Master of Science Thesis

Investigating the impact of an optimally configured self-service luggage check-in system on airport terminal performance

M. Torsij

Master of Science Thesis

Investigating the impact of an optimally configured self-service luggage check-in system on airport terminal performance

by

M. Torsij

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thurday November 30th, 2023.

Student number:	4533399	
Project duration:	October 2022 – November 2023	
Thesis committee:	Dr. B. F. Santos	TU Delft, Chair
	Ir. O. Stroosma	TU Delft, Examiner
	Dr. O. A. Sharpanskykh	TU Delft, Supervisor

TU Delft

Acknowledgements

This master thesis marks the end of my studies here at the Delft University of Technology and is the result of 12 months of work. In the almost seven and a half years in Delft, I have learned more than I could have ever imagined. I am thankful for all the people that helped facilitate this.

First of all I would like to thank my supervisor Dr. Alexei Sharpanskykh. Even tough this project did not always go as planned, you guided and motivated me to keep going. I think this is very valuable lesson which made me a better researcher. Having the opportunity to discuss issues with peers who are also working on their thesis is very valuable. Therefore I would like to thank Misha and Antonio for our biweekly meetings. I would also like to express my gratitude towards Benjamin. Even after finishing his thesis he still provided me with his valuable insights.

Aside from the academic support, I also want to thank my friends. The many cups of coffee on the in the Fellowship were maybe not always increasing productivity, but were very welcome nonetheless.

Furthermore I want to express my gratitude to Karin. She was always there to support me in all the things I do. Even in the busy exam period she found the time to meticulously proofread this for me which was of great help.

At last, I am thankful for my family always supporting me and being there if I need them. Being given the opportunity to study here and really do what I want to do, is something I will always be grateful for.

M. Torsij
Delft, November 2023

Contents

List of Figures

List of Tables

List of Abbreviations

AATOM	Agent-Based Airport Terminal Operations Model
BT	Bag tag kiosk
BT SR	Bag tag kiosk service rate
CI	Check-in counter
CoV	Coefficient of Variation
DL	Demand level
DO	Drop-off kiosk
DO SR	Drop-off kiosk service rate
GUI	Graphical User Interface
IATA	The International Air Transport Association
RTHA	Rotterdam The Hague Airport

Introduction

The capacity of airport terminals approaches their limits and long queues are becoming more common. Therefore, airports are looking for ways to increase the capacity. Airport terminals consist of sequential facilities making it important to streamline the individual facilities while closely monitoring the performance of the airport terminal as a whole. The check-in facility for passengers and luggage is the first facility passenger encounters. As checking in online is getting more common, a large increase in terms of capacity can be made by optimizing the self-service luggage check-in system before security. On the other hand at the very end of airport terminal process, a phenomenon is occurring increasingly often. More passengers are bringing carry-on luggage due to not wanting to wait at check-in and usually extra charges for bringing checked luggage. As a result, carry-on luggage has to be checked in at the gate due to limited space onboard the aircraft. This is a time consuming process, which could have large consequences such as delaying of flights. Therefore by automating this check-in process of carry-on luggage at the gate, a boarding can decreased thus increasing performance.

Rotterdam The Hague Airport (RTHA) wants to replace their current self-service luggage check-in system to create more capacity. RTHA wants to achieve this by implementing a new self-service luggage check-in system before security and at the gate. Before implementing this new system however, the configuration needs to be determined. By means of simulation, different configurations can be tested to investigate the impact of this system on the airport terminal performance.

This thesis aims to investigate the impact of an optimally configured self-service luggage check-in system before security and at the gate on airport terminal performance. This is done by combining an agent-based airport terminal simulator with a Simheuristic simulation optimization framework.

The structure of this thesis report is as follows. In part I, the scientific paper is presented. Part II contains the literature study that supports this research. Part III is the supporting work which consists of multiple appendices. In this part, the development of the agent-based model is elaborated upon, the development of the optimization software is explained more in depth and additional results are presented.

I

Scientific Paper

Investigating the impact of an optimally configured self-service luggage check-in system on airport terminal performance

M. Torsij,*

Delft University of Technology, Delft, The Netherlands

Abstract

Rotterdam the Hague Airport (RTHA) is planning to implement a new self-service luggage check-in system which has a different concept of operations compared to the current self-service system for luggage check-in. To efficiently develop, evaluate and implement this new system, a simulation optimization model is required to test different system configurations. The simulation optimization model is required to have a high level of detail to accurately represent the airport terminal process. An agent-based model will be used as the basis as that approach provides the necessary level of detail. This model is then combined with the metaheuristic tabu search algorithm to optimize the new self-service luggage check-in system. To address challenges related to the large required computational resources to perform simulation optimization with a high level of detail, the Simheuristic framework will be used. Combining an agent-based model with the Simheuristic framework is a novel approach to simulation optimization. This approach requires the development of a deterministic version of the airport terminal model. It was found that implementing an optimally configured self-service luggage check-in system can improve the performance of the check-in process while not compromising the performance of the security checkpoint. Additionally, the Simheuristic framework is capable of reducing the computational resources required.

Keywords: Airport check-in, self-service luggage check-in, agent-based modeling, simheuristics, simulation optimization

1 Introduction

As of 2022, the airport services market size was valued at $97.87 billion and is projected to grow to $290.23 billion by 2029 [4]. These figures illustrate the immense size and growing potential of the aviation industry. However, this growth also poses challenges in terms of capacity of airports. A simple solution would be for airports to keep expanding and add more terminals. More often than not, this is not possible, resulting in the need to create more capacity without expanding the facility. Development in self-service technology, planning and operations helps to address the challenges of creating more capacity.

Rotterdam The Hague Airport (RTHA) is the third largest airport of the Netherlands and serves roughly 2 million passenger every year [9]. RTHA is considering to implement a new self-service luggage check-in system and new gate kiosks to increase overall airport terminal performance. This system has separate bag tag, drop-off and gate kiosks. The first two are used for checking in luggage before security and the latter is used for bag tagging carry-on luggage at the gate. The luggage that needs to be checked in before security is referred to as 'checked luggage' for the remainder of this paper. This self-service system requires less actions from the passenger and has a higher service rate [2]. The research objective is to investigate the impact of an optimally configured self-service luggage check-in system on airport terminal performance.

In order to evaluate the performance of an airport terminal without physical tests, simulations will be performed. The agent-based modeling and simulation paradigm was found to be most suitable as it provides a high level of detail, representing all the necessary interactions and dynamics, and is able to model individual passengers [11]. Therefore, passenger flows can be visualized and individual passenger interactions and behavior can be modeled and analyzed. A model which is available and fulfills the requirements is AATOM, which stands for Agent-Based Airport Terminal Operations Model [6]. This model is chosen to serve as the basis for this research.

To determine the optimal configuration for the self-service kiosks using AATOM, a simulation optimization approach will be used. This approach combines optimization techniques with simulation modeling to evaluate solutions. Simulation optimization is often applied to discrete event simulations and systems of stochastic

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

nonlinear and/or differential equations [1]. This technique combines a stochastic simulation model and an algorithm to explore the solution space and find an optimal solution. The main downside of using a highly detailed simulation model such as AATOM for simulation optimization, is the need for multiple simulation runs. Juan et al. [8] proposes the Simheuristic framework, which decreases this computational time and can potentially be used in combination with an agent-based model. This framework relies on a simplified or deterministic version of the simulation model to identify potential good solutions. The full stochastic simulation model is then used to evaluate these good solutions. In remainder of this paper, these good solutions are referred to as 'elite solutions'. Combining an agent-based model with the Simheuristic framework is a novel approach to a simulation optimization problem. Therefore, the feasibility of this approach will also be investigated in this research.

This paper is structured in the following way. First, the related work will be discussed in section 2. Second, a description of the case study will be given in section 3. Next, the methodology used to achieve the research objective can be found in section 4 and section 5. After that, the experimental setup will be discussed in section 6 and results of the experiments will be shown in section 7. A discussion will be given in section 8 and finally, the conclusions following from the experiments will presented in section 9.

2 Related Work

In this section, we review the state of the art for the topics: self-service kiosks, airport terminal models, Simheuristics and metaheuristic optimization algorithms.

2.1 Next generation self-service kiosks

It is estimated that in the near future 33%-50% of the check-in counters will have self-service capabilities [5]. Additionally, The International Air Transport Association (IATA) introduced the Fast Travel Program in 2007. This program focuses on self-check-in, self-rebooking and self-bag tagging.

