61.642	28.134	3	16	449.820	226.257	161.921	0.346	1.000	34.360
60	30	61.630	28.179	3	18	451.512	226.196	163.686	0.349	1.000	34.354
60	30	61.542	28.160	3	20	451.313	226.213	163.557	0.349	1.000	34.359
60	30	61.514	28.098	4	12	451.362	225.981	164.253	0.351	0.994	34.362
60	30	61.650	28.181	5	12	452.069	225.942	164.875	0.351	0.994	34.362
60	30	61.534	28.178	6	12	451.491	226.074	164.269	0.350	0.994	34.359
60	30	61.598	28.084	7	12	450.631	225.971	163.455	0.349	0.994	34.356
60	30	61.699	28.211	8	12	451.557	225.954	164.285	0.350	0.994	34.359
60	60	77.331	47.449	1	12	563.165	225.173	263.025	0.451	0.970	34.253
60	60	77.228	47.415	2	12	509.758	225.707	208.001	0.396	0.985	34.319
60	60	77.454	47.509	3	12	506.364	225.626	204.488	0.391	0.984	34.330
60	60	77.064	47.354	4	12	503.856	225.621	202.384	0.390	0.984	34.331
60	60	77.432	47.472	5	12	505.004	225.633	203.174	0.390	0.984	34.328
60	60	77.086	47.384	6	12	505.852	225.495	204.487	0.392	0.984	34.328
60	60	77.451	47.761	7	12	508.153	225.681	206.248	0.393	0.984	34.327
60	60	77.051	47.346	8	12	504.595	225.640	203.156	0.390	0.984	34.327
70	30	70.770	29.015	1	12	467.638	225.932	171.562	0.354	0.991	34.322
70	30	70.717	28.975	2	12	460.573	225.810	164.609	0.344	0.992	34.340
70	30	70.907	29.007	3	12	459.354	225.965	163.068	0.342	0.992	34.345
70	30	70.703	28.938	3	12	460.517	225.917	164.490	0.344	0.992	34.348
70	30	70.689	28.974	4	12	461.246	225.958	165.222	0.345	0.991	34.345
70	30	70.917	29.079	5	12	460.719	225.917	164.490	0.344	0.991	34.352
70	30	70.801	28.898	6	12	461.759	225.897	165.685	0.346	0.991	34.346
70	30	70.825	28.950	7	12	460.951	225.983	164.748	0.344	0.991	34.349
70	30	70.602	28.949	8	12	459.545	225.876	163.689	0.343	0.991	34.347
80	30	80.301	29.356	1	12	486.380	225.823	181.217	0.360	0.988	34.302
80	30	80.350	29.463	2	12	471.099	225.783	165.804	0.339	0.990	34.335
80	30	80.380	29.521	3	12	470.645	225.715	165.441	0.339	0.989	34.343
80	30	80.300	29.517	3	12	470.869	225.816	165.619	0.339	0.989	34.340
80	30	80.336	29.625	4	12	472.676	225.830	167.430	0.342	0.989	34.334
80	30	80.415	29.494	5	12	472.889	225.757	167.654	0.342	0.988	34.340

Input μ_{Ts}	Input σ_{Ts}	Result μ_{Ts}	Result σ_{Ts}	Lim_s	$\#T$	T_{tot}	T_{mov}	T_{dly}	RD	TR	DTR
80	30	80.267	29.443	6	12	470.363	225.762	165.219	0.339	0.989	34.337
80	30	80.450	29.433	7	12	470.531	225.757	165.222	0.338	0.989	34.340
80	30	80.453	29.385	8	12	469.601	225.793	164.291	0.337	0.988	34.339
90	30	90.133	29.723	1	12	522.383	225.505	208.538	0.386	0.980	34.273
90	30	90.088	29.774	2	12	482.276	225.698	167.767	0.336	0.986	34.327
90	30	0.000	0.000	3	0	347.533	191.436	156.097	0.431	0.000	34.572
90	30	89.958	29.559	3	2	441.428	198.804	223.795	0.498	0.209	34.383
90	30	90.384	29.629	3	4	460.986	205.725	218.202	0.466	0.410	34.346
90	30	90.156	29.633	3	6	472.183	212.095	206.305	0.428	0.597	34.343
90	30	90.116	29.816	3	8	477.698	218.011	190.724	0.390	0.765	34.333
90	30	90.138	29.709	3	10	480.487	222.841	176.037	0.355	0.905	34.332
90	30	90.130	29.595	3	12	481.831	225.698	167.286	0.335	0.986	34.323
90	30	90.041	29.723	3	12	481.330	225.690	166.906	0.335	0.985	34.325
90	30	90.115	29.756	3	12	480.737	225.648	166.326	0.334	0.985	34.321
90	30	89.952	29.678	3	14	482.099	226.315	165.844	0.332	1.000	34.325
90	30	90.060	29.737	3	16	482.359	226.167	166.132	0.332	1.000	34.326
90	30	90.078	29.681	3	18	482.888	226.144	166.666	0.333	1.000	34.326
90	30	90.255	29.624	3	20	482.688	226.156	166.277	0.332	1.000	34.327
90	30	90.116	29.701	4	12	479.577	225.847	164.937	0.332	0.985	34.325
90	30	90.106	29.770	5	12	481.833	225.704	167.429	0.335	0.984	34.325
90	30	90.107	29.695	6	12	480.897	225.667	166.481	0.334	0.985	34.329
90	30	90.092	29.736	7	12	479.678	225.590	165.323	0.333	0.985	34.326
90	30	90.174	29.768	8	12	479.770	225.727	165.230	0.332	0.985	34.323
100	30	100.080	29.734	1	12	603.765	224.742	283.146	0.451	0.958	34.187
100	30	100.123	29.838	2	12	494.384	225.588	170.461	0.333	0.982	34.307
100	30	99.967	29.830	3	12	492.045	225.478	168.428	0.330	0.982	34.313
100	30	100.204	29.840	3	12	491.804	225.649	167.822	0.330	0.981	34.312
100	30	99.994	29.859	4	12	491.062	225.616	167.376	0.329	0.981	34.317
100	30	99.932	29.770	5	12	489.912	225.510	166.405	0.328	0.981	34.321
100	30	100.061	29.946	6	12	491.766	225.579	168.060	0.330	0.981	34.318
100	30	100.127	29.834	7	12	490.931	225.513	167.231	0.329	0.981	34.315
100	30	100.020	29.751	8	12	490.559	225.501	166.973	0.329	0.981	34.308
110	30	110.039	29.823	1	12	728.917	223.022	405.922	0.539	0.909	33.871
110	30	110.121	29.876	2	12	506.188	225.425	173.112	0.331	0.978	34.294
110	30	109.930	29.912	3	12	502.467	225.364	169.765	0.327	0.976	34.298
110	30	110.052	29.901	3	12	503.023	225.389	170.143	0.327	0.977	34.299
110	30	110.025	29.852	4	12	501.917	225.481	168.997	0.326	0.976	34.302
110	30	110.042	29.842	5	12	502.131	225.409	169.340	0.326	0.976	34.305
110	30	109.992	29.947	6	12	500.065	225.365	167.413	0.324	0.975	34.308
110	30	109.982	30.004	7	12	502.629	225.374	170.020	0.327	0.975	34.302
110	30	110.104	29.785	8	12	501.817	225.308	169.168	0.326	0.975	34.304
120	0	120.000	0.000	1	12	785.800	221.541	460.303	0.576	0.866	33.079
120	0	120.000	0.000	2	12	491.786	225.362	149.057	0.292	0.978	34.283
120	0	120.000	0.000	3	12	492.527	225.407	149.942	0.293	0.976	34.297
120	0	120.000	0.000	4	12	490.046	225.443	147.494	0.290	0.976	34.301
120	0	120.000	0.000	5	12	489.853	225.418	147.408	0.290	0.975	34.298
120	0	120.000	0.000	6	12	489.536	225.190	147.384	0.290	0.975	34.301
120	0	120.000	0.000	7	12	490.786	225.344	148.499	0.291	0.975	34.303
120	0	120.000	0.000	8	12	491.001	225.354	148.648	0.292	0.975	34.301
120	5	119.969	5.013	3	12	491.355	225.327	148.851	0.292	0.977	34.293
120	10	119.988	9.972	1	12	800.890	221.340	476.104	0.584	0.862	33.190
120	10	120.036	9.998	2	12	496.171	225.383	153.455	0.298	0.977	34.280
120	10	119.950	9.986	3	12	493.350	225.457	150.816	0.294	0.976	34.290
120	10	120.026	9.971	3	12	495.977	225.399	153.465	0.298	0.976	34.294
120	10	119.989	9.994	3	12	493.839	225.427	151.284	0.295	0.976	34.297
120	10	120.025	9.986	4	12	492.967	225.304	150.655	0.294	0.975	34.297
120	10	119.978	10.000	5	12	493.308	225.306	151.126	0.295	0.974	34.290
120	10	119.981	9.953	6	12	493.280	225.319	151.142	0.295	0.974	34.298
120	10	119.998	9.980	7	12	494.242	225.226	152.176	0.297	0.974	34.298
120	10	119.985	9.956	8	12	493.500	225.319	151.366	0.296	0.974	34.296
120	15	119.972	14.946	3	12	497.245	225.341	154.918	0.301	0.975	34.295
120	20	120.045	19.902	1	12	819.584	221.104	495.774	0.592	0.856	33.331
120	20	120.041	19.922	2	12	503.841	225.291	161.490	0.310	0.975	34.279
120	20	120.088	19.993	3	12	500.976	225.307	158.691	0.306	0.974	34.294
120	20	119.930	20.003	3	12	501.256	225.239	159.166	0.307	0.974	34.287
120	20	119.993	19.987	3	12	500.707	225.305	158.513	0.306	0.974	34.293
120	20	119.979	19.962	4	12	499.803	225.194	157.833	0.305	0.973	34.298
120	20	119.930	19.952	5	12	500.556	225.162	158.718	0.306	0.973	34.292
120	20	119.908	19.967	6	12	501.773	225.270	159.907	0.308	0.972	34.296
120	20	120.081	19.953	7	12	499.610	225.221	157.640	0.305	0.972	34.299
120	20	119.866	19.892	8	12	501.448	225.152	159.722	0.308	0.973	34.296
120	25	120.029	24.935	3	12	505.529	225.173	163.588	0.313	0.973	34.294