The current generation of self-service luggage check-in kiosks are designed to make the luggage check-in process completely automatic. To ensure complete automation, extra features such as weighing the luggage and extra luggage payment options are often available. This is a result of airports designing the self service kiosks to maximize perceived usefulness instead of perceived ease of use [10]. The kiosks that will be used in this research simplify the process by having no extra features besides providing the bag tag and drop-off location. The procedure of the self-service luggage check-in system consists of the following steps:

1. Going to bag tag kiosk
2. Queuing and waiting until a kiosk is free
3. Scanning the boarding pass
4. Waiting for the bag tag to be printed and attach the bag tag
5. Going to the drop-off kiosk
6. Queuing and waiting until a kiosk is free
7. Scanning the boarding pass and placing luggage on belt

Checking in carry-on luggage at the gate is also a growing problem. Bagchain[2], a manufacturer of self-service kiosks, estimates that manually checking in carry-on luggage at the gate takes between 60 and 90 seconds. This time can be decreased using a kiosk while at the same time freeing up the gate operator to continue to board passengers. Important to note is that these bag tag kiosks are movable and can be moved strategically between gates. The procedure for these kiosks is defined as follows:

1. Going to the gate operator to board
2. Gate operator indicates that carry-on luggage must be checked in
3. Going to the gate kiosk
4. Queuing and waiting until a kiosk is free
5. Scanning the boarding pass
6. Waiting for the bag tag to be printed and attach the bag tag
7. Placing the carry-on luggage at the designated area

2.2 Agent-Based Airport Terminal Operations Model (AATOM)

As mentioned in section 1, the modeling technique which will be used to model the airport terminal with self-service kiosks is agent-based modeling. The AATOM simulator, developed by Janssen et al. [6] deploys this technique. This is an open-source airport terminal simulator with readily available and calibrated airport terminal components. AATOM follows an activity based architecture, which allows for analysis per activity. This model can therefore be used to implement the self-service luggage check-in system and evaluate the performance.

The architecture of AATOM is structured in three different layers: the strategic layer, the tactical layer and the operational layer [7]. A graphical representation of the architecture can be found in Figure 1. The strategic layer is the first layer and consists of the goal, belief and reasoning module. The tactical layer handles the actuation of the activities. It consists of a lower level belief module, which is connected to the belief module in the strategic layer, the interpretation module, activity module and navigation module. The final layer is the operational layer which interacts with the environment and other agents. This layer therefore consists of the perception and the actuation module.

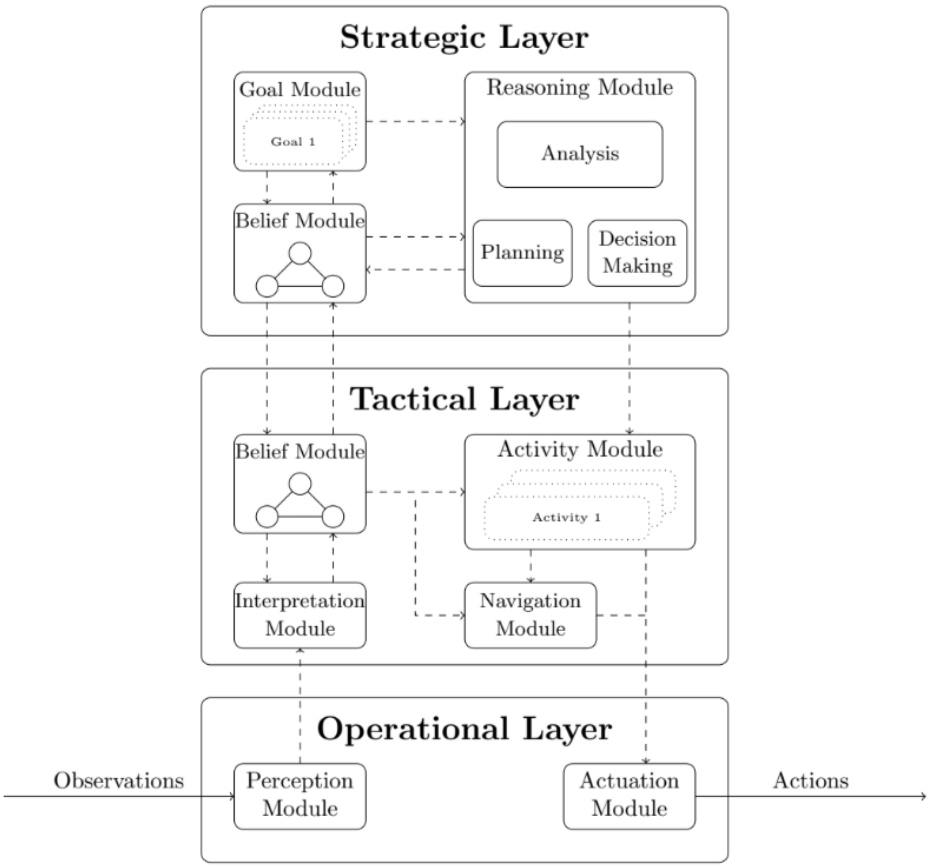


Figure 1: Schematic representation of the AATOM architecture[7]

All agent-based models have three main elements: agents, environment and interactions. The passengers and operators are the agents in AATOM. The environment is the terminal and all its physical objects, such as the check-in counters. The interactions are defined as agent to agent interactions (passenger - check-in operator for example) or agent to environment interaction (passenger - luggage belt for example).

2.3 Simulation optimization

The usage of an agent-based model with a high degree of stochasticity, such as AATOM, requires a different optimization approach than a mathematical model for example. When finding an optimal solution for a mathematical model, an objective function can often be evaluated quickly. Additionally, derivative information could be available which allows for efficient exploration and exploitation of the solution space. For models with a high degree stochasticity, derivative information may not be available or hard to estimate [1]. Therefore, a derivative information free optimization procedure most often referred to as simulation optimization can be used. In simulation optimization, an algorithm is used to explore the solution space and a simulation model is

used to evaluate the solution proposed by the algorithm.

For small solution spaces, evaluating every solution and ranking the solutions based on the fitness is the preferred approach as a true optimal solution is then found. For larger solution spaces this is too computationally intensive thus requiring an efficient method to explore the solution space. Different approaches to exploring the solution space are: model-based methods, metaheuristics and response surface methodology [1]. A metaheuristic algorithm was deemed to be most suitable for this research. The increased simplicity and scalalability of metaheuristics outweighed the decrease in efficiency compared to the other two methods. Many algorithms fall under metaheuristics such as: iterated local search, tabu search and genetic algorithms [1].

In order to determine which metaheuristic is preferred for this research, different characteristics of the algorithms were examined in a trade-off. These characteristics include: adaptability, scalability, efficiency and simplicity. This trade-off can be found in supporting work Appendix 2. Tabu search was deemed to be the best fit for this research as it is relatively simple while still offering control in terms of the exploration and exploitation of solutions.

Tabu search is essentially a modified form of neighborhood search where a form of adaptive memory is added. The adaptive memory of tabu search is referred to as the tabu list. This list tracks recently visited solutions preventing these solutions to be visited again [1]. Through the tabu list, tabu search offers control in terms of exploration and exploitation of solutions.

The tabu search procedure starts by evaluating the initial solution, which becomes the best solution s. The iterative process is started by finding the neighborhood of the best solution s. The neighborhood consists of solutions that are close to the best solution s. The neighborhood solutions are then evaluated after which the best solution s', which is not part of the tabu list, is chosen from the neighborhood. If the chosen solution s' is better than best solution s, solution s' becomes the new best solution s. Additionally, best solution s will also be added to the tabu list. In the case that best neighborhood solution s' is not better than best solution s, a local optimum is found. A new best solution s is then generated to continue the optimization procedure which is referred to as the perturbation of the best solution. Perturbing a solution can be done in various ways such as choosing a random solution. The optimization procedure continues until a stop condition is met, which often is a maximum number of iterations. Psuedocode of the tabu search logic can be found in algorithm 1.

Algorithm 1 Tabu search

```
1:  Choose initial solution in solution space
2:  Best solution s ← initial solution
3:  Tabu list ← ∅
4:  while not stopping criterion do
5:      Find neighborhood solutions N(s)
6:      Select best solution s' ∈ N(s) not in Tabu list
7:      if fitness s' > fitness s then
8:          s = 's
9:          Add solution s to Tabu list
10:     else if fitness s' < fitness s then
11:         Tabu list ← s
12:         Choose new random solutions                ▷ Perturbation of the solution
13:     end if
14: end while
```

2.4 Simheuristics

To effectively evaluate a solution using AATOM, multiple simulation instances of the same input are required to get an indication of the fitness of a solution. Bearing in mind that one simulation instance could take between 30 and 60 minutes depending on the number of passengers generated during the simulation, a simulation optimization procedure would require large computational resources. Juan et al. [8] proposes a method for reducing the required computational resources by using a deterministic version of the model and a metaheuristic algorithm to identify a set of elite solutions. This method assumes that high quality solutions for the deterministic version are also high quality for the stochastic version for scenarios with moderate variance [8]. After finding the set of elite solutions, these solutions will be evaluated using the stochastic version of the model to find the optimal solution. A schematic representation of this procedure can be found in Figure 2.

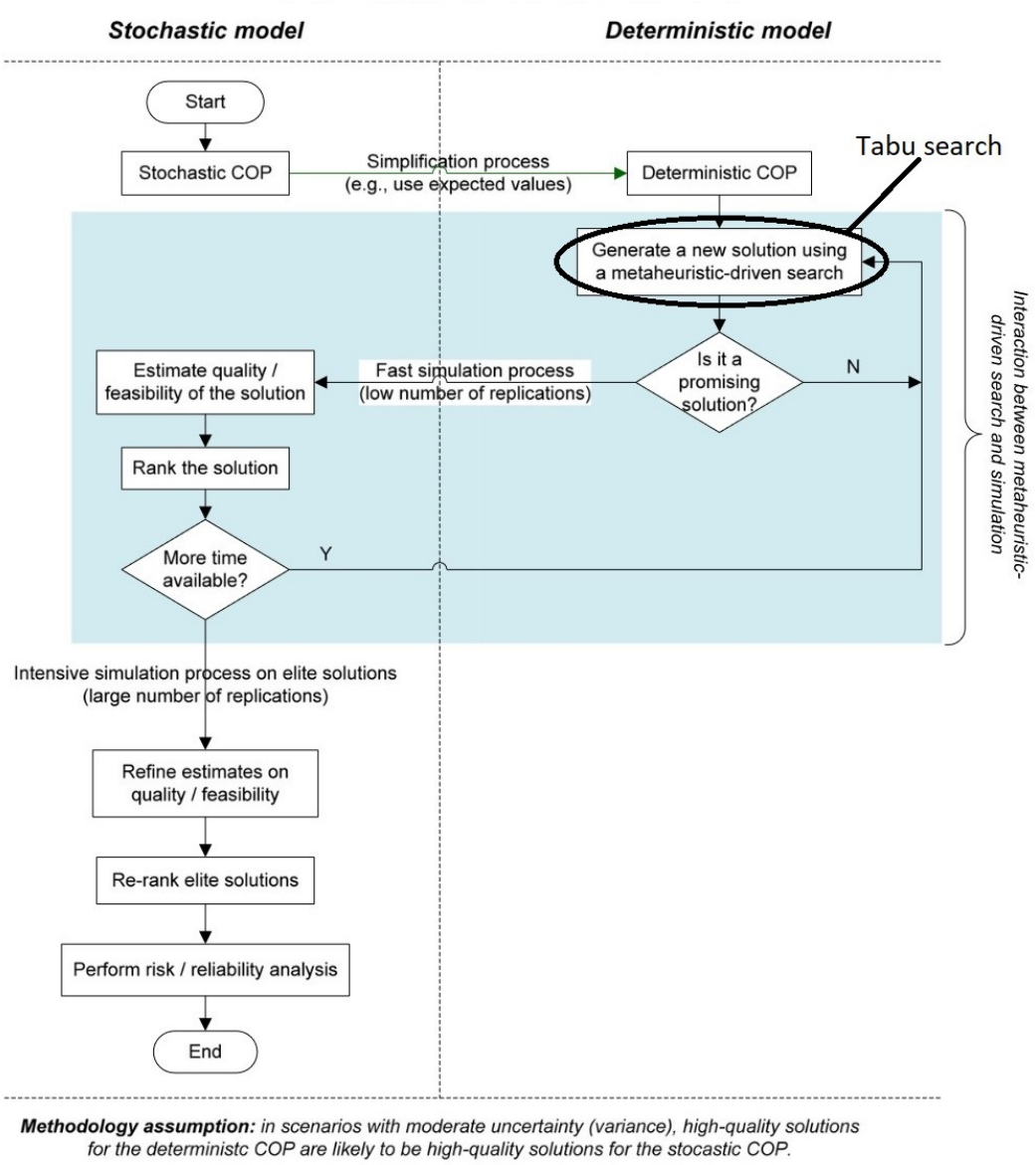


Figure 2: Simheuristic framework [8]

Juan et al. [8] proposes the use of an analytical version of the stochastic model or a surrogate model if available as the deterministic version. If such a model is not available, as is the case for AATOM, replacing all random variables by their expected value is proposed. Therefore, if a Simheuristic framework is used in combination with AATOM, a deterministic version of AATOM must be created.

3 Description of the Case Study

Together with RTHA, Bagchain and TU Delft, this project was set up to help RTHA implement a new self-service luggage check-in system. The current self-service luggage check-in system is an all-in-one bag tag and drop-off system and has to be renewed. This section discusses the case study on which this research is applied. In Figure 3 the layout of RTHA can be found with the kiosks added. The in red squares circled in orange indicate the old self-service luggage check-in system. This location will be used for the drop-off kiosk of the new system as there is connection to the luggage handling system. The circled green squares, denoted with the number 1, are the bag tag kiosks inside the terminal. The circled green squares, denoted with numbers 2 and 3, are the outside bag tag locations. These outside locations were not initially considered as a viable option for a bag tag location. However, to investigate the effect of having multiple locations, these were added.

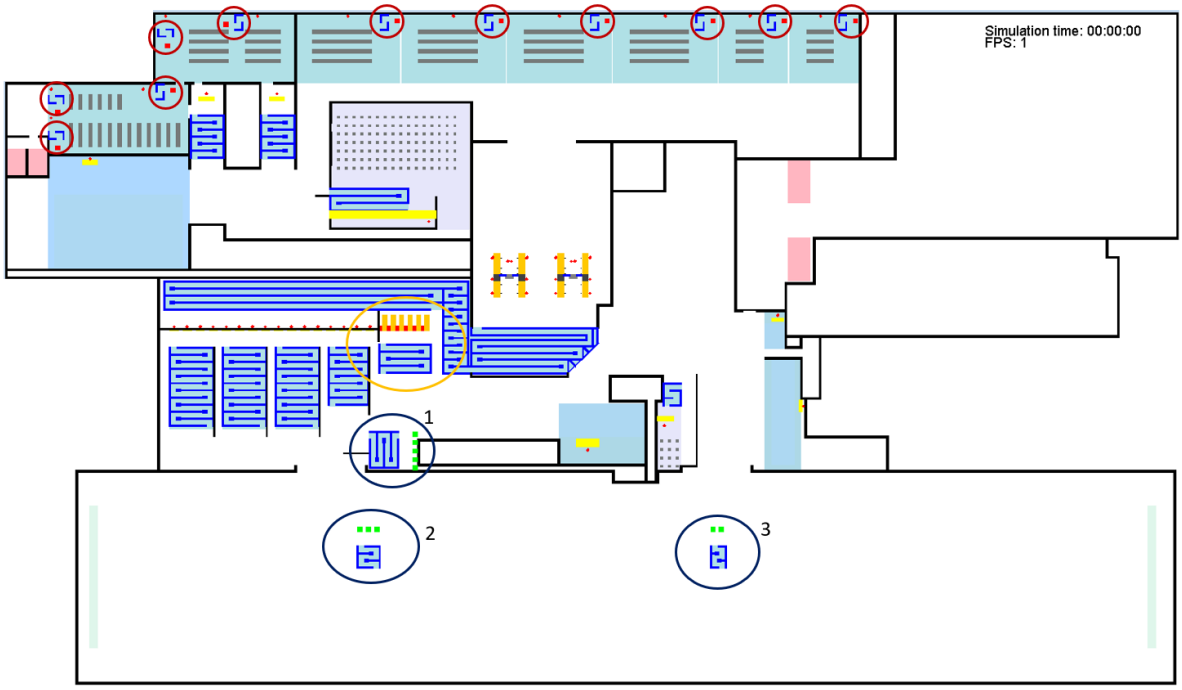


Figure 3: RTHA terminal in AATOM

The red squares circled with red in Figure 3 are the gate kiosks that will be used to bag tag carry-on luggage at the gate. The location of these kiosks at each gate were determined together with RTHA.

4 Extending the Agent-Based Model

In the previous version of AATOM, no self-service luggage check-in system was implemented. In this section the extension of the agent-based model with the self-service luggage check-in system and the verification of this system is discussed.