Input μ_{Ts}	Input σ_{Ts}	Result μ_{Ts}	Result σ_{Ts}	Lim_s	$\#T$	T_{tot}	T_{mov}	T_{dly}	RD	TR	DTR
120	30	119.918	29.929	1	12	847.309	220.871	524.809	0.605	0.848	33.466
120	30	119.995	29.877	1	12	843.334	220.868	520.907	0.602	0.846	33.462
120	30	119.912	29.808	2	12	519.551	225.366	177.660	0.331	0.972	34.275
120	30	119.913	29.855	2	12	519.541	225.130	177.930	0.332	0.971	34.275
120	30	0.000	0.000	3	0	346.539	191.316	155.223	0.430	0.000	34.574
120	30	120.109	29.651	3	2	470.569	198.257	248.518	0.522	0.198	34.360
120	30	119.996	29.704	3	4	488.428	204.819	236.985	0.481	0.389	34.316
120	30	120.037	29.972	3	6	500.057	211.147	220.853	0.437	0.567	34.311
120	30	119.945	29.898	3	8	507.585	216.771	203.160	0.393	0.731	34.306
120	30	120.176	29.903	3	10	512.155	221.758	185.425	0.353	0.874	34.291
120	30	120.103	29.851	3	12	513.060	225.221	171.174	0.323	0.971	34.289
120	30	120.065	29.798	3	12	511.503	225.309	169.583	0.321	0.971	34.290
120	30	120.002	29.913	3	12	514.531	225.132	172.869	0.325	0.971	34.290
120	30	120.116	29.964	3	12	514.051	225.243	172.100	0.324	0.972	34.292
120	30	120.169	29.973	3	12	513.734	225.250	171.738	0.324	0.972	34.291
120	30	119.953	29.776	3	12	513.313	225.127	171.710	0.324	0.971	34.287
120	30	120.037	29.923	3	14	513.869	226.133	167.795	0.316	0.999	34.284
120	30	120.001	29.893	3	16	514.142	226.151	167.990	0.316	1.000	34.289
120	30	119.919	29.794	3	18	515.133	226.217	168.996	0.317	1.000	34.287
120	30	119.961	29.939	3	20	513.134	226.126	167.048	0.314	1.000	34.289
120	30	120.098	29.871	4	12	511.427	225.093	169.790	0.321	0.970	34.299
120	30	119.953	29.951	4	12	513.289	225.039	171.867	0.324	0.970	34.290
120	30	120.005	29.968	5	12	509.806	225.153	168.227	0.319	0.970	34.297
120	30	120.014	29.924	5	12	509.477	225.142	167.921	0.319	0.970	34.294
120	30	119.821	29.900	6	12	510.582	225.187	169.167	0.321	0.970	34.295
120	30	120.106	29.878	6	12	510.997	225.155	169.339	0.321	0.970	34.295
120	30	119.922	29.887	7	12	511.458	225.128	170.058	0.322	0.970	34.297
120	30	119.902	29.977	7	12	510.933	225.119	169.459	0.321	0.970	34.295
120	30	119.845	29.828	8	12	512.105	225.203	170.731	0.323	0.969	34.296
120	30	120.066	29.834	8	12	512.557	225.187	170.847	0.323	0.970	34.293
120	35	120.077	34.792	3	12	521.903	225.160	180.437	0.335	0.969	34.285
120	40	120.011	39.580	1	12	884.374	220.493	563.681	0.621	0.835	33.517
120	40	120.195	39.611	2	12	540.982	225.128	199.677	0.359	0.967	34.269
120	40	120.225	39.628	3	12	528.862	224.999	187.527	0.344	0.968	34.279
120	40	120.117	39.696	3	12	530.350	225.121	189.051	0.346	0.967	34.280
120	40	120.136	39.630	3	12	530.382	225.164	188.992	0.346	0.967	34.288
120	40	120.244	39.781	4	12	528.045	225.099	186.651	0.343	0.967	34.291
120	40	120.334	39.656	5	12	527.810	225.050	186.431	0.343	0.967	34.292
120	40	120.163	39.625	6	12	525.711	225.021	184.543	0.341	0.967	34.284
120	40	120.287	39.635	7	12	526.688	225.128	185.255	0.341	0.967	34.287
120	40	120.167	39.706	8	12	524.615	225.076	183.402	0.339	0.966	34.292
120	45	120.500	44.149	3	12	540.776	224.975	199.547	0.358	0.965	34.279
120	50	121.045	48.499	1	12	937.173	219.860	618.245	0.641	0.819	33.501
120	50	121.162	48.386	2	12	570.124	224.732	229.172	0.391	0.959	34.255
120	50	121.118	48.566	3	12	553.315	224.804	212.040	0.373	0.962	34.270
120	50	120.976	48.459	3	12	552.451	224.816	211.294	0.372	0.962	34.272
120	50	121.176	48.316	3	12	550.895	224.928	209.397	0.370	0.962	34.272
120	50	121.380	48.398	4	12	544.102	224.808	202.470	0.362	0.962	34.275
120	50	121.101	48.582	5	12	543.835	224.945	202.312	0.362	0.963	34.282
120	50	120.955	48.447	6	12	540.917	224.847	199.794	0.359	0.961	34.283
120	50	121.091	48.551	7	12	543.021	224.804	201.680	0.361	0.962	34.274
120	50	121.056	48.537	8	12	543.887	224.819	202.591	0.362	0.962	34.282
120	55	122.146	52.352	3	12	562.800	224.729	220.898	0.382	0.959	34.266
120	60	123.409	56.304	1	12	994.190	219.199	676.698	0.662	0.797	33.480
120	60	123.357	56.442	2	12	602.248	224.416	260.818	0.422	0.949	34.230
120	60	123.493	56.435	3	12	576.493	224.580	233.999	0.395	0.955	34.259
120	60	123.055	56.374	3	12	574.194	224.602	232.007	0.394	0.956	34.255
120	60	123.345	56.259	3	12	576.321	224.664	233.845	0.396	0.955	34.258
120	60	123.118	56.357	4	12	567.938	224.701	225.538	0.387	0.956	34.264
120	60	123.383	56.297	5	12	563.532	224.650	220.885	0.382	0.956	34.263
120	60	123.113	56.371	6	12	558.577	224.606	216.165	0.377	0.957	34.267
120	60	123.337	56.163	7	12	560.033	224.585	217.525	0.379	0.956	34.273
120	60	123.301	56.444	8	12	560.518	224.647	217.941	0.379	0.956	34.274
120	65	124.663	59.954	3	12	591.932	224.476	248.878	0.409	0.951	34.246
120	70	126.807	63.546	1	12	1074.317	218.149	758.336	0.686	0.772	33.383
120	70	126.514	63.430	2	12	638.883	224.021	296.410	0.453	0.936	34.195
120	70	127.011	63.437	3	12	601.894	224.389	257.242	0.417	0.947	34.232
120	70	126.490	63.673	3	12	601.737	224.306	257.758	0.418	0.946	34.237
120	70	126.840	63.581	3	12	599.168	224.313	254.790	0.415	0.947	34.234
120	70	126.720	63.607	4	12	592.430	224.438	247.775	0.408	0.949	34.250
120	70	126.661	63.396	5	12	588.066	224.463	243.369	0.404	0.949	34.248
120	70	126.724	63.579	6	12	585.183	224.372	240.626	0.402	0.948	34.250

Input μ_{Ts}	Input σ_{Ts}	Result μ_{Ts}	Result σ_{Ts}	Lim_s	$\#T$	T_{tot}	T_{mov}	T_{dly}	RD	TR	DTR
120	70	127.096	63.470	7	12	584.881	224.377	239.770	0.400	0.950	34.254
120	70	126.456	63.260	8	12	583.388	224.341	239.000	0.400	0.949	34.249
120	75	129.037	66.898	3	12	614.878	224.130	269.199	0.428	0.942	34.219
120	80	131.310	70.233	1	12	1145.641	217.323	830.553	0.706	0.745	33.300
120	80	131.384	70.106	2	12	689.005	223.421	344.768	0.487	0.920	34.133
120	80	131.208	69.920	3	12	632.814	223.953	286.016	0.441	0.936	34.202
120	80	131.214	70.118	3	12	631.217	223.800	284.633	0.441	0.936	34.209
120	80	130.829	70.238	3	12	632.198	223.990	285.673	0.441	0.937	34.204
120	80	130.956	70.152	4	12	619.931	224.108	272.816	0.430	0.939	34.216
120	80	131.422	70.110	5	12	610.307	224.237	262.411	0.420	0.941	34.226
120	80	130.915	70.267	6	12	608.995	224.194	261.522	0.420	0.942	34.234
120	80	131.445	70.178	7	12	605.747	224.153	257.734	0.416	0.942	34.234
120	80	130.752	69.903	8	12	602.357	224.154	255.049	0.414	0.942	34.241
120	85	133.406	73.192	3	12	645.076	223.910	296.955	0.450	0.931	34.190
120	90	136.575	76.478	1	12	1219.803	216.273	905.968	0.722	0.715	33.169
120	90	136.164	76.292	2	12	742.352	222.781	397.074	0.520	0.900	34.067
120	90	136.298	76.868	3	12	659.987	223.571	310.331	0.460	0.925	34.173
120	90	136.588	76.671	3	12	666.610	223.540	316.944	0.465	0.924	34.172
120	90	136.138	76.505	3	12	666.529	223.435	317.241	0.465	0.925	34.174
120	90	135.931	76.512	4	12	647.906	223.740	297.779	0.449	0.930	34.194
120	90	136.406	76.688	5	12	639.577	223.797	288.763	0.442	0.931	34.200
120	90	135.745	76.251	6	12	634.062	223.942	283.576	0.438	0.932	34.210
120	90	136.200	76.410	7	12	632.609	223.928	281.688	0.436	0.932	34.211
120	90	136.451	76.573	8	12	629.768	223.877	278.485	0.433	0.934	34.213
120	95	139.140	79.593	3	12	673.724	223.403	322.610	0.468	0.918	34.153
120	100	142.418	82.511	1	12	1307.114	215.226	994.044	0.740	0.688	33.091
120	100	142.226	82.597	2	12	811.261	221.800	464.891	0.557	0.876	33.982
120	100	142.059	82.950	3	12	703.522	223.069	351.155	0.488	0.910	34.129
120	100	141.877	82.944	3	12	704.634	223.006	352.278	0.489	0.912	34.129
120	100	141.862	82.568	3	12	695.531	223.094	343.203	0.483	0.911	34.130
120	100	141.843	83.008	4	12	677.662	223.425	324.095	0.468	0.918	34.154
120	100	141.977	82.858	5	12	665.693	223.427	311.530	0.458	0.921	34.172
120	100	141.785	82.800	6	12	657.883	223.449	303.651	0.452	0.922	34.180
120	100	142.037	82.660	7	12	653.662	223.619	298.969	0.448	0.923	34.184
120	100	142.176	82.845	8	12	653.202	223.519	298.318	0.448	0.924	34.189
120	105	144.444	85.786	3	12	718.105	222.864	364.624	0.497	0.904	34.108
120	110	148.246	89.029	1	12	1400.019	214.338	1087.778	0.757	0.661	32.999
120	110	147.856	88.913	2	12	884.996	221.078	538.135	0.590	0.851	33.865
120	110	147.657	88.729	3	12	736.122	222.597	381.036	0.507	0.897	34.095
120	110	148.354	88.783	3	12	738.081	222.592	382.476	0.507	0.897	34.085
120	110	147.903	89.119	3	12	735.649	222.593	380.393	0.506	0.897	34.092
120	110	147.643	88.718	4	12	710.471	222.897	353.860	0.487	0.906	34.122
120	110	147.848	88.720	5	12	696.599	223.039	339.133	0.477	0.909	34.142
120	110	148.053	88.728	6	12	688.936	223.191	330.857	0.470	0.911	34.147
120	110	148.012	88.547	7	12	679.016	223.242	320.654	0.463	0.913	34.161
120	110	147.799	88.763	8	12	679.819	223.258	321.592	0.464	0.913	34.160
120	115	151.456	91.890	3	12	756.774	222.301	399.858	0.517	0.889	34.068
120	120	155.245	94.810	1	12	1460.292	213.333	1148.881	0.767	0.632	32.884
120	120	154.557	95.014	2	12	978.066	220.023	631.159	0.624	0.821	33.711
120	120	155.029	95.231	3	12	780.449	221.976	421.980	0.529	0.881	34.037
120	120	155.057	95.555	3	12	784.623	221.981	426.143	0.532	0.881	34.034
120	120	154.329	94.951	3	12	784.996	222.061	427.066	0.532	0.881	34.029
120	120	154.218	94.927	4	12	738.208	222.597	377.805	0.502	0.894	34.088
120	120	154.536	95.111	5	12	725.181	222.559	363.983	0.492	0.897	34.110
120	120	154.958	95.158	6	12	719.370	222.630	357.352	0.487	0.900	34.114
120	120	154.477	94.645	7	12	706.956	222.840	344.958	0.479	0.901	34.127
120	120	154.760	95.030	8	12	707.042	222.737	344.591	0.478	0.903	34.134
120	130	161.477	100.853	3	12	824.947	221.457	464.106	0.551	0.863	33.977
120	140	168.397	107.257	3	12	865.179	220.892	502.014	0.568	0.845	33.923
120	150	175.859	113.137	3	12	929.812	220.134	564.769	0.594	0.824	33.840
120	160	182.775	119.515	3	12	991.311	219.347	625.054	0.616	0.804	33.755
120	170	189.500	124.827	3	12	1054.856	218.611	687.887	0.636	0.783	33.663
120	180	197.011	130.713	3	12	1128.989	217.927	761.371	0.657	0.760	33.573
120	190	204.323	137.317	3	12	1206.352	217.080	838.727	0.676	0.737	33.456
120	200	211.320	143.173	3	12	1279.498	216.250	912.316	0.693	0.715	33.339
120	210	218.719	148.620	3	12	1358.081	215.558	991.276	0.711	0.692	33.230
120	220	228.200	154.822	3	12	1461.382	214.691	1094.455	0.729	0.668	33.100
120	230	234.662	160.764	3	12	1545.061	213.969	1179.525	0.742	0.647	32.982
120	240	242.421	166.402	3	12	1612.404	213.285	1247.599	0.753	0.626	32.882
120	250	249.464	173.107	3	12	1705.919	212.490	1342.600	0.766	0.606	32.746
120	260	258.517	179.083	3	12	1813.758	211.653	1451.677	0.778	0.583	32.638
120	270	267.153	185.113	3	12	1885.698	211.260	1523.765	0.786	0.565	32.522