4.1 Implementation of kiosks before security

The implementation of the self-service luggage check-in system before security requires the addition of bag tag and drop-off kiosks to AATOM. These kiosks are part of the environment and observable by the agents. To decide when an agent needs to utilize the self-service system, the agents get a new characteristic. This new characteristic is an attribute to track whether or not the agent has checked luggage. The actions that an agent needs to perform when it utilizes a self-service kiosk are defined in the activities. New activities have been developed and integrated with AATOM for the bag tag and drop-off kiosks. In the following sections, these newly developed activities will be discussed. In supporting work Appendix 1, a more in depth explanation of the implementation and verification of the self-service system can be found.

Bag tag activity

If a passenger is checked in online and has checked luggage, the passenger must utilize the self-service system and therefore the bag tag activity is added to the goals of that passenger. The bag tag activity consists of choosing a queue, getting into the queue, waiting until a kiosk is free, start interaction with the kiosk, waiting at the kiosk (simulating the kiosk processing information) and retrieving the bag tag. As there are multiple bag tag locations in- and outside the terminal, a bag tag location must be chosen based on the queue. The mechanism that takes care of choosing the queue first observes the number of passengers in the queue of the closest bag tag location. If more than 5 passengers are in that queue, the next closest location is checked. In the case that all locations have queues with more than 5 passengers in them, the queue with the least passengers in it is chosen. This mechanism was chosen to simulate a passenger quickly scanning which queue is least crowded. If all queues are seemingly crowded, a more precise observation is made to choose the least busy one.

Drop-off activity

The drop-off activity always follows the bag tag activity. The drop off activity consists of getting into the queue, waiting until a drop-off kiosk is free, starting interaction with the kiosk, waiting at the kiosk (simulating scanning the bag tag, processing information and dropping of luggage) and leaving the kiosk. A visualization of the luggage getting on the belt was also added.

4.2 Implementation of gate kiosks

The gate bag tag activity is almost an exact copy of the regular bag tag kiosk. The main difference is that carry-on luggage is bag tagged instead of checked luggage. The implementation of the gate kiosk also requires an extension of the gate operator functionality. The gate operator is responsible for deciding whether or not the passenger must utilize the gate kiosk to check-in its carry-on luggage. Additionally, the gate operator must have the capability to bag tag carry-on luggage manually in case no gate kiosk is present.

To decide if a piece of carry-on luggage must be checked in, the gate operator tracks how many pieces of carry-on luggage have boarded the aircraft. If at some point during the boarding process the capacity of the aircraft for carry-on luggage is reached, the gate operator starts checking-in carry-on luggage. This capacity is dependent on the aircraft and can be defined per flight in AATOM. In case of a kiosk being present at the gate, the passenger will need to complete the bag tag activity at the kiosk before it is allowed to board the aircraft type. When no kiosk is present, the gate operator will manually bag tag that piece of carry-on luggage. Manual bag tagging is simulated by letting the passenger wait at the gate operator after which the bag tag status and checked in status of that piece of carry-on luggage is updated.

4.3 Verification of self-service luggage check-in process

To ensure that the self-service system was implemented correctly, several aspects of the self-service system were tested. The tests were conducted by utilizing the graphical user interface (GUI) of AATOM as well as manual and automated tests. The verification is done on two levels: the facility level and the terminal level. On the facility level, only one facility, such a bag tag location, is simulated at a time. On the terminal level, all facilities are simulated simultaneously.

Self-service luggage check-in before security

By verifying via the GUI that the passengers follow the procedure for the bag tag and drop-off kiosks as discussed in section 2, it is determined that the self-service luggage system was implemented correctly. The waiting time which simulates the passengers interacting the kiosks is also verified to be correct via the GUI and tests. Getting the passengers to queue correctly and to not get stuck required inserting physical objects such as walls. These physical objects affect the path planning of the passenger to guide the passengers along the desired path.

Gate kiosk

The functionality of the gate kiosk is verified in a similar fashion as the bag tag kiosk before security. The gate operator is verified to correctly send passengers to the kiosk as well as manual bag tagging by utilizing the GUI. The tracking of the boarded pieces of large carry-on luggage is also verified via an automated test. The point in the boarding process where the operator decides to start checking in carry-on luggage is also verified to be correct.

Complete terminal

As the individual elements of the self-service system are verified, the system as a whole (the terminal) can be verified. This is done by utilizing the GUI to investigate if no passenger get stuck due to the added kiosks and queues. Additionally, the passenger flows that emerge are verified to be correct. The flows identified emerging flows are:

- entrance - check-in - security - gate
- entrance - security - gate
- entrance - bag tag kiosk - drop-off kiosk - security - gate

As all three bag tag locations are implemented, the queue selection process can also be verified. Via the GUI, the threshold number of passengers queuing at which another queue is chosen is verified to be correct.

5 Integration of the Agent-Based Model with Simheuristic Framework

The next step towards the objective of this research is to integrate AATOM with the Simheuristic optimization framework. To achieve this, a deterministic version AATOM must be constructed and verified. Additionally, the tabu search algorithm must be integrated with AATOM. In this section, the steps taken to prepare AATOM for a Simheuristic optimization procedure are explained. A more elaborate explanation of the implementation of the Simeheuristic framework can be found in supporting work Appendix 2.

5.1 Creating the deterministic version of AATOM

Since there is no deterministic version of AATOM readily available, one must be created. Juan et al.[8] proposes a method for developing such a deterministic version. By identifying the random variables and replacing them with their expected value, a stochastic model can be made deterministic. This method can be applied to the majority of the random variables in AATOM, but poses problems for the deterministic agent generator. The steps taken to make the agent generator deterministic are explained in the section below.

Deterministic agent generator

The agent generator is responsible for generating the agents during the simulation. During the generation of the agent the following characteristics need to be determined:

- Arrival time: the time at which the agent is generated
- Check in status: whether or not the agent is checked in online
- Checked luggage status: whether or not the agent has checked luggage

In the stochastic version of AATOM, the arrival times are generated as a Poisson process resulting in exponentially distributed inter-arrival times. Therefore, if we know the rate parameter λ, which is the number of passengers that will arrive in a certain time window, the expected inter-arrival time will simply be $\frac{1}{\lambda}$. The check-in and checked luggage status is determined using a uniformly distributed random variables between zero and one, and a predefined threshold. This threshold represents the percentage of passengers having certain characteristics such as being checked in online. If the random variable is higher than the specified threshold, the passenger is not checked in online, if it is lower, the passenger is checked in online. Using the expected value of this random variable is not possible as it would always result in all passengers being either checked in or not checked in online. To overcome this issue, instead of deciding during the simulation what the characteristics are of the passengers, the characteristics are determined before the start of the simulation.

Verification

The verification process of the deterministic version of AATOM consists of four tests. The first test consists of testing the new deterministic agent generator. The following aspects were tested using manual and automated tests:

- Test whether correct number of agents are generated (manual test)
- Test whether ratio of checked in passengers and luggage types per passenger are as defined in the inputs (automated test)
- Test whether arrival intervals are uniformly distributed (manual test)

The second test utilized the coefficient of variation (CoV) of the average waiting times for the separate facilities (check-in, security, etc.). The CoV resulting from the stochastic version was compared to the CoV resulting from the deterministic version. This test was used to verify that making a deterministic version of AATOM was successful. The calculated CoV of the average waiting time after multiple simulation runs must be approaching zero for the deterministic version. Consequently, the CoV resulting from the deterministic version will be lower than the CoV of the average waiting times resulting from the stochastic version. The results of this comparison can be found in Table 1. In this table, CI stands for check-in counters, BT for bag tag kiosk and DO for drop-off kiosk. The CoV was calculated after 256 simulation runs of the same input. The CoV for the deterministic version, has not become zero but at least a factor 10 smaller than the stochastic version. The CoV not becoming zero is most likely caused by the getting stuck prevention function. This function detects when a passenger is stuck and performs a random movement to resolve this issue.

Table 1: CoV of the average queue times for the separate facilities after 256 runs

Version	**CI1**	**CI2**	**CI3**	**BT**	**DO**	**Security**
Stochastic	0.11	0.12	0.14	0.0036	0.014	0.00087
Deterministic	0.00047	0.0019	0.00067	0.00056	0.00047	0.000051

The next test consists of examining whether the average waiting times for each separate facility is similar for the stochastic and deterministic version. In Table 2, the results of this comparison can be found. The average waiting times were calculated after 256 simulation runs of the same input. As can be seen, the average waiting times are similar for both versions of AATOM. This is an indication that the deterministic version of AATOM can be used to estimate the waiting times per facility.