Input μ_{Ts}	Input σ_{Ts}	Result μ_{Ts}	Result σ_{Ts}	Lim_s	$\#T$	T_{tot}	T_{mov}	T_{dly}	RD	TR	DTR
120	280	272.770	189.796	3	12	1982.360	210.383	1622.696	0.798	0.549	32.453
120	290	281.865	197.003	3	12	2115.229	209.899	1757.003	0.808	0.527	32.274
120	300	287.591	202.770	3	12	2157.116	209.383	1799.437	0.812	0.517	32.229
130	30	130.033	29.896	1	12	955.313	218.841	633.966	0.650	0.788	33.163
130	30	129.987	29.860	2	12	538.096	224.918	187.826	0.339	0.964	34.258
130	30	129.991	29.939	3	12	526.541	225.002	176.089	0.324	0.965	34.276
130	30	130.128	29.904	3	12	525.682	225.002	175.176	0.323	0.964	34.275
130	30	130.142	29.928	4	12	524.378	224.921	173.932	0.322	0.965	34.286
130	30	129.982	29.870	5	12	523.770	224.895	173.564	0.321	0.964	34.284
130	30	130.029	30.011	6	12	518.766	224.890	168.633	0.315	0.963	34.283
130	30	129.998	29.925	7	12	521.235	224.973	170.980	0.318	0.964	34.281
130	30	129.830	29.974	8	12	521.299	224.894	171.364	0.319	0.963	34.281
140	30	139.955	29.898	1	12	1051.504	217.104	731.225	0.684	0.737	33.025
140	30	139.958	30.006	2	12	558.867	224.568	200.550	0.349	0.956	34.238
140	30	140.051	29.873	3	12	541.655	224.730	182.917	0.328	0.957	34.258
140	30	140.070	29.973	3	12	541.530	224.696	182.796	0.328	0.957	34.261
140	30	139.928	29.938	4	12	534.727	224.678	176.207	0.320	0.957	34.270
140	30	139.977	29.884	5	12	533.850	224.606	175.456	0.319	0.956	34.270
140	30	140.104	29.858	6	12	531.320	224.660	172.608	0.315	0.957	34.270
140	30	140.059	29.956	7	12	533.080	224.579	174.579	0.318	0.956	34.271
140	30	140.001	29.892	8	12	532.416	224.638	173.956	0.317	0.956	34.272
150	30	149.852	30.152	1	12	1131.881	215.437	812.930	0.707	0.691	32.856
150	30	150.032	30.036	2	12	580.066	224.386	213.915	0.360	0.945	34.218
150	30	0.000	0.000	3	0	346.717	191.255	155.462	0.431	0.000	34.575
150	30	150.087	29.604	3	2	499.364	197.959	273.226	0.544	0.188	34.333
150	30	149.931	29.799	3	4	521.865	204.256	262.358	0.502	0.369	34.287
150	30	149.857	29.802	3	6	533.878	210.193	242.797	0.453	0.540	34.275
150	30	149.990	30.063	3	8	545.122	215.742	224.532	0.408	0.699	34.270
150	30	149.997	29.760	3	10	551.069	220.516	204.724	0.365	0.839	34.257
150	30	150.114	29.929	3	12	556.802	224.424	190.222	0.333	0.947	34.246
150	30	149.898	29.984	3	12	557.544	224.315	191.192	0.334	0.948	34.244
150	30	150.053	29.836	3	12	558.358	224.320	191.843	0.334	0.948	34.247
150	30	150.197	30.006	3	14	560.018	225.942	184.607	0.320	0.995	34.243
150	30	149.702	29.906	3	16	559.814	226.184	183.929	0.319	1.000	34.243
150	30	150.115	30.008	3	18	559.351	226.225	183.011	0.317	1.000	34.240
150	30	150.030	30.033	3	20	560.314	226.241	184.044	0.318	1.000	34.245
150	30	149.958	29.953	4	12	549.539	224.408	182.766	0.324	0.949	34.253
150	30	150.040	29.890	5	12	546.688	224.366	180.000	0.320	0.949	34.257
150	30	149.981	29.900	6	12	545.087	224.376	178.551	0.318	0.948	34.261
150	30	150.054	29.807	7	12	543.330	224.431	176.561	0.316	0.949	34.262
150	30	149.959	29.856	8	12	542.238	224.325	175.607	0.315	0.949	34.259
160	30	160.035	30.002	1	12	1219.437	214.139	901.227	0.729	0.650	32.709
160	30	160.035	29.999	2	12	603.352	223.822	230.347	0.374	0.932	34.196
160	30	159.956	29.939	3	12	577.335	224.049	203.408	0.344	0.937	34.227
160	30	160.054	29.932	3	12	576.382	223.907	202.466	0.343	0.937	34.233
160	30	159.995	29.865	4	12	565.411	224.029	191.165	0.330	0.939	34.240
160	30	159.973	29.844	5	12	561.420	224.029	187.147	0.325	0.939	34.244
160	30	160.058	29.967	6	12	557.016	224.141	182.478	0.319	0.940	34.246
160	30	160.160	29.839	7	12	556.271	224.141	181.547	0.318	0.940	34.247
160	30	159.979	29.758	8	12	553.238	224.106	178.588	0.314	0.941	34.250
170	30	170.036	29.827	1	12	1270.092	212.586	953.115	0.740	0.614	32.576
170	30	170.125	29.975	2	12	631.944	223.222	252.697	0.392	0.917	34.172
170	30	169.991	29.842	3	12	595.997	223.593	215.003	0.353	0.926	34.211
170	30	170.049	29.957	3	12	597.916	223.677	216.717	0.355	0.926	34.210
170	30	169.829	29.965	4	12	585.075	223.649	203.668	0.340	0.929	34.223
170	30	170.173	29.908	5	12	576.162	223.761	194.174	0.329	0.930	34.228
170	30	170.046	29.837	6	12	572.940	223.791	190.918	0.325	0.931	34.234
170	30	169.954	29.943	7	12	569.933	223.760	187.976	0.322	0.931	34.240
170	30	170.043	29.960	8	12	567.604	223.801	185.427	0.318	0.931	34.238
180	0	180.000	0.000	1	12	1214.340	211.745	897.243	0.736	0.585	32.362
180	0	180.000	0.000	2	12	610.836	223.334	222.574	0.358	0.916	34.166
180	0	180.000	0.000	3	12	583.797	223.430	194.659	0.326	0.921	34.207
180	0	180.000	0.000	4	12	573.907	223.677	183.863	0.313	0.924	34.221
180	0	180.000	0.000	5	12	568.512	223.690	178.268	0.306	0.925	34.228
180	0	180.000	0.000	6	12	563.905	223.662	173.482	0.300	0.926	34.231
180	0	180.000	0.000	7	12	560.342	223.636	169.745	0.295	0.928	34.232
180	0	180.000	0.000	8	12	559.223	223.513	168.681	0.294	0.928	34.231
180	30	179.934	29.794	1	12	1358.798	211.643	1042.441	0.758	0.582	32.444
180	30	180.070	29.905	2	12	666.829	222.655	282.388	0.416	0.898	34.133
180	30	0.000	0.000	3	0	347.480	191.380	156.100	0.431	0.000	34.573
180	30	180.072	29.561	3	2	524.198	197.708	294.411	0.561	0.178	34.304
180	30	179.939	29.951	3	4	547.301	203.731	280.443	0.515	0.351	34.241

Input μ_{Ts}	Input σ_{Ts}	Result μ_{Ts}	Result σ_{Ts}	Lim_s	$\#T$	T_{tot}	T_{mov}	T_{dly}	RD	TR	DTR
180	30	179.930	29.889	3	6	573.422	209.433	271.491	0.475	0.514	34.244
180	30	180.018	29.910	3	8	594.459	214.533	259.834	0.436	0.667	34.228
180	30	179.952	29.980	3	10	611.798	219.305	248.174	0.401	0.802	34.204
180	30	179.929	29.883	3	12	621.278	223.152	233.847	0.369	0.913	34.195
180	30	180.023	29.992	3	12	618.061	223.158	230.648	0.366	0.912	34.194
180	30	179.989	29.883	3	12	617.955	223.105	230.596	0.366	0.913	34.197
180	30	180.023	29.962	3	14	623.598	225.519	221.236	0.346	0.982	34.191
180	30	180.231	29.988	3	16	625.367	226.169	219.036	0.341	1.000	34.188
180	30	179.885	29.974	3	18	621.853	226.224	215.744	0.338	1.000	34.191
180	30	180.038	29.862	3	20	622.568	226.135	216.395	0.339	1.000	34.186
180	30	180.019	29.928	4	12	604.789	223.270	216.426	0.351	0.917	34.210
180	30	180.011	29.932	5	12	597.119	223.369	208.379	0.341	0.919	34.221
180	30	180.104	29.888	6	12	590.020	223.402	200.820	0.333	0.921	34.216
180	30	180.040	29.926	7	12	586.518	223.429	197.314	0.329	0.921	34.224
180	30	180.018	29.983	8	12	583.202	223.527	193.655	0.325	0.922	34.227
180	60	180.387	59.287	1	12	1479.790	211.263	1165.177	0.774	0.573	32.480
180	60	180.309	59.290	2	12	882.498	220.431	511.671	0.564	0.834	33.843
180	60	180.238	59.400	3	12	691.393	222.458	308.096	0.438	0.892	34.140
180	60	180.024	59.296	4	12	668.432	222.671	283.915	0.417	0.899	34.161
180	60	180.339	59.165	5	12	658.191	222.827	272.671	0.406	0.902	34.178
180	60	180.139	59.319	6	12	648.961	222.891	263.166	0.398	0.904	34.188
180	60	180.060	59.559	7	12	641.932	222.877	255.940	0.391	0.906	34.192
180	60	180.127	59.308	8	12	634.024	222.909	247.759	0.383	0.907	34.197
180	90	184.837	84.232	1	12	1628.046	210.692	1315.535	0.793	0.551	32.470
180	90	185.120	84.678	2	12	1129.147	217.764	770.722	0.664	0.760	33.478
180	90	185.183	84.413	3	12	792.318	221.280	412.072	0.512	0.859	34.029
180	90	184.513	84.638	4	12	751.924	221.926	369.073	0.483	0.872	34.077
180	90	184.530	84.348	5	12	739.316	221.930	355.381	0.472	0.878	34.096
180	90	185.084	84.635	6	12	719.139	222.079	333.948	0.456	0.881	34.111
180	90	185.390	84.705	7	12	716.409	222.190	330.393	0.453	0.884	34.115
180	90	184.716	84.145	8	12	704.085	222.190	318.309	0.444	0.886	34.128
180	120	197.007	104.556	1	12	1819.976	209.326	1509.181	0.812	0.515	32.302
180	120	197.206	105.699	2	12	1380.476	215.403	1029.881	0.726	0.686	33.163
180	120	196.780	105.272	3	12	932.526	219.492	553.817	0.583	0.809	33.846
180	120	196.816	105.316	4	12	850.148	220.532	464.818	0.538	0.837	33.961
180	120	196.419	105.253	5	12	827.114	220.826	440.055	0.524	0.846	33.994
180	120	196.721	104.741	6	12	809.084	220.998	420.615	0.511	0.851	34.012
180	120	196.576	105.198	7	12	794.909	221.193	405.770	0.502	0.854	34.025
180	120	196.928	105.147	8	12	791.602	221.214	401.688	0.499	0.857	34.033
190	30	190.130	30.013	1	12	1407.476	210.574	1091.897	0.768	0.552	32.359
190	30	190.065	29.872	2	12	732.476	221.547	346.076	0.464	0.867	34.049
190	30	190.037	29.856	3	12	639.678	222.727	245.917	0.378	0.900	34.178
190	30	190.215	29.854	3	12	641.014	222.673	247.222	0.379	0.900	34.182
190	30	190.145	29.897	4	12	625.038	222.927	230.025	0.361	0.905	34.197
190	30	190.035	29.902	5	12	618.055	222.906	222.889	0.354	0.906	34.193
190	30	190.117	29.836	6	12	608.432	223.012	212.455	0.342	0.910	34.211
190	30	190.106	29.804	7	12	605.622	223.091	209.506	0.339	0.910	34.214
190	30	189.839	29.910	8	12	600.416	223.096	204.213	0.333	0.912	34.212
200	30	200.021	29.892	1	12	1480.904	209.826	1165.787	0.779	0.526	32.221
200	30	200.036	29.886	2	12	869.325	219.727	487.103	0.547	0.812	33.780
200	30	200.032	29.991	3	12	663.413	222.221	264.041	0.392	0.886	34.158
200	30	199.803	29.970	3	12	661.913	222.282	262.521	0.391	0.886	34.154
200	30	200.021	29.939	4	12	647.902	222.470	246.988	0.375	0.892	34.180
200	30	200.067	30.012	5	12	639.288	222.522	237.767	0.365	0.895	34.186
200	30	199.818	29.812	6	12	628.368	222.670	226.277	0.353	0.898	34.187
200	30	200.061	29.947	7	12	621.215	222.677	218.657	0.345	0.899	34.195
200	30	200.058	29.834	8	12	620.100	222.743	217.319	0.343	0.900	34.196
210	30	210.081	29.840	1	12	1526.107	209.014	1211.592	0.787	0.502	32.082
210	30	210.016	29.961	2	12	1064.949	217.283	692.146	0.632	0.741	33.456
210	30	0.000	0.000	3	0	348.075	191.292	156.783	0.432	0.000	34.573
210	30	210.322	29.743	3	2	554.824	197.075	322.017	0.582	0.170	34.268
210	30	210.337	29.813	3	4	580.111	203.033	306.764	0.534	0.334	34.211
210	30	210.018	29.821	3	6	620.352	208.526	308.833	0.501	0.490	34.203
210	30	210.057	29.978	3	8	655.169	213.487	308.614	0.473	0.634	34.172
210	30	209.939	29.885	3	10	673.601	217.875	295.925	0.438	0.761	34.151
210	30	210.098	29.927	3	12	685.041	221.746	280.221	0.404	0.871	34.135
210	30	209.884	29.997	3	12	684.135	221.669	279.518	0.404	0.872	34.140
210	30	210.024	29.893	3	12	685.476	221.761	280.702	0.405	0.871	34.141
210	30	209.947	29.982	3	14	689.776	224.707	263.963	0.376	0.958	34.131
210	30	209.996	29.960	3	16	691.363	225.970	256.360	0.363	0.995	34.126
210	30	209.957	29.934	3	18	689.941	226.142	253.842	0.360	1.000	34.128
210	30	209.912	29.837	3	20	689.695	226.251	253.532	0.360	1.000	34.124