Table 2: Mean values of the average queue times for the separate facilities after 256 runs

Version	**CI1**	**CI2**	**CI3**	**BT**	**DO**	**Security**
Stochastic	1006.3	914.3	675.5	22.2	63.8	309.8
Deterministic	1127.4	926.2	636.8	23.6	62.5	309.7

With the verification on facility level finished we can look at the whole terminal. The resulting waiting times per facility can be found in Table 3. The average waiting times are similar except for the waiting times at the security checkpoint. This is most likely caused by interaction of the different stochastic distributions. Once a passenger gets further in the terminal, their behavior and moving paths are increasingly influenced by diverse stochastic processes. This confirms the need for evaluation of elite solutions with the stochastic version as the deterministic version alone is not able to capture the interaction of stochastic processes.

Table 3: Average waiting time per facility resulting from the whole terminal after 256 runs

Version	**CI1**	**CI2**	**CI3**	**BT1**	**BT2**	**BT3**	**DO**	**Security**
Stochastic	234	192	19	41	59	59	65	480
Deterministic	214	208	22	41	55	55	64	331

The final verification step of the deterministic version of AATOM is investigating whether the outputs of the deterministic version react logical to input changes. These inputs are in the form of a four digit number where each digit refers to the number of kiosks at a location. The first digit refers to the number of bag tag kiosks at location one, the second digit to the number of bag tag kiosks at location two, the third digit to the number of bag tag kiosks at location three and the final digit to the number of drop-off kiosks. The inputs of AATOM are explained further in section 5.2.

The inputs for this test are chosen in such a way that a change in output is expected. In Table 4 the output for each evaluated input can be found. The output is the average waiting time per passenger for each facility. Comparing the output of simulation setting one to the output of simulation setting two, a decrease in average waiting times for the bag tag kiosk locations and drop-off location can be seen. This is a result of the number of kiosks being increased substantially, resulting in a higher throughput of passengers and lowering the waiting times. This increase in throughput also result in an increased average waiting time at security.

Comparing the output of simulation setting three and four, the number of bag tag kiosks at locations two and three are swapped. At simulation setting three, bag tag location two has a longer average waiting time than bag tag location three. For simulation setting four, bag tag location three has a longer average waiting time than bag tag location two. This is in line with the expected behavior.

Table 4: Average waiting times per passenger resulting from the deterministic version for different settings

Simulation setting	**BT1 (s)**	**BT2 (s)**	**BT3 (s)**	**DO (s)**	**Security (s)**
1 [2,1,1,3]	24.6	45.8	48.4	70.2	340
2 [4,3,3,6]	0	33.3	33.2	62.5	345.2
3 [4,1,3,6]	19.2	45.8	33.7	62.2	346.1
4 [4,3,1,6]	23.6	33.3	42.2	63.4	344.1

5.2 Optimization procedure

The final aspect of the Simheuristic simulation optimization procedure that need to be defined is the optmization procedure. In this section first the relevant decision variables are discussed after which the outputs of AATOM are discussed together with the fitness function. To conclude this section, the determination of the neighborhood of a solution and solution perturbation are discussed.

Decision variables

For the simulation optimization procedure using tabu search, AATOM must be prepared for integration. AATOM must be able to receive inputs as decision variables for evaluation and output waiting times to determine the fitness. The decision variables of the optimization procedure can be found below.

- Number of bag tag kiosks at location 1
- Number of bag tag kiosks at location 2
- Number of bag tag kiosks at location 3
- Number of drop-off kiosks

Fitness function

After the evaluation of a solution, AATOM outputs the average waiting times at the bag tag kiosk locations, drop-off kiosk location and security. Using these outputs the fitness of a solution can be determined. The fitness of a solution is determined using a weighted and linearized function. The objective is to minimize the waiting times at the bag tag kiosk locations, drop-off kiosk location and security while also minimizing the number of kiosks. Security is taken into account as this is the first facility after check-in. Optimizing the check-in process to the point that the security checkpoint can not cope with the passenger in-flow is undesirable. The variables that are used for the determination of the fitness and their weights can be found in Table 5.

Table 5: Variables of the fitness function and their weights

Variable	Weight
Average bag tag waiting time of all locations	0.15
Drop-off waiting time	0.2
Security waiting time	0.2
Number of bag tag kiosks	0.3
Number drop-off kiosks	0.15

The weights were chosen in such a way that, for the bag tag kiosks, reducing the number of kiosks was prioritized due to the three locations. For drop-off kiosks reducing the waiting time was prioritized since there is only one location. If this location gets congested, there is no alternative. The remainder of the weight was assigned to the security waiting time.

The outputs used in the fitness function will be linearized. For the waiting times, multiple simulations were run for settings which were expected to yield high waiting times. These values serve as the maximum value in the linearization function while the minimum values are set to zero. Additionally, the minimum and maximum number of bag tag and drop-off kiosks are constrained by the available space in the terminal. The minimum and maximum number of bag tag kiosks are set to 4 and 14 respectively. For the drop-off kiosks these values are set to 2 and 8. The linearization function can be found in Equation 1.

$$x_{linearized} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{1}$$

Neighborhood determination

As mentioned in section 2.3, during each iteration the neighborhood of the current best solution is determined. The neighborhood of a solution is determined by increasing and decreasing the number of kiosks for each decision variable by one. This results in a neighborhood of eight solutions for a solution consisting of four decision variables. To illustrate this, the neighborhood of solution [2,2,2,2] is given as an example:

- [3,2,2,2]: decision variable one increased by one kiosk

- [1,2,2,2]: decision variable one decreased by one kiosk
- [2,3,2,2]: decision variable two increased by one kiosk
- [2,1,2,2]: decision variable two decreased by one kiosk
- [2,2,3,2]: decision variable three increased by one kiosk
- [2,2,1,2]: decision variable three decreased by one kiosk
- [2,2,2,3]: decision variable four increased by one kiosk
- [2,2,2,1]: decision variable four decreased by one kiosk

Solution perturbation

The solution perturbation is used when a local minimum, as the fitness value is minimized, is found. This is done by generating a new random solution that is not part of the tabu list.

6 Experimental Setup

To investigate the effect of an optimally configured self-service luggage check-in system before security and gate kiosks on the terminal of RTHA, simulations will be performed. The experimental setup will be discussed in this section.

During the development of the model it was found that running the full terminal process is very computationally intensive. Therefore, splitting the experiment in two separate ones is proposed. The 'configuration experiment' consists of finding an optimal configuration for the kiosks before security and investigating the impact of this kiosk configuration on airport terminal performance. In this experiment the passengers will be simulated from entering the terminal to leaving the security checkpoint. Removing the passengers after they leave security reduces the maximum number of passengers in the simulation, thus reducing the computational resources required. The second experiment, called the 'gate experiment', investigates the effect of having a bag tag kiosk at the gate on the total boarding time.

6.1 Configuration Experiment

In the configuration experiment the performance of the terminal with the optimized kiosk configuration is compared to the performance of the baseline configuration. This baseline configuration has no self-service luggage check-in system, meaning that all passengers that are not checked in or have checked luggage, have to utilize regular check-in. The performance will be measured by the waiting times of three passenger types. The passenger types and facilities they need to go through can be found in Table 6.

Table 6: Passenger types and the facilities they need to go through

Passenger type	**Optimized configuration**	**Baseline configuration**
Checked in online, Checked luggage	Self-service kiosks - Security	Regular check-in - Security
Checked in online, No checked luggage	Security	Security
Not checked in online	Regular check-in - Security	Regular check-in - Security

An important setting for this experiment which is yet to be defined, is the flight schedule. The schedule used in this experiment consists of 7 flights that leave within a time window of approximately an hour. The total number of simulated passengers is 849 and these passengers start arriving two and a half hours before their flight. This flight schedule is comparable to a moderately busy morning at RTHA. A peak busy morning could not be simulated due to limitations in terms of computational resources. Another factor that needs to be taken into account during the definition of the flight schedule, is the number of passengers that physically fit in a queue. This limitation will be discussed further in section 8.

The next aspect of the setup of this experiment is the definition of the settings for the tabu search algorithm. These settings consist of the number of iterations, number of elite solutions that will be evaluated with the stochastic version and the tabu list length. Fifty iterations of the optimization procedure and a total of four elite solutions will be evaluated. These settings are constraint by the computational resources available. The

final setting that needs to be defined is the tabu list length which is to set 10.

Finally, the constraints for the number of bag tag and drop-off kiosks can be defined. For the bag tag kiosks inside (location one) a minimum of two and a maximum of six kiosks is determined. This minimum is defined to ensure that kiosks would be placed inside as this is the preferred location by RTHA. The number of outside bag tag kiosks (locations two and three) is limited to six. Additionally, the total number of bag tag kiosks of all locations combines must always be more than three. The number of drop-off kiosks is constrained between two and eight. The maximum number of drop-off kiosks is limited by the available space in the terminal.

Sensitivity analysis

To investigate the effect of certain variables on the optimal configuration, a sensitivity analysis will be performed. Below, the variables which will be used for the sensitivity analysis can be found.

- Percentage of passengers having checked luggage, will be referred to as demand level (DL)
- Service rate of bag tag kiosks (BT SR) (average time a passenger spends interacting with a bag tag kiosk)
- Service rate of drop-off kiosks (DO SR) (average time a passenger spends interacting with a drop-off kiosk)

Using aforementioned variables for sensitivity analysis, settings can be defined for which the Simheuristic simulation optimization procedure will be performed. In Table 7, the 12 settings that will be used for the configuration experiment can be found.

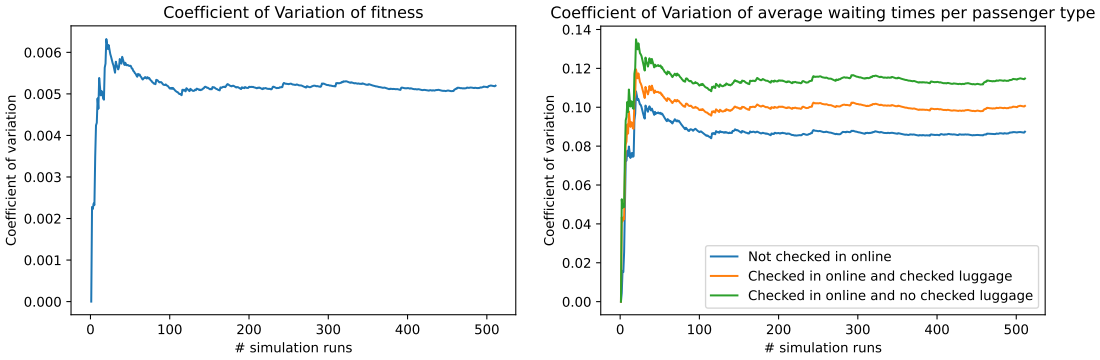
Table 7: Settings that will used in the configuration experiment

Setting	**Checked luggage**	**Bag tag service rate** [s]	**Drop-off service rate** [s]
1	40%	20	30
2	60%	20	30
3	80%	20	30
4	40%	20	50
5	60%	20	50
6	80%	20	50
7	40%	40	30
8	60%	40	30
9	80%	40	30
10	40%	40	50
11	60%	40	50
12	80%	40	50

Number of simulation runs

During the Simheuristic optimization procedure, no repetitions are required since the deterministic version is used. When the elite solutions have to be evaluated however, repetitions are required as the stochastic version is used for this evaluation. This is also the case when the optimized configuration will be compared against the baseline configuration.

The CoV is analyzed to understand how many simulation runs are necessary. For the elite solution evaluation the CoV of the fitness of a solution is examined. For the configuration comparison, the CoV of the waiting times per passenger type is analysed. The results of this analysis can be found in Figure 4.



(a) Elite solution evaluation

(b) Configuration comparison

Figure 4: Results from CoV analysis for the elite solutions evaluation and configuration comparison

From this CoV analysis the number repetitions required for the elite solutions was set to 128 runs as the CoV has then stabilized. The number of runs required for the configuration comparison was set to 256.

Parallel tabu search optimization

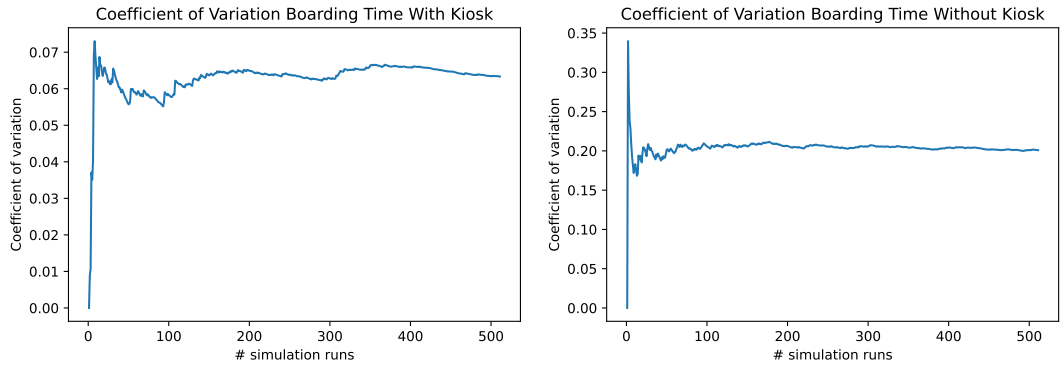
To decrease the time required to run all simulations for this experiment, a server, where up to 256 simulations can be run in parallel, will be used. To perform this experiment as efficient as possible, the capacity of the server must be used maximally. Therefore, the Simheuristic simulation optimization software is designed to run multiple optimization procedures in parallel. This reduces the number of simulation runs during the tabu search optimization phase from 600 (12 settings times 50 iterations) to 50. The optimization software is discussed more extensively in supporting work Appendix 2.

6.2 Gate Experiment

For the gate experiment the boarding process of one gate will be simulated. The simulation of the boarding process will be done for three passenger settings, with and without kiosk, totaling six evaluated settings. The passenger settings refer to the percentage of passengers bringing carry-on luggage. The passenger settings are set to: 60%, 70% and 80%. In total 140 passengers will need to board the aircraft. The settings for the service rate of the gate kiosk and manual bag tagging were obtained from Bagchain[2]. On average a passenger will spend 30 seconds at the gate kiosks and manual bag tagging will take 75 seconds. The Boeing 737-800 is used as a reference for the capacity of large carry-on luggage which will be set to 65% [3].

Number of simulation runs

As the stochastic version of AATOM is used for this experiment, the CoV needs to be analyzed to determine the number of required simulation runs. The CoV of the boarding time is used for this analysis and the results can be found in Figure 5.



(a) With kiosk

(b) Without kiosk

Figure 5: Coefficient of variation plot to determine number of runs for experiment with and without kiosk

For both with and without a kiosk at the gate, 256 simulation instances were deemed to be sufficient.

7 Results

The results of the two experiments can be found in this section. Fist the results from the configuration experiment section 7.1 will be discussed and after that the results from the gate experiment in section 7.2.

7.1 Configuration experiment

This experiment is designed to evaluate two outcomes simultaneously: the performance of the Simheuristic framework and the impact of an optimally configured self-service system on the terminal performance. Two hypotheses will be tested:

- **Hypothesis 1A:** The average waiting times resulting from the optimal configuration for passenger types one and three are lower than the average waiting times for passenger types one and three resulting from the baseline configuration.
- **Hypothesis 1B:** The average waiting times resulting from the optimal configuration for passenger type two will be similar to the average waiting for passenger type two resulting from the baseline configuration.

Simheuristic performance

To examine the performance of the Simheuristic framework we will first look at the best candidate fitness throughout the simulation procedure. In Figure 6, the fitness of the best candidate for each iteration for sensitivity analysis setting one can be found. It can clearly be seen that the algorithm searches for a local minimum and resets when a local minimum is found. Each iteration, the fitness decreases until a local minimum is found. When this minimum is found, a new random solution is generated which often results in worse fitness. This indicates that the tabu search algorithm works as expected.