Input μ_{Ts}	Input σ_{Ts}	Result μ_{Ts}	Result σ_{Ts}	Lim_s	$\#T$	T_{tot}	T_{mov}	T_{dly}	RD	TR	DTR
210	30	209.980	29.953	4	12	667.108	221.930	260.828	0.386	0.878	34.163
210	30	209.868	29.992	5	12	657.075	222.048	249.755	0.374	0.883	34.169
210	30	210.043	29.867	6	12	652.501	222.056	244.509	0.369	0.885	34.175
210	30	209.940	29.880	7	12	644.270	222.268	235.559	0.360	0.888	34.178
210	30	210.076	29.866	8	12	637.249	222.384	227.960	0.351	0.890	34.187
220	30	219.880	29.875	1	12	1596.386	208.061	1282.460	0.797	0.481	31.983
220	30	219.840	29.895	2	12	1236.889	215.030	872.159	0.687	0.681	33.245
220	30	219.913	29.918	3	12	705.479	221.316	295.577	0.415	0.858	34.120
220	30	220.065	30.009	3	12	704.961	221.243	295.011	0.415	0.858	34.122
220	30	219.956	30.072	4	12	689.747	221.413	278.202	0.399	0.864	34.132
220	30	219.754	29.897	5	12	680.512	221.701	267.428	0.388	0.871	34.155
220	30	219.931	29.878	6	12	673.833	221.631	260.227	0.381	0.873	34.156
220	30	219.996	29.938	7	12	664.354	221.924	249.619	0.370	0.876	34.164
220	30	220.182	29.951	8	12	660.182	222.018	244.998	0.365	0.877	34.163
230	30	230.085	29.975	1	12	1660.764	207.375	1347.227	0.805	0.461	31.877
230	30	230.161	29.727	2	12	1421.486	213.172	1064.306	0.729	0.626	33.001
230	30	230.059	29.910	3	12	727.207	220.787	312.643	0.427	0.842	34.098
230	30	230.118	29.797	3	12	726.049	220.702	311.328	0.426	0.843	34.096
230	30	229.867	30.031	4	12	711.093	221.065	294.329	0.410	0.851	34.122
230	30	230.079	29.954	5	12	702.740	221.224	284.469	0.400	0.856	34.132
230	30	229.908	29.888	6	12	693.628	221.375	274.431	0.391	0.860	34.134
230	30	230.019	29.814	7	12	686.824	221.406	266.620	0.383	0.864	34.140
230	30	229.898	29.828	8	12	683.806	221.523	263.306	0.380	0.866	34.148
240	0	240.000	0.000	1	12	1573.121	206.836	1259.867	0.799	0.443	31.718
240	0	240.000	0.000	2	12	1456.108	212.521	1098.192	0.733	0.606	32.351
240	0	240.000	0.000	3	12	715.681	220.521	294.535	0.410	0.836	34.102
240	0	240.000	0.000	4	12	703.725	220.759	280.548	0.396	0.843	34.113
240	0	240.000	0.000	5	12	700.133	220.901	275.710	0.391	0.848	34.134
240	0	240.000	0.000	6	12	690.286	221.062	264.706	0.380	0.852	34.133
240	0	240.000	0.000	7	12	682.133	221.175	255.382	0.370	0.857	34.143
240	0	240.000	0.000	8	12	679.162	221.268	252.036	0.367	0.858	34.148
240	30	239.861	29.819	1	12	1710.515	206.692	1397.565	0.812	0.443	31.753
240	30	240.101	29.910	1	12	1721.435	206.808	1408.271	0.813	0.443	31.753
240	30	240.124	29.797	2	12	1595.657	211.602	1244.192	0.760	0.583	32.805
240	30	240.061	29.855	2	12	1590.424	211.578	1239.299	0.760	0.581	32.808
240	30	0.000	0.000	3	0	348.033	191.444	156.588	0.432	0.000	34.573
240	30	240.170	29.290	3	2	582.835	197.097	346.821	0.599	0.162	34.242
240	30	239.992	29.749	3	4	619.415	202.552	340.343	0.556	0.319	34.179
240	30	239.985	29.971	3	6	674.154	207.679	354.577	0.532	0.466	34.152
240	30	240.128	29.867	3	8	716.721	212.235	360.696	0.507	0.599	34.114
240	30	239.959	29.914	3	10	739.312	216.457	350.373	0.475	0.719	34.092
240	30	240.138	29.849	3	12	747.831	220.163	328.791	0.437	0.828	34.073
240	30	239.881	29.910	3	12	748.633	220.275	329.636	0.438	0.828	34.077
240	30	240.028	29.865	3	12	747.159	220.194	328.237	0.437	0.828	34.076
240	30	239.789	29.887	3	14	752.696	223.495	307.755	0.404	0.924	34.066
240	30	239.913	30.019	3	16	756.640	225.700	294.188	0.382	0.987	34.083
240	30	240.097	29.906	3	18	756.666	226.172	290.517	0.376	0.999	34.089
240	30	239.928	30.021	3	20	757.598	226.228	291.442	0.377	1.000	34.086
240	30	240.237	29.972	4	12	732.854	220.615	310.922	0.421	0.838	34.102
240	30	240.073	29.889	5	12	721.655	220.785	298.336	0.410	0.844	34.109
240	30	240.028	29.883	6	12	713.573	220.991	289.247	0.401	0.847	34.119
240	30	239.950	29.990	7	12	705.638	221.008	280.220	0.393	0.852	34.126
240	30	240.067	29.876	8	12	702.080	221.088	276.101	0.389	0.853	34.128
240	60	240.218	59.625	1	12	1888.609	206.764	1576.221	0.825	0.440	31.785
240	60	239.946	59.076	2	12	1685.675	211.370	1336.367	0.774	0.575	32.833
240	60	239.878	60.089	3	12	839.206	219.312	427.276	0.505	0.803	33.975
240	60	240.060	59.763	4	12	795.113	220.048	377.594	0.471	0.823	34.039
240	60	240.291	59.581	5	12	776.925	220.268	357.179	0.455	0.830	34.058
240	60	240.157	59.891	6	12	770.929	220.470	350.031	0.449	0.835	34.065
240	60	240.090	59.738	7	12	754.267	220.627	332.193	0.435	0.839	34.075
240	60	240.135	59.990	8	12	750.301	220.626	327.432	0.431	0.842	34.081
240	90	241.477	87.904	1	12	2026.845	206.443	1715.639	0.833	0.434	31.761
240	90	241.105	88.321	2	12	1803.859	210.937	1457.782	0.790	0.561	32.684
240	90	241.241	88.457	3	12	1006.133	217.790	605.838	0.592	0.757	33.746
240	90	240.973	88.706	4	12	872.754	219.247	460.470	0.523	0.801	33.947
240	90	240.870	88.251	5	12	855.202	219.593	440.481	0.510	0.810	33.969
240	90	241.286	88.336	6	12	836.868	219.724	420.144	0.496	0.817	33.985
240	90	240.574	88.169	7	12	820.783	220.010	403.122	0.486	0.822	33.999
240	90	241.336	88.104	8	12	816.438	220.104	397.154	0.481	0.825	34.009
240	120	246.585	112.507	1	12	2134.484	206.015	1824.809	0.841	0.421	31.718
240	120	246.671	112.709	2	12	1940.779	210.277	1597.328	0.804	0.541	32.527
240	120	246.707	112.890	3	12	1250.239	215.652	862.334	0.673	0.699	33.363

Input μ_{Ts}	Input σ_{Ts}	Result μ_{Ts}	Result σ_{Ts}	Lim_s	$\#T$	T_{tot}	T_{mov}	T_{dly}	RD	TR	DTR
240	120	247.403	113.077	4	12	987.411	218.079	579.469	0.581	0.768	33.810
240	120	246.995	112.799	5	12	937.524	218.722	524.981	0.554	0.785	33.867
240	120	246.181	112.064	6	12	912.474	219.087	498.078	0.540	0.793	33.896
240	120	247.018	112.711	7	12	896.179	219.208	479.753	0.529	0.798	33.908
240	120	247.296	112.522	8	12	888.135	219.256	470.732	0.524	0.801	33.923
250	30	249.880	29.906	1	12	1781.585	206.152	1468.909	0.820	0.426	31.640
250	30	250.017	29.908	2	12	1734.486	210.355	1387.437	0.781	0.547	32.666
250	30	249.911	29.911	3	12	769.540	219.659	346.764	0.449	0.813	34.055
250	30	249.965	29.887	3	12	771.288	219.729	348.304	0.450	0.813	34.056
250	30	250.058	29.820	4	12	753.625	220.040	327.484	0.432	0.824	34.080
250	30	249.915	29.972	5	12	745.646	220.350	317.366	0.423	0.832	34.087
250	30	250.065	29.838	6	12	737.632	220.459	308.536	0.415	0.834	34.097
250	30	250.034	29.977	7	12	728.127	220.603	297.572	0.405	0.840	34.107
250	30	250.132	29.841	8	12	723.028	220.730	291.752	0.400	0.842	34.110
260	30	259.962	29.776	1	12	1828.624	205.673	1516.153	0.825	0.411	31.518
260	30	260.115	29.800	2	12	1896.987	209.251	1554.065	0.800	0.514	32.475
260	30	259.804	29.805	3	12	792.083	219.160	365.646	0.461	0.798	34.031
260	30	260.020	29.982	3	12	792.603	219.197	366.057	0.461	0.797	34.027
260	30	260.068	29.949	4	12	774.368	219.546	343.718	0.442	0.812	34.062
260	30	259.952	29.898	5	12	765.702	219.938	333.005	0.433	0.818	34.069
260	30	260.031	29.880	6	12	756.037	219.971	322.504	0.424	0.821	34.076
260	30	260.074	30.031	7	12	748.762	220.278	313.171	0.415	0.828	34.088
260	30	260.078	29.902	8	12	743.550	220.256	307.164	0.410	0.831	34.092
270	30	269.957	29.902	1	12	1896.242	205.173	1584.063	0.831	0.396	31.399
270	30	270.138	29.955	2	12	2040.100	208.261	1700.541	0.815	0.486	32.264
270	30	0.000	0.000	3	0	347.904	191.371	156.533	0.432	0.000	34.574
270	30	269.881	29.582	3	2	612.853	196.753	374.284	0.616	0.155	34.210
270	30	270.104	29.757	3	4	656.335	202.003	371.968	0.576	0.305	34.134
270	30	269.942	29.787	3	6	733.777	206.508	408.121	0.564	0.441	34.099
270	30	269.859	29.967	3	8	786.955	210.966	424.598	0.546	0.561	34.042
270	30	270.018	29.857	3	10	799.562	214.879	402.571	0.507	0.674	34.023
270	30	270.025	29.840	3	12	814.900	218.631	385.655	0.473	0.780	33.998
270	30	270.095	29.942	3	12	816.917	218.365	388.135	0.475	0.779	34.005
270	30	270.084	29.897	3	12	813.334	218.618	383.907	0.472	0.781	34.008
270	30	269.981	29.833	3	14	824.794	222.002	365.796	0.440	0.878	33.980
270	30	270.062	29.975	3	16	831.914	224.881	346.661	0.411	0.964	34.013
270	30	270.161	30.088	3	18	838.258	226.096	343.103	0.402	0.996	34.013
270	30	270.044	29.915	3	20	836.271	226.167	340.095	0.400	1.000	34.014
270	30	269.900	29.968	4	12	791.599	219.142	356.777	0.450	0.799	34.036
270	30	269.876	29.930	5	12	782.394	219.550	345.277	0.440	0.806	34.052
270	30	269.948	29.934	6	12	779.747	219.485	342.019	0.437	0.808	34.060
270	30	270.068	29.952	7	12	766.288	219.784	326.083	0.423	0.816	34.067
270	30	269.992	30.000	8	12	763.260	219.999	322.255	0.419	0.819	34.073
280	30	280.069	29.737	1	12	1939.123	204.814	1627.166	0.835	0.383	31.294
280	30	280.170	29.747	2	12	2175.510	207.375	1838.550	0.827	0.463	32.144
280	30	279.982	29.950	3	12	849.588	217.826	419.284	0.494	0.759	33.960
280	30	280.044	29.970	3	12	848.974	217.783	418.526	0.493	0.759	33.961
280	30	280.033	29.801	4	12	814.247	218.663	375.638	0.461	0.785	34.014
280	30	280.143	29.851	5	12	803.600	219.064	362.276	0.450	0.793	34.029
280	30	280.227	30.075	6	12	796.610	219.124	354.422	0.443	0.796	34.040
280	30	280.044	29.906	7	12	785.686	219.342	341.006	0.432	0.805	34.050
280	30	280.169	29.961	8	12	783.613	219.419	338.050	0.429	0.807	34.049
290	30	289.753	29.690	1	12	2006.612	204.402	1694.877	0.841	0.370	31.184
290	30	289.937	29.804	2	12	2316.163	206.566	1981.873	0.838	0.441	32.016
290	30	289.928	29.940	3	12	916.703	216.712	489.137	0.532	0.727	33.870
290	30	289.904	29.914	3	12	913.414	216.764	485.708	0.531	0.728	33.860
290	30	290.126	29.881	4	12	834.270	218.354	391.801	0.470	0.772	33.998
290	30	290.128	29.980	5	12	822.569	218.585	377.352	0.458	0.781	34.009
290	30	290.049	29.768	6	12	814.430	218.622	368.368	0.451	0.784	34.022
290	30	289.973	29.889	7	12	806.603	218.998	358.147	0.442	0.791	34.023
290	30	290.042	29.903	8	12	804.755	219.180	354.588	0.439	0.796	34.033
300	0	300.000	0.000	1	12	1936.260	203.994	1624.639	0.838	0.359	31.066
300	0	300.000	0.000	2	12	2380.496	206.200	2046.986	0.841	0.424	31.301
300	0	300.000	0.000	3	12	845.453	217.190	405.793	0.483	0.742	33.945
300	0	300.000	0.000	4	12	824.255	217.935	377.320	0.460	0.763	33.999
300	0	300.000	0.000	5	12	820.026	218.268	370.120	0.452	0.772	34.018
300	0	300.000	0.000	6	12	817.422	218.356	366.581	0.449	0.775	34.016
300	0	300.000	0.000	7	12	804.054	218.688	350.441	0.435	0.783	34.034
300	0	300.000	0.000	8	12	801.062	218.845	345.670	0.431	0.788	34.037
300	30	300.267	29.755	1	12	2062.763	203.831	1751.229	0.846	0.359	31.080
300	30	299.876	29.859	1	12	2045.389	203.861	1734.013	0.845	0.359	31.076
300	30	300.041	29.730	2	12	2431.317	206.174	2098.372	0.846	0.423	31.847