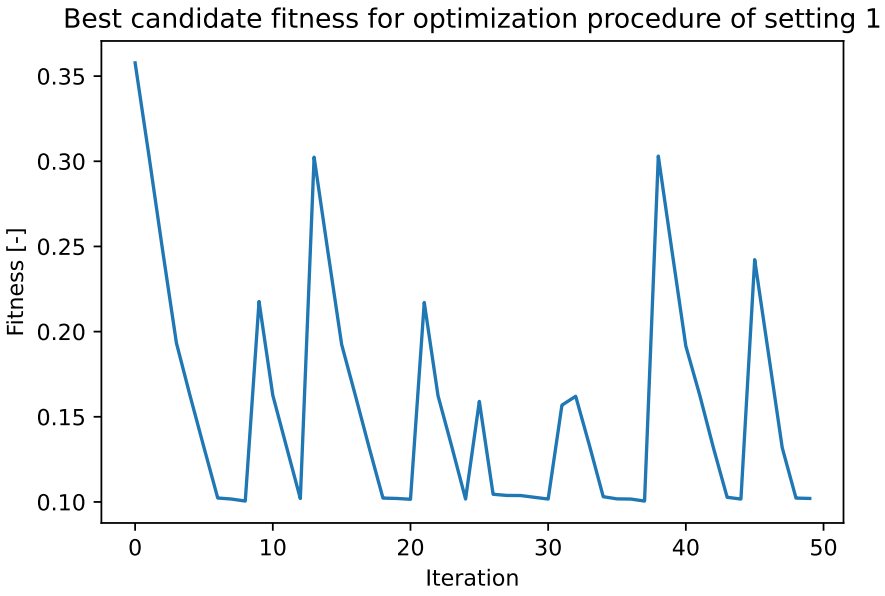


Figure 6: Best candidate fitness during optimization procedure for setting 1

To determine whether the Simheuristic framework can be used in combination with an agent-based model, the fitness values for the four elite solutions are calculated by both the deterministic and stochastic version of AATOM. The resulting fitness values are then ranked from lowest to highest to investigate if the ranking resulting from both versions is comparable. In Table 8 the elite solution ranking for sensitivity analysis setting nine can be found. The ranking is not the same for both versions. Important to note is that the fitness values for rank one, two and three are very close. The fitness of rank four is, compared to the others, substantially worse, which is found by both the stochastic and deterministic version. This gives reason to believe that the deterministic version can be used to roughly estimate which solutions will perform well.

Table 8: Ranking comparison between deterministic and stochastic version for setting: (dl:0.8, bt sr: 40 s, do sr 30)

Input	Rank from Deterministic Version	Rank from Stochastic Version
[2,5,5,3]	1 (fitness 0.1021)	3 (fitness 0.1402)
[2,2,5,3]	2 (fitness 0.1021)	1 (fitness 0.1392)
[2,3,2,3]	3 (fitness 0.1038)	2 (fitness 0.1396)
[3,3,4,3]	4 (fitness 0.1320)	4 (fitness 0.1696)

The final aspect of the Simheuristic performance that is investigated is the required computational resources. As was found in section 6.1, 128 repetitions are required to get an accurate fitness value of a solution using the stochastic version. Due to the neighborhood size of eight, per iteration of the tabu search procedure 1024 simulations would have to be run when using the stochastic version. This can be done by running 256 simulations on the server in parallel four times. An iteration of the optimization procedure using the stochastic version is therefore four times more computationally intensive than the Simheuristic optimization procedure. Additionally, parallel tabu search simulation optimization is not possible when using the stochastic version. Keeping in mind that the deterministic version also runs approximately 50% faster on the server, most likely due to the generation of agents being more spread out, a large gain can be made in terms of required computational resources.

Configuration comparison

As the implementation of the self-service kiosks could not be validated with real world data, the decision has been made to investigate the relative waiting times per passenger type. Before looking at these waiting times, we must look at the optimal configuration per setting. The optimal number of bag tag and drop-off kiosks can be found in Table 9.

Table 9: Optimal number of self-service kiosks per setting

Setting	Optimal number of bag tag kiosks				Optimal number of drop-off kiosks
	Loc. 1	Loc. 2	Loc. 3	Total	
1	2	3	4	9	3
2	2	0	5	7	3
3	2	4	3	9	3
4	2	0	4	6	3
5	2	2	3	7	3
6	2	0	5	7	4
7	2	3	5	10	3
8	2	0	4	6	3
9	2	2	5	9	3
10	2	2	2	6	3
11	2	2	2	6	3
12	2	0	5	7	4

From the optimal configuration presented above, no effect can be found of changing the service rates and percentage of passengers bringing checked luggage on the optimal number of bag tag kiosks. The optimal number of bag tag kiosks per location also shows that the optimal number of bag tag kiosks inside the terminal is always two. This is equal to the minimum number of kiosks at this location due to the constraints. This phenomenon could be caused by the combination of the flight schedule not being busy enough and the queue selection mechanism. As a consequence, not enough passengers are generated in a short period of time to force the passengers to use the inside location requiring more kiosks inside the terminal.

The effect of changing the service rates and percentage of passengers bringing checked luggage on the optimal number of drop-off kiosks, is limited but apparent. For the setting where the drop-off kiosks have a long service rate setting and a high percentage of passengers are bringing checked luggage, an extra kiosk is deemed necessary.

To investigate the effect of the optimal configuration on airport terminal performance, the relative waiting times resulting from the optimal and baseline configuration are compared. The results of this comparison can be found in Table 10. For each setting, the waiting times are shorter for passenger types one and three while the waiting time for passenger type two remains equal.

Table 10: Relative waiting times per passenger type of optimized configuration vs. base configuration for all settings

Setting	Waiting time per passenger type of optimized configuration relative to base configuration		
	Type 1	**Type 2**	**Type 3**
1	-17%	-0.6%	-9%
2	-21%	-2%	-12%
3	-22%	-2%	-16%
4	-19%	-1%	-10%
5	-15%	-4%	-14%
6	-23%	-6%	-19%
7	-18%	-1%	-11%
8	-25%	-3%	-17%
9	-24%	-3%	-18%
10	-17%	-0.9%	-10%
11	-16%	-5%	-15%
12	-22%	-5%	-18%

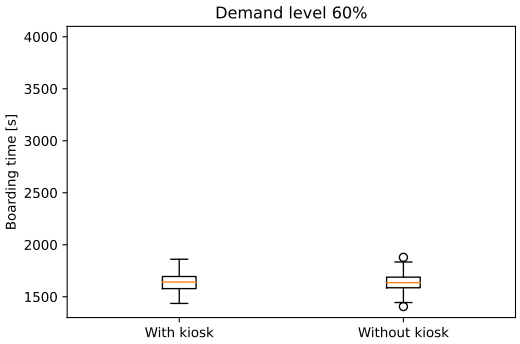
To investigate the statistical significance of the results, three tests have been conducted. First, a normality test has been applied to the experiment results to determine whether or not these results are normally distributed. Based on the normality test, a paired t-test or a Wilcoxon signed rank test is used. The paired t-test is used in case two normally distributed data sets are compared and the Wilcoxon signed rank test is used when one of the two or both data sets are not normally distributed. These tests return the probability that both data sets are sampled from the same statistical distribution. The final test the experiment results have been subjected to is the Vargha-Delaney A-test. This test is a non-parametric effect magnitude two sample test. It returns the probability that a randomly sampled observation from one sample is larger than a randomly sampled observation from the other sample. These tests combined give a complete insight in the statistical significance of the results. The findings presented above are confirmed by the statistical tests that have been performed, thus supporting hypothesis **1A** and **1B**.

Complementary results of the configuration experiment and the results of the statistical tests can be found in supporting work Appendix 3.

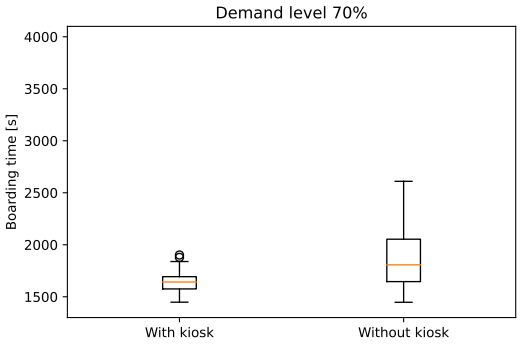
7.2 Gate Experiment

The second experiment that has been conducted is the gate experiment. A hypothesis is formulated for this experiment: **The boarding time of a gate with a bag tag kiosk will be shorter than the boarding time of a gate without a bag tag kiosk.**

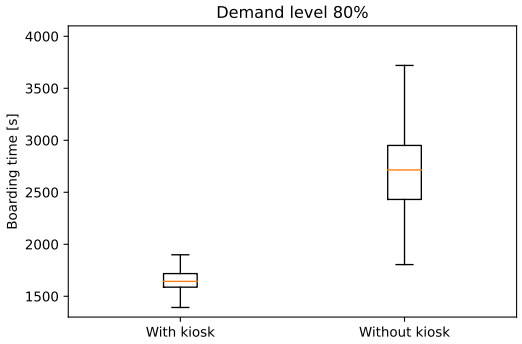
The results of the gate experiment in boxplot format can be found in Figure 7. From these figures, the increase in boarding time without a kiosk becomes apparent as the percentage of passengers carrying large carry-on luggage increases. Next to this increase, it is important to note that variance of the boarding time for a gate without kiosk also increases, meaning that the predictability decreases.