Input μ_{Ts}	Input σ_{Ts}	Result μ_{Ts}	Result σ_{Ts}	Lim_s	$\#T$	T_{tot}	T_{mov}	T_{dly}	RD	TR	DTR
300	30	300.222	29.708	2	12	2423.431	206.045	2090.245	0.845	0.424	31.856
300	30	0.000	0.000	3	0	347.446	191.570	155.876	0.431	0.000	34.574
300	30	299.875	29.557	3	2	642.038	196.490	401.096	0.632	0.148	34.181
300	30	300.135	29.483	3	4	700.484	201.455	411.461	0.598	0.292	34.079
300	30	299.935	29.799	3	6	805.824	205.669	476.978	0.600	0.411	34.007
300	30	300.061	29.766	3	8	889.183	209.159	526.502	0.598	0.512	33.896
300	30	299.975	29.826	3	10	960.911	212.269	568.151	0.591	0.602	33.738
300	30	300.122	29.975	3	12	1068.728	214.800	652.400	0.602	0.672	33.534
300	30	300.072	29.909	3	12	1058.178	214.779	641.276	0.598	0.674	33.495
300	30	299.995	29.894	3	12	1058.607	214.815	641.447	0.598	0.675	33.514
300	30	300.024	29.977	3	14	1178.427	216.678	742.981	0.615	0.729	33.287
300	30	300.024	29.915	3	16	1288.655	218.473	836.744	0.626	0.778	33.099
300	30	300.018	29.838	3	18	1392.667	220.114	926.036	0.635	0.822	32.916
300	30	299.987	29.951	3	20	1470.933	221.393	991.549	0.639	0.860	32.754
300	30	299.979	29.841	4	12	856.002	217.866	410.519	0.481	0.759	33.976
300	30	300.006	29.801	5	12	841.291	218.162	392.267	0.467	0.770	33.996
300	30	300.054	29.997	6	12	833.373	218.390	383.058	0.459	0.773	33.997
300	30	299.922	29.902	7	12	824.633	218.606	372.119	0.450	0.780	34.012
300	30	299.869	30.085	8	12	818.332	218.797	363.947	0.443	0.786	34.022
300	60	299.848	59.556	1	12	2205.052	203.961	1893.957	0.853	0.357	31.143
300	60	300.197	59.609	2	12	2473.209	206.242	2139.044	0.847	0.426	31.930
300	60	300.263	59.766	3	12	1369.409	212.707	973.293	0.693	0.611	33.067
300	60	299.960	59.583	4	12	918.072	217.324	477.396	0.520	0.745	33.912
300	60	300.074	59.790	5	12	901.107	217.667	456.219	0.505	0.757	33.927
300	60	300.053	59.540	6	12	885.503	218.047	438.589	0.494	0.763	33.941
300	60	299.877	59.809	7	12	875.166	218.166	426.226	0.485	0.770	33.955
300	60	299.880	59.807	8	12	868.173	218.531	416.963	0.478	0.776	33.964
300	90	300.041	89.134	1	12	2385.720	203.662	2075.635	0.861	0.355	31.149
300	90	300.678	89.062	2	12	2520.155	206.373	2185.048	0.850	0.429	31.906
300	90	300.522	88.994	3	12	1589.993	211.435	1206.094	0.739	0.574	32.847
300	90	300.027	89.485	4	12	1015.902	216.537	583.299	0.571	0.720	33.783
300	90	300.378	89.904	5	12	970.346	217.111	530.692	0.545	0.741	33.849
300	90	300.229	89.133	6	12	954.363	217.509	511.988	0.534	0.749	33.860
300	90	300.215	89.556	7	12	941.260	217.793	496.569	0.524	0.756	33.872
300	90	300.143	89.634	8	12	927.436	217.908	481.085	0.516	0.761	33.887
300	120	302.041	116.751	1	12	2538.462	203.464	2229.182	0.866	0.350	31.102
300	120	302.637	115.978	2	12	2589.619	206.202	2254.044	0.853	0.428	31.811
300	120	302.102	116.989	3	12	1783.838	210.494	1408.072	0.769	0.548	32.640
300	120	301.878	116.642	4	12	1166.760	215.270	745.138	0.632	0.684	33.571
300	120	301.952	116.517	5	12	1054.265	216.391	620.642	0.586	0.720	33.744
300	120	302.289	117.212	6	12	1029.934	216.924	592.233	0.572	0.730	33.765
300	120	302.641	116.947	7	12	1012.461	217.080	572.131	0.561	0.738	33.787
300	120	302.781	116.795	8	12	1001.106	217.389	558.178	0.553	0.745	33.792
330	30	330.047	29.857	1	12	2238.939	202.819	1927.936	0.858	0.328	30.772
330	30	329.916	29.689	2	12	2772.804	204.645	2443.364	0.864	0.378	31.388
330	30	330.128	29.898	3	12	1652.397	209.706	1267.865	0.744	0.530	32.492
330	30	329.972	29.895	4	12	914.979	216.400	461.118	0.507	0.720	33.909
330	30	329.972	29.871	5	12	897.702	216.871	438.585	0.490	0.734	33.937
330	30	329.880	29.854	6	12	893.370	217.052	432.561	0.485	0.739	33.940
330	30	329.919	29.959	7	12	884.961	217.261	422.215	0.477	0.744	33.953
330	30	330.002	29.956	8	12	877.879	217.656	411.365	0.468	0.754	33.962
360	30	359.876	29.797	1	12	2398.365	201.848	2087.801	0.868	0.302	30.469
360	30	360.287	29.854	1	12	2402.109	202.010	2091.401	0.868	0.302	30.475
360	30	360.082	29.891	2	12	3074.683	203.586	2746.207	0.877	0.347	31.042
360	30	360.380	29.812	2	12	3086.199	203.535	2758.184	0.877	0.345	31.008
360	30	359.829	29.929	3	12	2197.985	206.711	1832.819	0.808	0.440	31.819
360	30	359.945	29.715	4	12	973.461	215.037	514.549	0.533	0.678	33.859
360	30	360.127	29.822	5	12	959.653	215.725	491.526	0.515	0.701	33.871
360	30	360.205	29.808	6	12	953.315	215.984	482.415	0.509	0.708	33.876
360	30	359.907	29.730	7	12	941.126	216.211	469.391	0.500	0.710	33.889
360	30	360.106	30.013	8	12	935.158	216.492	458.122	0.491	0.724	33.903

2

Appendix 2 - Python Source Code

```python
import salabim as sim
import numpy as np

from Parameters import *
from Functions import createArrSchedule
from movingAgents.Flights import Flight

class ArrivalGenerator(sim.Component):
    def __init__(self, env):
        sim.Component.__init__(self, env=env)

        self.schedule = createArrSchedule()

    def process(self):
        while True:

            for i in range(len(self.schedule)):

                yield self.activate(at=self.schedule[i])

                x = np.random.random()
                if x <= aircraft_types_db.iloc[0].share:
                    aircraft_type = 0
                elif x <= aircraft_types_db.iloc[0].share+aircraft_types_db.iloc[1].share:
                    aircraft_type = 1
                else:
                    aircraft_type = 2

                route = arr_route
                towed = np.random.random() <= self.env.towRatio

                self.env.arrivals.append(
                    Flight(flight_id = self.env.flight_id,
                           route = route,
                           towed = towed,
                           type = "arrival",
                           vehicle_type = aircraft_type
                    )
                )
                self.env.flight_id += 1

            yield self.passivate()
```
Listing 2.1: ArrivalGenerator.py

```python
import salabim as sim
import numpy as np

from Parameters import *
from Functions import createDepSchedule
from movingAgents.Flights import Flight

class DepartureGenerator(sim.Component):
    def __init__(self, env):
        sim.Component.__init__(self, env=env)

        self.schedule = createDepSchedule()

    def process(self):
        while True:

            for i in range(len(self.schedule)):

                yield self.activate(at=self.schedule[i])

                x = np.random.random()
                if x <= aircraft_types_db.iloc[0].share:
                    aircraft_type = 0
                elif x <= aircraft_types_db.iloc[0].share+aircraft_types_db.iloc[1].share:
                    aircraft_type = 1
                else:
                    aircraft_type = 2

                if self.env.double_runway_entry:
                    route = np.random.choice(dep_routes)
                else:
                    route = dep_routes[0]

                towed = np.random.random() <= self.env.towRatio

                self.env.departures.append(
                    Flight(flight_id = self.env.flight_id,
                           route = route,
                           type = "departure",
                           towed = towed,
                           vehicle_type = aircraft_type
                    )
                )
                self.env.flight_id += 1

            yield self.passivate()
```

Listing 2.2: DepartureGenerator.py

```python
import salabim as sim

from Parameters import *

class modelEnvironment(sim.Environment):
    def __init__(self,
                 seed,
                 tow_ratio,
                 force_tow,
                 serial_decoupling,
                 n_tugs,
                 n_tugs_buffer_limit,
                 sigma_Tc,
```

```
                 sigma_Td,
                 couple_mu,
                 decouple_mu,
                 run_decouple_cap,
                 run_couple_cap,
                 double_runway_entry
                 ):


        sim.Environment.__init__(self, random_seed=seed)

        self.links = []
        self.nodes = []
        self.departures = []
        self.arrivals = []
        self.tugs = []

        self.node_queues = []
        self.link_queues = []
        self.service_queues = []

        self.flight_id = 0
        self.tug_id = 0

        self.towRatio = tow_ratio
        self.forceTow = force_tow
        self.serialDecoupling = serial_decoupling
        self.nTugs = n_tugs
        self.nTugsBufferLimit = n_tugs_buffer_limit
        self.double_runway_entry = double_runway_entry

        self.sigma_Tc = sigma_Tc
        self.sigma_Td = sigma_Td
        self.couple_mu = couple_mu
        self.decouple_mu = decouple_mu

        self.run_decouple_cap = run_decouple_cap
        self.run_couple_cap = run_couple_cap



    def setApronTugBuffer(self, queue):

        self.apron_tug_buffer = queue

    def setRunwayTugBuffer(self, queue):

        self.runway_tug_buffer = queue
```

Listing 2.3: Environment.py

```
import salabim as sim

from Parameters import *
from Functions import *

from movingAgents.Flights import Flight
from movingAgents.Tugs import Tug

class Node(sim.Component):
    def __init__(self, id, type, queue, capacity):
        sim.Component.__init__(self)
```

```

        self.type = type
        self.id = id
        self.my_queue = queue
        self.capacity = capacity

        if id == 0:
            self.my_buffer = self.env.apron_tug_buffer
        elif id == 8:
            self.my_buffer = self.env.runway_tug_buffer
        else:
            self.my_buffer = None

        # if id == 7:
        #     self.t_last_departure = 0

    def process(self):
        while True:

            ## Generals checks: The node can only deliver services if

            # 1. Someone is in the queue
            if len(self.my_queue) == 0:
                yield self.passivate()
                continue
            # 2. It has capacity to deliver service
            if len(self.env.service_queues[self.id]) >= self.capacity:
                yield self.passivate()
                continue

            ## Actions per node type:

            # 1. Node is intersection
            if self.type == "intersection":
                self.customer = self.my_queue.pop()

                if print_statements:
                    print(str(self.customer) + " is now in service at " + str(self.id))
                yield from self.serveIntersection()

            # 2. Node is the runway
            elif self.type == "runway":
                self.customer = self.my_queue.pop()

                if print_statements:
                    print(str(self.customer) + " is now in service at " + str(self.id))
                yield from self.serveRunway()

            # 3. Node is a couple station
            elif self.type == "couple":

                couple = False

                if self.id == 0:
                # if the customer does not need to be towed, we can always serve the customer
                    if not self.env.forceTow:
                        self.customer = self.my_queue.pop()

                        #a tow_truck is available, the customer can be served
                        if self.customer.shouldBeTowed and len(self.my_buffer) > 0 :
                            couple = True
```

```

                    if print_statements:
                        print(str(self.customer) + " is now in service at " + str(self.id))
                    self.serveCoupleParallel(couple)

                else:
                    if len(self.my_buffer) > 0:
                        self.customer = self.my_queue.pop()
                        if self.customer.shouldBeTowed:
                            couple = True

                        self.serveCoupleParallel(couple)

                    else:
                        customer_found = False
                        for index in range(len(self.my_queue)):

                            potential_customer = self.my_queue[index]

                            if type(potential_customer) == Tug:
                                raise Exception("This should not happen")

                            if not potential_customer.shouldBeTowed:
                                customer_found = True
                                self.customer = self.my_queue.pop(index)
                                break

                        if not customer_found:
                            yield self.passivate()
                            continue

                        else:
                            if print_statements:
                                print(str(self.customer) + " is now in service at " + str(self.id))
                            self.serveCoupleParallel(couple)
            else:

                if not self.checkForSerialCapacity():
                    yield self.passivate()
                    continue

                if not self.env.forceTow:
                    self.customer = self.my_queue.pop()

                    # a tow_truck is available, the customer can be served
                    if self.customer.shouldBeTowed and len(self.my_buffer) > 0:
                        couple = True

                    if print_statements:
                        print(str(self.customer) + " is now in service at " + str(self.id))
                    self.serveCoupleSerial(couple)

                else:

                    if self.my_queue.head().shouldBeTowed:
                        if len(self.my_buffer) > 0:

                            couple = True
```

```python
                                self.customer = self.my_queue.pop()

                                if print_statements:
                                    print(str(self.customer) + " is now in service at " + str(self.id))
                                self.serveCoupleSerial(couple)

                            else:
                                yield self.passivate()
                                continue

                        else:
                            self.customer = self.my_queue.pop()
                            couple = False

                            if print_statements:
                                print(str(self.customer) + " is now in service at " + str(self.id))
                            self.serveCoupleSerial(couple)

            # 4. Node is a decouple station
            elif self.type == "decouple":

                if self.id == 9:
                    if self.checkForParallelCapacity():
                        self.customer = self.my_queue.pop()

                        self.serveDecoupleParallel()

                    else:
                        yield self.passivate()
                        continue

                else:
                    #check capacity for serial decoupling
                    if self.checkForSerialCapacity():
                        self.customer = self.my_queue.pop()

                        self.serveDecoupleSerial()

                    else:
                        yield self.passivate()
                        continue


                # if print_statements:
                #     print(str(self.customer) + " is now in service at " + str(self.id))

    def checkForSerialCapacity(self):

        if len(self.env.service_queues[self.id]) == 0:
            return True

        else:
            tail_flight = self.env.service_queues[self.id].tail()
            if tail_flight.decouplePosition < self.capacity - 1:
                return True

            else:
                return False

```

```
    def checkForParallelCapacity(self):
        if len(self.env.service_queues[self.id]) <= self.capacity:
            return True
        else:
            return False

    def serveIntersection(self):
        self.customer.enter(self.env.service_queues[self.id])
        self.customer.my_monitor.tally(str(self.id) + ".C")

        if print_statements:
            print(str(self.customer) + " is crossing intersection " + str(self))

        travel_time = calcIntersectionTime(self.id, self.customer)

        self.customer.hold(travel_time)
        yield self.hold(max(travel_time, taxi_separation))

    def serveRunway(self):

        if type(self.customer) == Tug:
            raise Exception("This should not happen")

        if print_statements:
            print(str(self.customer) + " is using the runway")

        if len(self.my_queue) > 0:

            service_time = calcRunwaySeparationTime(self.customer, self.my_queue[0])

            self.customer.enter(self.env.service_queues[self.id])
            self.customer.my_monitor.tally(str(self.id) + ".C")

            if self.customer.type == 'departure':
                self.customer.activate()
            else:
                self.customer.hold(service_time)

            yield self.hold(service_time)

        else:

            self.last_departure_t = self.env.t()

            self.customer.enter(self.env.service_queues[self.id])
            self.customer.my_monitor.tally(str(self.id) + ".C")

            if self.customer.type == 'departure':
                self.customer.activate()
            else:
                self.customer.hold(calcIntersectionTime(7, self.customer))

            yield self.passivate()

            t = self.env.t()
            min_sep = calcRunwaySeparationTime(self.customer, self.my_queue[0])

            diff = min_sep - (t - self.last_departure_t)
            if diff > 0:
                yield self.hold(diff)
```

```
    def serveCoupleParallel(self, couple):

        if self.env.forceTow and self.customer.shouldBeTowed and (len(self.my_buffer) == 0 or not couple):
            raise Exception("This should not happen")

        if couple and len(self.my_buffer) == 0:
            raise Exception("This should not happen")

        if type(self.customer) == Tug:
            raise Exception("This should not happen")

        if couple:
            self.customer.isTowed = True
            self.customer.my_tug = self.my_buffer.pop()
            self.customer.my_tug.state = 1

            if print_statements:
                    print(str(self.customer) + " gets paired with " + str(self.customer.my_tug))

        service_time = calcCoupleDecoupleStationTime(self, self.customer, True)

        self.customer.enter(self.env.service_queues[self.id])
        self.customer.my_monitor.tally(str(self.id) + ".C")
        self.customer.tCouple = service_time

        self.customer.activate(delay=service_time)

    def serveCoupleSerial(self, couple):

        if self.env.forceTow and self.customer.shouldBeTowed and (len(self.my_buffer) == 0 or not couple):
            raise Exception("This should not happen")

        if couple and len(self.my_buffer) == 0:
            raise Exception("This should not happen")

        if type(self.customer) == Tug:
            raise Exception("This should not happen")

        if couple:
            self.customer.isTowed = True
            self.customer.my_tug = self.my_buffer.pop()
            self.customer.my_tug.state = 1

            if print_statements:
                    print(str(self.customer) + " gets paired with " + str(self.customer.my_tug))

        if len(self.env.service_queues[self.id]) == 0:
            t_predecessor = None
            self.customer.couplePosition = 0
        else:
            predecessor = self.env.service_queues[self.id].tail()
            t_predecessor = predecessor.scheduled_time()
            self.customer.decouplePosition = predecessor.decouplePosition + 1


        service_time = calcCoupleDecoupleStationTime(self, self.customer, True)

```

```
        self.customer.enter(self.env.service_queues[self.id])
        self.customer.my_monitor.tally(str(self.id) + ".C")
        self.customer.tCouple = service_time

        if t_predecessor == None:
            self.customer.activate(delay = service_time)
        elif t_predecessor >= self.env.t() + service_time:
            self.customer.activate(at=t_predecessor)
        else:
            self.customer.activate(delay=service_time)

    def serveDecoupleSerial(self):

        if type(self.customer) == Tug:
            raise Exception("This should not happen")

        #Determine values for next step
        service_time = float(calcCoupleDecoupleStationTime(self, self.customer, False))

        if len(self.env.service_queues[self.id]) == 0:
            t_predecessor = None
            self.customer.decouplePosition = 0

        else:
            predecessor = self.env.service_queues[self.id].tail()
            t_predecessor = predecessor.scheduled_time()
            self.customer.decouplePosition = predecessor.decouplePosition + 1


        #Log info for next step
        self.customer.enter(self.env.service_queues[self.id])
        self.customer.my_monitor.tally(str(self.id) + ".C")
        self.customer.tDecouple = service_time

        ## Activate passive agents at the appropriate time
        # Activate flight
        if t_predecessor == None:
            self.customer.activate(delay = service_time)
        elif t_predecessor >= self.env.t() + service_time:
            self.customer.activate(at=t_predecessor)
        else:
            self.customer.activate(delay=service_time)
        # Activate tug
        if self.customer.isTowed:
            uncoupled_tug = self.customer.my_tug
            uncoupled_tug.state = 2
            if self.id == 4:
                uncoupled_tug.route = [4, 2, 8]
            elif self.id == 6:
                uncoupled_tug.route = [6, 5, 2, 8]
            elif self.id == 9:
                uncoupled_tug.route = [9, 0]

            uncoupled_tug.progress = 0
            uncoupled_tug.activate(delay=service_time)

            if print_statements:
                print(str(uncoupled_tug) + " is decoupled from " + str(self.customer))

    def serveDecoupleParallel(self):

```

```
        if type(self.customer) == Tug:
            raise Exception("This should not happen")

        #Determine values for next step
        service_time = float(calcCoupleDecoupleStationTime(self, self.customer, False))

        #Log info for next step
        self.customer.enter(self.env.service_queues[self.id])
        self.customer.my_monitor.tally(str(self.id) + ".C")
        self.customer.tDecouple = service_time

        ## Activate passive agents at the appropriate time
        # Activate flight
        self.customer.activate(delay=service_time)

        # Activate tug
        if self.customer.isTowed:
            uncoupled_tug = self.customer.my_tug
            uncoupled_tug.state = 2
            if self.id == 4:
                uncoupled_tug.route = [4, 2, 8]
            elif self.id == 6:
                uncoupled_tug.route = [6, 5, 2, 8]
            elif self.id == 9:
                uncoupled_tug.route = [9, 0]

            uncoupled_tug.progress = 0
            uncoupled_tug.activate(delay=service_time)

            if print_statements:
                print(str(uncoupled_tug) + " is decoupled from " + str(self.customer))
```

Listing 2.4: Nodes.py

```
import salabim as sim

from Parameters import *
from Functions import *

class Flight(sim.Component):
    def __init__(self, flight_id, route, type, towed, vehicle_type):
        sim.Component.__init__(self)

        self.flight_id = flight_id
        self.route = route
        self.shouldBeTowed = towed
        self.type = type
        self.vehicle_type = vehicle_type

        self.progress = 0
        self.finished = False
        self.isTowed = False
        self.decouplePosition = 0
        self.couplePosition = 0

        self.my_monitor = sim.Monitor(name="stage_reached")
        self.tCouple = 0
        self.tDecouple = 0


    def animation_objects(self, id):

```

```
        '''
        the way the component is determined by the id, specified in AnimateQueue
        'text' means just the name
        any other value represents the colour
        '''

        if self.type == 'departure':
            text = "d."+str(self.flight_id)
            color = 'blue'
        else:
            text = 'a.'+str(self.flight_id)
            color = 'red'

        if self.isTowed:
            color = 'green'

        if id == 'text':
            ao0 = sim.AnimateText(text=self.name(), textcolor='fg', text_anchor='nw')
            return 0, 16, ao0


        elif id == "old":

            ao0 = sim.AnimateRectangle((-10, -10, 10, 10),
                text=self.name(), fillcolor=id, textcolor='white', arg=self)
            return 30, 0, ao0

        else:

            ao0 = sim.AnimateRectangle((0, 0, 60, 20),text=self.name(), fillcolor=color, text_anchor='c')
            return 0, 28, ao0

    def TravelToNextNode(self, last_node, next_node):

        travel_time = calcTravelTime(last_node_id=last_node.id, next_node_id=next_node.id, agent=self)

        if print_statements:
            print(str(self) + ' is travelling to node ' + str(next_node))


        self.my_monitor.tally(str(next_node.id)+".A")
        self.my_link = self.env.link_queues[getLinkId(last_node.id, next_node.id)]
        self.enter(self.my_link)
        yield self.hold(travel_time)

    def process(self):
        #keep going as long as route is not finished
        while not self.finished:

            if self.progress == 0:
                first_node = self.env.nodes[self.route[0]]
                self.enter(first_node.my_queue)
                if first_node.ispassive():
                    first_node.activate()

                yield self.passivate()

            previous_node = self.env.nodes[self.route[self.progress]]
            self.leave(self.env.service_queues[previous_node.id])
```

```python
            if previous_node.ispassive() and previous_node.id != 7:
                previous_node.activate()

            if print_statements:
                print(str(self) + " has finished service at " + str(previous_node))

            if self.progress == len(self.route)-1:
                self.finished = True
                break

            self.progress += 1

            current_node = self.env.nodes[self.route[self.progress]]
            current_queue = current_node.my_queue

            yield from self.TravelToNextNode(last_node = previous_node, next_node = current_node)
            self.leave(self.my_link)

            self.enter(current_queue)
            self.my_monitor.tally(str(current_node.id)+".B")
            if print_statements:
                print(str(self) + " is waiting at " + str(current_node))

            if current_node.ispassive():
                current_node.activate()

            yield self.passivate()

        self.my_monitor.tally('T')
        if print_statements:
            print(str(self) + ' = finished')
```

Listing 2.5: Flights.py

```python
from Functions import *


class Tug(sim.Component):
    def __init__(self, tug_id, vehicle_type):
        sim.Component.__init__(self)

        self.state = 0
        self.progress = 0
        self.route = []
        self.tug_id = tug_id
        self.vehicle_type = vehicle_type


        self.my_monitor = sim.Monitor(name=str(self)+' monitor')


    def animation_objects(self, id):

        '''
        the way the component is determined by the id, specified in AnimateQueue
        'text' means just the name
        any other value represents the colour
        '''

        if id == 'text':
            ao0 = sim.AnimateText(text=self.name(), textcolor='fg', text_anchor='nw')
```

```
            return 0, 16, ao0


        elif id == "old":

            ao0 = sim.AnimateRectangle((-10, -10, 10, 10),
                text=self.name(), fillcolor=id, textcolor='white', arg=self)
            return 30, 0, ao0

        elif id == 'buffer':

            ao0 = sim.AnimateRectangle((0,  0, 40, 20),text=self.name(), fillcolor='orange', text_anchor='c')
            return 45, 0, ao0

        else:

            ao0 = sim.AnimateRectangle((0, 0, 40, 20),text=self.name(), fillcolor='orange', text_anchor='c')
            return 0, 28, ao0

    def TravelToNextNode(self, last_node, next_node):

        travel_time = calcTravelTime(last_node_id=last_node.id, next_node_id=next_node.id, agent=self)

        if print_statements:
            print(str(self) + ' is travelling to node ' + str(next_node))


        # self.my_monitor.tally(next_node.id)
        self.my_link = self.env.link_queues[getLinkId(last_node.id, next_node.id)]
        self.enter(self.my_link)
        yield self.hold(travel_time)

    def process(self):
        while True:

            while self.state == 0 or self.state == 1:
                yield self.passivate()

            ####
            # tug is moving on its own
            if print_statements:
                print(str(self) + " is now on its own")

            previous_node = self.env.nodes[self.route[self.progress]]
            if self.progress != 0:
                self.leave(self.env.service_queues[previous_node.id])
                if previous_node.ispassive() and previous_node.id != 7:
                    previous_node.activate()

            if print_statements:
                print(str(self) + " has finished service at " + str(previous_node))

            self.progress += 1

            current_node = self.env.nodes[self.route[self.progress]]
            current_queue = current_node.my_queue

            yield from self.TravelToNextNode(previous_node, current_node)
```

```
            self.leave(self.my_link)

            if self.progress == len(self.route)-1:
                if current_node == self.env.nodes[0]:
                    self.enter(self.env.apron_tug_buffer)
                    if print_statements:
                        print(str(self) + " has entered the apron buffer")
                    # exert limit on buffer size ( = 15)
                    if len(self.env.apron_tug_buffer) > self.env.nTugsBufferLimit:
                        tug_to_move = self.env.apron_tug_buffer.pop()
                        tug_to_move.state = 3
                        tug_to_move.route = [0, 1, 2, 8]
                        tug_to_move.progress = 0
                        tug_to_move.activate()

                elif current_node == self.env.nodes[8]:
                    self.enter(self.env.runway_tug_buffer)
                    if print_statements:
                        print(str(self) + " has entered the runway buffer")
                    if len(self.env.runway_tug_buffer) > self.env.nTugsBufferLimit:
                        tug_to_move = self.env.runway_tug_buffer.pop()
                        tug_to_move.state = 3
                        tug_to_move.route = [8,0]
                        tug_to_move.progress = 0
                        tug_to_move.activate()
                else:
                    raise Exception("This should not happen")

                self.state = 0

                if current_node.ispassive():
                    current_node.activate()

                yield self.passivate()

            else:

                self.enter(current_queue)
                if current_node.ispassive():
                    current_node.activate()

                yield self.passivate()
```

Listing 2.6: Tugs.py

```
import pandas as pd
import numpy as np

# simulation params

animation_speed = 150
animate_sim = True
synced = True
trace_sim = False
print_statements = False
n_monte_carlo = 400

# input variables

force_tow = False
serial_decoupling = True
double_runway_entry = False
```

```

tow_ratio = 1
n_tugs = 12
n_tugs_buffer_limit = n_tugs-2
n_deps = 300
n_arrs = 300

sigma_Tc_delta_array = []
sigma_Td_delta_array = []
mu_Tc_delta_array = []
mu_Td_delta_array = []

run_decouple_cap_array = np.arange(1, 7, 1)
run_couple_cap_array = np.arange(1, 4, 1)

taxi_separation = 20

DepRate = (60/35) * 60  # seconds per departure
DepStd = 0.6 * DepRate

ArrRate = (60/35) * 60
ArrStd = 0.8 * ArrRate # seconds per arrival

separation_runway_cross = 30

runway_width = 80 #m

taxi_way_width = 50 #m




# towing_states_flight = {
#     0: 'not paired (passive)',
#     1: 'paired, waiting for pickup (passive)',
#     2: 'paired, coupling (passive)',
#     3: 'paired, towing in progress (active)',
#     4: 'paired, decoupling in progress (passive)',
#     5: 'ready for take-off'
# }

# tug_states = {
#     0: 'in buffer, passive',
#     1: 'paired, traveling to towards runway, passive',
#     2: 'unpaired, active, travelling to next buffer',
#     3: 'unpaired, active travelling from buffer to buffer without flight for balance'
# }

dep_routes = [
    [0, 1, 2, 4, 7],
    [0, 1, 3, 5, 6, 7]
]

arr_route = [8, 9]

node_db = pd.read_excel('./network_params2.xlsx', index_col=0, header=0, sheet_name='nodes')
link_db = pd.read_excel('./network_params2.xlsx', index_col=0, header=0, sheet_name='links')
aircraft_types_db = pd.read_excel('./network_params2.xlsx', index_col=0, header=0, sheet_name=
    'aircraft_types')
runway_separation = pd.read_excel('./network_params2.xlsx', header=None, sheet_name='
    runway_separation').to_numpy()
```

```python

distance_dict = {}
link_id_dict = {}
link_id = 0

for link in link_db.iterrows():
    distance_dict[link[1]['origin_node'], link[1]['dest_node']] = link[1]['distance']
    link_id_dict[link[1]['origin_node'], link[1]['dest_node']] = link_id
    link_id += 1

# service times

# mu_Tc = 2 * 60  # [s]
# sigma_Tc = 0.5 * 60  # [s]
#
# mu_Td = 2 * 60  # [s]
# sigma_Td = 1 * 60  # [s]

# mu_dep = 300
# mu_arr = 300

# v_coupled = 15 #[m/s]
# v_aircraft_nom = 10  # [m/s]
# v_tow_nom = 7 #[m/s]

# v_corner_factor = 0.8

# T_takeOff_mu = 60
# T_takeOff_sigma = 5

# sim_length = 8 * 60 * 60  # [s] (8 hours)
#
# daily_flights = 600
#
```

Listing 2.7: Parameters.py

```python
#################################################################
# IMPORT PACKAGES
#################################################################

from Functions import *

from movingAgents.Tugs import Tug

from nonMovingAgents.Nodes import Node
from nonMovingAgents.DepartureGenerator import DepartureGenerator
from nonMovingAgents.ArrivalGenerator import ArrivalGenerator
from nonMovingAgents.Environment import modelEnvironment

def createEnvironment(seed,
                      tow_ratio,
                      force_tow,
                      serial_decoupling,
                      n_tugs,
                      n_tugs_buffer_limit,
                      double_runway_entry,
                      sigma_Tc,
                      sigma_Td,
                      couple_mu,
                      decouple_mu,
                      run_decouple_cap,
```

```
                         run_couple_cap
                         ):

    env = modelEnvironment(seed = seed,
                               tow_ratio = tow_ratio,
                               force_tow = force_tow,
                               serial_decoupling = serial_decoupling,
                               n_tugs = n_tugs,
                               n_tugs_buffer_limit = n_tugs_buffer_limit,
                               sigma_Tc = sigma_Tc,
                               sigma_Td = sigma_Td,
                               couple_mu = couple_mu,
                               decouple_mu = decouple_mu,
                               run_decouple_cap = run_decouple_cap,
                               run_couple_cap = run_couple_cap,
                               double_runway_entry = double_runway_entry
                               )

    env.speed(animation_speed)
    env.animate(animate_sim)
    env.animation_parameters(width=1700, height=1200, x0=0, y0=0, x1=1700)
    env.synced(synced)
    env.trace(trace_sim)

    env.setApronTugBuffer(queue=sim.Queue("apron_buffer"))
    env.setRunwayTugBuffer(queue=sim.Queue("runway_buffer"))

    # CREATE trucks/nodes/queues and flights

    #tow-trucks
    for i in range(int(n_tugs/2)):
        env.apron_tug_buffer.append(Tug(tug_id = env.tug_id, vehicle_type = 3))
        env.tug_id += 1
        env.runway_tug_buffer.append(Tug(tug_id = env.tug_id, vehicle_type = 3))
        env.tug_id += 1

    # nodes & queues
    for i in range(len(node_db)):
        env.node_queues.append((sim.Queue()))
        if i == 4 or i == 6:
            env.nodes.append(
                Node(id = i, type = node_db.iloc[i].type, queue = env.node_queues[i], capacity = env.run_decouple_cap))
        elif i == 8:
            env.nodes.append(
                Node(id=i, type=node_db.iloc[i].type, queue=env.node_queues[i], capacity=env.run_couple_cap))
        else:
            env.nodes.append(
                Node(id=i, type=node_db.iloc[i].type, queue=env.node_queues[i], capacity=node_db.iloc[i].capacity))

        env.service_queues.append(sim.Queue(name='service monitor '+str(i)))

    for i in range(len(link_db)):
        env.link_queues.append((sim.Queue(name='link '+ str(i))))


    # fligths
    DepartureGenerator(env=env)
    ArrivalGenerator(env=env)
```

```python

    ####################################################################
    # Animate monitors
    ####################################################################

    queue_monitors = []
    link_monitors = []
    service_monitors = []

    sim.AnimateQueue(queue=env.apron_tug_buffer, x=100, y=725, title='apron_buffer', direction='e', id='buffer', titleoffsetx=0)
    sim.AnimateQueue(queue=env.runway_tug_buffer, x=420, y=1075, title='runway_buffer', direction='e', id='buffer', titleoffsetx=0)


    queue_monitors.append(sim.AnimateQueue(queue=env.node_queues[0],
                                           x=100, y=600, title='0-queue', id='custom', direction='s', titleoffsetx=0))
    service_monitors.append(sim.AnimateQueue(queue=env.service_queues[0],
                                             x=180, y=600, title='0-service', id='custom', direction='s', titleoffsetx=0))


    link_monitors.append(sim.AnimateQueue(queue=env.link_queues[0],
                                          x=300, y=650, title='1-link', id='custom', direction='s', titleoffsetx=0))

    queue_monitors.append(sim.AnimateQueue(queue=env.node_queues[1],
                                           x=420, y=600, title='1-queue', id='custom', direction='s', titleoffsetx=0))
    service_monitors.append(sim.AnimateQueue(queue=env.service_queues[1],
                                             x=500, y=600, title='1-service', id='custom', direction='s', titleoffsetx=0))

    link_monitors.append(sim.AnimateQueue(queue=env.link_queues[1],
                                          x=620, y=650, title='2-link', id='custom', direction='s', titleoffsetx=0))

    queue_monitors.append(sim.AnimateQueue(queue=env.node_queues[2],
                                           x=740, y=600, title='2-queue', id='custom', direction='s', titleoffsetx=0))
    service_monitors.append(sim.AnimateQueue(queue=env.service_queues[2],
                                             x=820, y=600, title='2-service', id='custom', direction='s', titleoffsetx=0))


    link_monitors.append(sim.AnimateQueue(queue=env.link_queues[3],
                                          x=940, y=650, title='4-link', id='custom', direction='s', titleoffsetx=0))
    link_monitors.append(sim.AnimateQueue(queue=env.link_queues[7],
                                          x=940, y=550, title='4\"-link', id='custom', direction='s', titleoffsetx=0))

    queue_monitors.append(sim.AnimateQueue(queue=env.node_queues[4],
                                           x=1060, y=600, title='4-queue', id='custom', direction='s', titleoffsetx=0))
    service_monitors.append(sim.AnimateQueue(queue=env.service_queues[4],
                                             x=1140, y=600, title='4-service', id='custom', direction='s', titleoffsetx=0))

```

```
    link_monitors.append(sim.AnimateQueue(queue=env.link_queues[6],
                                    x=1260, y=650, title='7-link', id='custom', direction='s', titleoffsetx=0))

    queue_monitors.append(sim.AnimateQueue(queue=env.node_queues[7],
                                    x=1380, y=600, title='7-queue', id='custom', direction='s', titleoffsetx=0))
    service_monitors.append(sim.AnimateQueue(queue=env.service_queues[7],
                                        x=1460, y=600, title='7-service', id='custom', direction='s', titleoffsetx=0))


    link_monitors.append(sim.AnimateQueue(queue=env.link_queues[2],
                                    x=470, y=450, title='3-link', id='custom', direction='s', titleoffsetx=0))

    queue_monitors.append(sim.AnimateQueue(queue=env.node_queues[3],
                                    x=420, y=300, title='3-queue', id='custom', direction='s', titleoffsetx=0))
    service_monitors.append(sim.AnimateQueue(queue=env.service_queues[3],
                                        x=500, y=300, title='3-service', id='custom', direction='s', titleoffsetx=0))


    link_monitors.append(sim.AnimateQueue(queue=env.link_queues[4],
                                    x=620, y=350, title='5-link', id='custom', direction='s', titleoffsetx=0))

    queue_monitors.append(sim.AnimateQueue(queue=env.node_queues[5],
                                    x=740, y=300, title='5-queue', id='custom', direction='s', titleoffsetx=0))
    service_monitors.append(sim.AnimateQueue(queue=env.service_queues[5],
                                        x=820, y=300, title='5-service', id='custom', direction='s', titleoffsetx=0))

    link_monitors.append(sim.AnimateQueue(queue=env.link_queues[5],
                                    x=940, y=350, title='6-link', id='custom', direction='s', titleoffsetx=0))
    link_monitors.append(sim.AnimateQueue(queue=env.link_queues[9],
                                    x=940, y=250, title='6\"-link', id='custom', direction='s', titleoffsetx=0))


    queue_monitors.append(sim.AnimateQueue(queue=env.node_queues[6],
                                    x=1060, y=300, title='6-queue', id='custom', direction='s', titleoffsetx=0))
    service_monitors.append(sim.AnimateQueue(queue=env.service_queues[6],
                                        x=1140, y=300, title='6-service', id='custom', direction='s', titleoffsetx=0))


    link_monitors.append(sim.AnimateQueue(queue=env.link_queues[8],
                                    x=1260, y=350, title='7B-link', id='custom', direction='s', titleoffsetx=0))



    link_monitors.append(sim.AnimateQueue(queue=env.link_queues[11],
                                    x=790, y=450, title='52-link', id='custom', direction='s', titleoffsetx=0))

```

```
    link_monitors.append(sim.AnimateQueue(queue=env.link_queues[12],
                                    x=790, y=890, title='28-link', id='custom', direction='s', titleoffsetx=0))

    queue_monitors.append(sim.AnimateQueue(queue=env.node_queues[8],
                                    x=500, y=1000, title='8-queue', id='custom', direction='s', titleoffsetx=0))
    service_monitors.append(sim.AnimateQueue(queue=env.service_queues[8],
                                        x=420, y=1000, title='8-service', id='custom', direction='s', titleoffsetx=0))

    link_monitors.append(sim.AnimateQueue(queue=env.link_queues[13],
                                    x=300, y=1050, title='89-link', id='custom', direction='s', titleoffsetx=0))

    queue_monitors.append(sim.AnimateQueue(queue=env.node_queues[9],
                                    x=180, y=1000, title='9-queue', id='custom', direction='s', titleoffsetx=0))
    service_monitors.append(sim.AnimateQueue(queue=env.service_queues[9],
                                        x=100, y=1000, title='9-service', id='custom', direction='s', titleoffsetx=0))


    link_monitors.append(sim.AnimateQueue(queue=env.link_queues[14],
                                    x=100, y=890, title='09-link', id='custom', direction='s', titleoffsetx=0))

    link_monitors.append(sim.AnimateQueue(queue=env.link_queues[10],
                                    x=1260, y=890, title='78-link', id='custom', direction='s', titleoffsetx=0))


    stage_db = {}

    for node in env.nodes:
        stage_db[node.id] = "taxiway " + str(node.type) + ' ' + str(node.id)
        stage_db[node.id+0.5] = 'queue ' + str(node.type) + ' ' + str(node.id)
        stage_db[node.id+0.9] = 'service ' + str(node.type) + ' ' + str(node.id)


    return env


## run environment
```

Listing 2.8: simulate.py

```
from Functions import *
from Parameters import *

import matplotlib.pyplot as plt
from matplotlib.ticker import MaxNLocator
import pickle

from simulate import createEnvironment

np.warnings.filterwarnings('ignore', category=np.VisibleDeprecationWarning)

service_mu_array = np.arange(10,310,10)
mask = (service_mu_array % 30 != 0)
service_mu_array_special = service_mu_array[mask]
service_sigma_array = np.arange(10,310,10)
```

```
n_tugs_array = np.arange(0,22,2)
tow_ratio_array = np.arange(0.1, 1.1, 0.1)
run_cap_array = np.arange(1,2,1)

## Run

# tow_rate_results = {}
#
# dep_dfs = []
# arr_dfs = []

INPUT_n_tugs = []
INPUT_avg_Tservice = []
INPUT_std_Tservice = []

# output lists
OUTPUT_DEP_Ttot = []
OUTPUT_DEP_Tmov = []
OUTPUT_DEP_TserMu = []
OUTPUT_DEP_TserSigma = []
OUTPUT_DEP_Tdly = []
OUTPUT_DEP_TR = []
OUTPUT_DEP_RD = []

OUTPUT_DRT = []

OUTPUT_ARR_Ttot = []
OUTPUT_ARR_Tmov = []
OUTPUT_ARR_TserMu = []
OUTPUT_ARR_TserSigma = []
OUTPUT_ARR_Tdly = []
OUTPUT_ARR_RD = []
OUTPUT_ARR_TR = []


for run_cap in run_cap_array:

    DEP_Ttot_simAvgs = []
    DEP_Tmov_simAvgs = []
    DEP_Tser_simAvgs = []
    DEP_Tser_simStds = []

    DEP_Tdly_simAvgs = []
    DEP_TR_simAvgs = []
    DEP_RD_simAvgs = []

    DRT_simAvgs = []

    ARR_Ttot_simAvgs = []
    ARR_Tmov_simAvgs = []
    ARR_Tser_simAvgs = []
    ARR_Tser_simStds = []

    ARR_Tdly_simAvgs = []
    ARR_TR_simAvgs = []
    ARR_RD_simAvgs = []

    sigma = 30
    mu = 300
```

```
    for i in range(n_monte_carlo):

        print("Running iteration " + str(i+1) + " of " + str(n_monte_carlo) + " for run_cap = " + str(run_cap))

        env = createEnvironment(seed = "*",
                                tow_ratio = 1,
                                force_tow = False,
                                serial_decoupling = True,
                                n_tugs = 12,
                                n_tugs_buffer_limit = n_tugs-2 if n_tugs > 2 else 1,
                                double_runway_entry = False,
                                sigma_Tc = 30,
                                sigma_Td = 30,
                                couple_mu = 360,
                                decouple_mu = 360,
                                run_decouple_cap = 1,
                                run_couple_cap = 1
                                )

        env.run()

        ## departures
        DEP_Ttot_flights = []
        DEP_Tmov_flights = []
        DEP_Tser_flights = []
        DEP_Tdly_flights = []
        DEP_RD_flights = []


        for departure in env.departures:

            #process data monitor
            monitor_df = departure.my_monitor.as_dataframe()
            monitor_df['stage_duration'] = abs(monitor_df['t'].diff(periods=-1))

            T_take_off = monitor_df.iloc[-1].t
            T_taxi_out = departure.creation_time()

            #calculate desired values

            Ttot = T_take_off-T_taxi_out
            Tser = departure.tDecouple
            Tmov = calc_nominal_travel_time(departure)

            Tdly = Ttot - Tmov - Tser
            if Tdly < -0.1:
                raise Exception("this should not happen")
            elif Tdly < 0:
                Tdly = 0

            RD = Tdly / Ttot

            #append to data
            DEP_Ttot_flights.append(Ttot)
            DEP_Tmov_flights.append(Tmov)
            if Tser > 0:
                DEP_Tser_flights.append(Tser)
            DEP_Tdly_flights.append(Tdly)
            DEP_RD_flights.append(RD)

```

```
        output_tow_rate = len([departure for departure in env.departures if departure.isTowed
    ])/len(env.departures)
        DRT = env.nodes[7].my_queue.departure_rate()*3600

        DEP_Ttot_simAvgs.append(np.mean(DEP_Ttot_flights))
        DEP_Tmov_simAvgs.append(np.mean(DEP_Tmov_flights))
        DEP_Tser_simAvgs.append(np.mean(DEP_Tser_flights) if len(DEP_Tser_flights) > 0 else 0)
        DEP_Tser_simStds.append(np.std(DEP_Tser_flights) if len(DEP_Tser_flights) > 0 else 0)
        DEP_Tdly_simAvgs.append(np.mean(DEP_Tdly_flights))
        DEP_RD_simAvgs.append(np.mean(DEP_RD_flights))
        DEP_TR_simAvgs.append(output_tow_rate)

        DRT_simAvgs.append(DRT)


        ## arrivals
        ARR_Ttot_flights = []
        ARR_Tmov_flights = []
        ARR_Tser_flights = []
        ARR_Tdly_flights = []
        ARR_RD_flights = []

        for arrival in env.arrivals:
            # process data monitor
            monitor_df = arrival.my_monitor.as_dataframe()
            monitor_df['stage_duration'] = abs(monitor_df['t'].diff(periods=-1))

            T_take_off = monitor_df.iloc[-1].t
            T_taxi_out = arrival.creation_time()

            # calculate desired values

            Ttot = T_take_off - T_taxi_out
            Tser = arrival.tCouple
            Tmov = calc_nominal_travel_time(arrival)

            Tdly = Ttot - Tmov - Tser
            if Tdly < -0.1:
                raise Exception("this should not happen")
            elif Tdly < 0:
                Tdly = 0

            RD = Tdly / Ttot

            # append to data
            ARR_Ttot_flights.append(Ttot)
            ARR_Tmov_flights.append(Tmov)
            if Tser > 0:
                ARR_Tser_flights.append(Tser)
            ARR_Tdly_flights.append(Tdly)
            ARR_RD_flights.append(RD)

        output_tow_rate = len([arrival for arrival in env.arrivals if arrival.isTowed]) / len(
    env.arrivals)

        ARR_Ttot_simAvgs.append(np.mean(ARR_Ttot_flights))
        ARR_Tmov_simAvgs.append(np.mean(ARR_Tmov_flights))
        ARR_Tser_simAvgs.append(np.mean(ARR_Tser_flights) if len(ARR_Tser_flights) > 0 else 0)
        ARR_Tser_simStds.append(np.std(ARR_Tser_flights) if len(ARR_Tser_flights) > 0 else 0)
        ARR_Tdly_simAvgs.append(np.mean(ARR_Tdly_flights))
        ARR_RD_simAvgs.append(np.mean(ARR_RD_flights))
```

```
            ARR_TR_simAvgs.append(output_tow_rate)


        OUTPUT_DEP_Ttot.append(np.mean(DEP_Ttot_simAvgs))
        OUTPUT_DEP_Tmov.append(np.mean(DEP_Tmov_simAvgs))
        OUTPUT_DEP_TserMu.append(np.mean(DEP_Tser_simAvgs))
        OUTPUT_DEP_TserSigma.append(np.mean(DEP_Tser_simStds))
        OUTPUT_DEP_Tdly.append(np.mean(DEP_Tdly_simAvgs))
        OUTPUT_DEP_RD.append(np.mean(DEP_RD_simAvgs))
        OUTPUT_DEP_TR.append(np.mean(DEP_TR_simAvgs))
        OUTPUT_DRT.append(np.mean(DRT_simAvgs))

        OUTPUT_ARR_Ttot.append(np.mean(ARR_Ttot_simAvgs))
        OUTPUT_ARR_Tmov.append(np.mean(ARR_Tmov_simAvgs))
        OUTPUT_ARR_TserMu.append(np.mean(ARR_Tser_simAvgs))
        OUTPUT_ARR_TserSigma.append(np.mean(ARR_Tser_simStds))
        OUTPUT_ARR_Tdly.append(np.mean(ARR_Tdly_simAvgs))
        OUTPUT_ARR_RD.append(np.mean(ARR_RD_simAvgs))
        OUTPUT_ARR_TR.append(np.mean(ARR_TR_simAvgs))


        output_array_length = len(OUTPUT_DEP_Ttot)

    dep_dict = {
        # "n": n_monte_carlo,
        # "\#D": n_deps,
        # "D_f": DepRate,
        # "D_sigma": DepStd,
        # "\#A": n_arrs,
        # "A_f": ArrRate,
        # "A_sigma": ArrStd,
        # 'V_N0' : aircraft_types_db.iloc[0].V_n,
        # 'V_N1': aircraft_types_db.iloc[1].V_n,
        # 'V_N2': aircraft_types_db.iloc[2].V_n,
        # 'V_N3': aircraft_types_db.iloc[3].V_n,
        # 'V_T0': aircraft_types_db.iloc[0].V_t,
        # 'V_T1': aircraft_types_db.iloc[1].V_t,
        # 'V_T2': aircraft_types_db.iloc[2].V_t,
        # "CF": aircraft_types_db.iloc[0].corner_factor,
        # "TS": taxi_separation,

        'input_mu_Ts': np.full((output_array_length),env.couple_mu),
        # 'input_mu_Ts': service_mu_array,

        'input_sigma_Ts': np.full((output_array_length),env.sigma_Tc),
        # 'input_sigma_Ts': service_sigma_array,

        "result_mu_Ts": OUTPUT_DEP_TserMu,
        "result_sigma_Ts": OUTPUT_DEP_TserSigma,

        'Lim_s': np.full((output_array_length), env.run_couple_cap),
        # 'Lim_s': run_cap_array,

        '\#T': np.full((output_array_length), env.nTugs),
        # '\#T': n_tugs_array,

        'T_tot' :  OUTPUT_DEP_Ttot,
        'T_mov' : OUTPUT_DEP_Tmov,
        'T_dly' : OUTPUT_DEP_Tdly,
        'RD' : OUTPUT_DEP_RD,
        'TR' : OUTPUT_DEP_TR,
```

```
    'DTR': OUTPUT_DRT
}

arr_dict = {

    'input_mu_Ts': np.full((output_array_length), env.couple_mu),
    # 'input_mu_Ts': service_mu_array,

    'input_sigma_Ts': np.full((output_array_length),env.sigma_Tc),
    # 'input_sigma_Ts': service_sigma_array,

    "result_mu_Ts": OUTPUT_ARR_TserMu,
    "result_sigma_Ts": OUTPUT_ARR_TserSigma,

    # 'Lim_s': np.full((output_array_length), env.run_couple_cap),
    'Lim_s': run_cap_array,

    '\#T': np.full((output_array_length), env.nTugs),
    # '\#T': n_tugs_array,

    'T_tot' :  OUTPUT_ARR_Ttot,
    'T_mov' : OUTPUT_ARR_Tmov,
    'T_dly' : OUTPUT_ARR_Tdly,
    'RD' : OUTPUT_ARR_RD,
    'TR' : OUTPUT_ARR_TR,
}

dep_df = pd.DataFrame(dep_dict)
arr_df = pd.DataFrame(arr_dict)

excel_path = './outputDF/output.xlsx'

append_to_excel(dep_df, excel_path, "DEP")
append_to_excel(arr_df, excel_path, "ARR")



        # fig, (ax1, ax2) = plt.subplots(2, sharex=True, gridspec_kw={'height_ratios': [3,2]},
     figsize=(6.4,6.4))
        # ax1.plot(dep_df['x'], dep_df['y'], 'tab:blue')
        # ax1.set_title('Departures')
        # ax2.plot(dep_df['x'], dep_df['z'], 'tab:green')
        #
        # ax1.set(xlabel='Mean decouple service time $\\mu_{Td}$ [s]')
        # ax2.set(xlabel='Mean decouple service time $\\mu_{Td}$ [s]')
        # ax1.set(ylabel='Taxi delay [s]')
        # ax2.set(ylabel='Tow-rate [1]')
        #
        # # Hide x labels and tick labels for top plots and y ticks for right plots.
        # ax1.label_outer()
        #
        # plt.show()
        #
        # fig, (ax1, ax2) = plt.subplots(2, sharex=True, gridspec_kw={'height_ratios': [3,2]},
     figsize=(6.4,6.4))
        # ax1.plot(dep_df['x'], dep_df['y'], 'tab:blue')
        # ax1.set_title('Departures')
        # ax2.plot(dep_df['x'], dep_df['z'], 'tab:green')
        #
        # ax1.set(xlabel='Mean decouple service time $\\mu_{Td}$ [s]')
        # ax2.set(xlabel='Mean decouple service time $\\mu_{Td}$ [s]')
```

```
        # ax1.set(ylabel=’Taxi delay [s]’)
        # ax2.set(ylabel=’Tow-rate [1]’)
        #
        # # Hide x labels and tick labels for top plots and y ticks for right plots.
        # ax1.label_outer()
        #
        # plt.show()
        #
        # fig, axs = plt.subplots(2, 2, sharex=’col’, sharey=’row’, gridspec_kw={’height_ratios’: [3,2]}, figsize=(12,6.4))
        # axs[0, 0].plot(dep_df[’x’], dep_df[’y’], ’tab:blue’)
        # axs[0, 0].set_title(’Departures’)
        # axs[0, 1].plot(arr_df[’x’], arr_df[’y’], ’tab:orange’)
        # axs[0, 1].set_title(’Arrivals’)
        # axs[1, 0].plot(dep_df[’x’], dep_df[’z’], ’tab:green’)
        # axs[1, 1].plot(arr_df[’x’], arr_df[’z’], ’tab:green’)
        #
        # for ax in axs[:,0]:
        #     ax.set(xlabel=’Mean decouple service time $\\mu_{Td}$ [s]’)
        # for ax in axs[:,1]:
        #     ax.set(xlabel=’Mean couple service time $\\mu_{Tc}$ [s]’)
        # for ax in axs[0,:]:
        #     ax.set(ylabel=’Relative taxi delay ($T_{idle}/T_{total}$) [1]’)
        # for ax in axs[1,:]:
        #     ax.set(ylabel=’Tow-rate [1]’)
        #
        # # Hide x labels and tick labels for top plots and y ticks for right plots.
        # for ax in axs.flat:
        #     ax.label_outer()
        #     ax.grid()
        #
        # plt.show()

# with open(’./outputBinaryFiles/sigmaVS.txt’, ’wb’) as exportfile:
#     pickle.dump((dep_dfs, arr_dfs, service_mu_array, n_tugs_array, run_cap_array, tow_ratio_array, n_monte_carlo, force_tow, double_runway_entry, n_tugs, n_tugs_buffer_limit, DepRate, ArrRate), exportfile)
#
#
#
# fig, axs = plt.subplots(1, 2, sharey=True, figsize=(12,4.8))
# axs[0].plot(n_tugs_array, dep_df[’y’], ’tab:blue’, label=’Relative taxi delay ($T_{dly}/T_{tot}$)’)
# axs[0].plot(n_tugs_array, dep_df[’z’], ’tab:green’, label=’Towing-rate’)
# axs[0].xaxis.set_major_locator(MaxNLocator(integer=True))
# axs[0].set_title(’Departures’)
# axs[1].plot(n_tugs_array, arr_df[’y’], ’tab:blue’, label=’Relative taxi delay ($T_{dly}/T_{tot}$)’)
# axs[1].plot(n_tugs_array, arr_df[’z’], ’tab:green’, label=’Towing-rate’)
# axs[1].xaxis.set_major_locator(MaxNLocator(integer=True))
# axs[1].set_title(’Arrivals’)
# axs[0].set(xlabel=’Number of tow-trucks #T’)
# axs[1].set(xlabel=’Number of tow-trucks #T’)
# axs[1].legend(loc=’upper right’, bbox_to_anchor=(0.55, 1), fancybox=True, shadow=True)
# axs[0].grid()
# axs[1].grid()
# plt.show()
```

Listing 2.9: run_results.py