(a) Result of gate experiment for demand level 60%



(b) Result of gate experiment for demand level 70%



(c) Result of gate experiment for demand level 80%

Figure 7: Boarding times resulting from gate experiment for kiosk and no kiosk configuration for different demand levels

Table 11 shows that the average boarding time is similar for the 60% setting and shows an increase in average boarding time for the 70% and 80% setting. The cause of the boarding time not increasing without a kiosk for the 60% setting is the capacity of carry-on luggage of the aircraft. This capacity is set to 65%, as discussed in section 6.2, therefore the chance that the maximum capacity of the aircraft is reached is small. This results in that very little carry-on luggage needs to be checked in.

Table 11: Average boarding time comparison between gate with kiosk and gate without kiosk for different demand level settings

Demand level	**Average boarding time without kiosk [s]**	**Average boarding time with kiosk [s]**	**Relative average boarding time kiosk vs. no kiosk**
60%	1636	1639	0.020%
70%	1875	1641	-13%
80%	2702	1649	-39%

An identical approach to statistical testing is taken for this experiment as for the configuration experiment. The results from the paired t-tests, Wilcoxon signed rank test and Vargha-Delaney A-test confirm the findings presented above. The results of this experiment therefore support the hypothesis formulated for this experiment.

Complementary results of the statistical tests of the gate experiment can be found in supporting work Appendix 3.

8 Discussion

To accurately draw conclusions from this research, the implications of the choices made in this research are first discussed. In this section, a discussion is presented for the model and result validation, assumptions and the application of Simheuristics on agent-based models.

8.1 Limitation of AATOM

The main limitation of AATOM that affected this research is AATOM requiring sufficient space for agents to queue. If a queue provides space for 20 passengers for example and 21 passengers try to queue, the last passenger will try to force itself into the queue. This is not possible since there is not enough space and after a certain amount of time the passenger will try to find another way in. This often results in the passenger skipping the queue and walking to the exit of the queue. A consequence of this action is that all passengers get stuck in the queue since the exit is blocked. This could render the simulation useless. For this research, where the terminal of RTHA is used, there is a limited amount of space in the terminal for queues. Without changing the queuing mechanism, this phenomenon can only be prevented by reducing the total number of passengers in the simulation.

8.2 Experiments

Both experiments that have been conducted show promising results which indicate that implementing a self-service luggage check-in system has a positive effect on airport terminal performance. These experiments however could be more extensive as they are now constraint by the computational resources available. For the configuration experiment, to determine an optimal configuration for a peak departing flight schedule, more passengers should be simulated for each evaluation run. Additionally, more optimization iterations could further improve the quality of the optimal solution.

For the gate experiment, one aircraft type with fixed carry-on luggage capacity and fixed number of passengers on the flight is now simulated. To better understand when placing a gate bag tag kiosk at a gate is most beneficial, different aircraft types with different carry-on luggage capacities, and varying number of passenger on the flight should be simulated.

8.3 Model and result validation

As mentioned in section 4, real-world data for validation was not available. Even though the concept of operations was validated by a self-service kiosk expert, some remarks have to be made about the validity of the results. The results presented only hold for the settings which were tested. The results seem promising but there is no guarantee that the results would be similar when RTHA implements a self-service luggage check-in system.

To have some form of validation, the results of this research have been presented to an airport operations expert. The results of the gate experiment have been validated to be intuitive and logical. The boarding time drastically increasing for the gate without a kiosk has been encountered in airports by the airport operations expert. An increase in boarding time was also expected at the gate with kiosk. This could be a result of the service rate of both kiosk and gate operator varying per kiosk manufacturer and gate operator. The results of the configuration experiment were also validated to be intuitive although such large improvements have not been encountered by the airport operations expert.

8.4 Assumptions

During the implementation of the self-service kiosks in AATOM a queue selection mechanism was developed. The queue selection process of passenger was developed to act in two stages: a quick global scan to see if a queue was not crowded and a more thorough scan if all queues were crowded. A different queue selection mechanism could change the spread of passengers, impacting the optimal configuration.

Another assumption was made during the development of the deterministic version of AATOM. This assumption regards the interaction of the different random variables in AATOM. In the deterministic version, each random variable is replaced with its expected value. This resulted in the average results of the waiting per facility for both the deterministic and stochastic version to be similar. Replacing the random variables however did not take into account the interaction of the different stochastic processes. From the results it becomes apparent that for small changes in fitness the deterministic version is unable to rank solutions correctly, which could be a result of the interaction of random variables. Other methods for developing a deterministic version could result in a more accurate estimation of a solution.

8.5 Simheuristics for agent-based models

When using agent-based models, computational resources are often the limiting factor. Simheuristics could decrease the computational resources required if an appropriate deterministic version of the model is available or could be developed. The development of a deterministic version for an agent-based model is more challenging due

to the high level of detail. If a deterministic version is too simplistic it could evaluate solutions incorrectly, which could result in solutions being incorrectly identified as 'elite'. Other methods for constructing a deterministic version could be explored. Surrogate modeling is a promising method as it can also be used for other purposes than the Simheuristic framework. Summarizing, the main limitation of the Simheuristic framework is the need for a deterministic version which essentially requires the development of an extra simulation model.

9 Conclusions and Recommendations

The objective of this research was to investigate the impact of an optimally configured self-service luggage check-in system on airport terminal performance. The optimization of the configuration of self-service luggage check-in kiosks was done using a Simhueuristic framework. Additionally, the effect of having a bag tag kiosk at the gate on the boarding time was investigated.

From the configuration experiment it has become apparent that, for the settings used and tuning of the tabu search algorithm, no definitive optimal configuration of the bag tag kiosks could be found. In terms of drop-off kiosks however, it was found that four kiosks are sufficient for the most demanding setting evaluated. The implementation of self-service luggage check-in showed a decrease in overall passenger waiting times at check-in while not increasing the waiting times at security. By carefully implementing a self-service system, the check-in process can thus be made more efficient while not compromising the performance of the security checkpoint.

The application of Simheuristic simulation optimization using an agent-based model showed promising results in the configuration experiment. The ranking of the elite solutions indicate that the deterministic version of AATOM is capable of roughly filtering the elite solutions from the solution space. More research is necessary however to claim this with certainty. The computational resources required show a substantial decrease when the Simheuristic framework is applied.

The gate experiment showed that boarding time is related to the percentage of passengers having carry-on luggage. This experiment also showed that the boarding time can be decreased by having a bag tag kiosk at the gate. An extra benefit of using a gate kiosk was found in terms of the predictability of the boarding time. It became more predictable, which could have a positive effect on the turnaround time of an aircraft for example.

For future research it is vital to validate the added self-service luggage check-in system with real-world data. Data on various airport and passenger characteristics is required to formulate a comprehensive recommendation regarding the implementation of such a system. Additionally, more research on the application of the Simheuristic framework on agent-based models can be done. This includes utilizing other approaches to develop the deterministic version of the simulation model, such as surrogate modeling. Finding an approach to making a deterministic version which has other purposes than usage in combination with the Simheuristic framework, would increase the value of the deterministic version. The final recommendation we would like to make is to improve the queuing mechanism of AATOM to make it more robust. This would eliminate the need to prevent a queue area becoming too crowded by adjusting the flight schedule.

As a final remark we would like to stress that the results only hold for the settings used and assumptions made. For an effective implementation of the self-service luggage check-in system, airports should collect data on the luggage types of passengers and service rates of the self-service luggage check-in system. Nonetheless, these findings could serve as a reference for airports that are looking to efficiently implement a self-service luggage check-in system. For the terminal layout of RTHA and the flight schedule used in this research, the waiting times at check-in can be reduced while not compromising the waiting times at the security checkpoint. Additionally, placing a gate bag tag kiosks at a gate decreases the boarding time and increases predictability. Combining the findings of this research, it can be concluded that the complete self-service luggage check-in system positively impacts airport terminal performance.

References

[1] Satyajith Amaran, Nikolaos V. Sahinidis, Bikram Sharda, and Scott J. Bury. Simulation optimization: a review of algorithms and applications. *Annals of Operations Research*, 240(1):351–380, May 2016. doi: 10.1007/s10479-015-2019-x.

[2] Bagchain. bagchain gate kiosk. URL: https://bagchain.aero/index.php/bagchain-gate/.

[3] Boeing. Space Bins, 2023. URL: https://www.boeing.com/commercial/737max/space-bins/.

[4] Fortune Business Insights. Airport Services Market Size, Share, Growth | Global Report, 2029, 2022. URL: https://www.fortunebusinessinsights.com/airport-services-market-102855.

[5] International Air Transport Association. *Airport development reference manual.* IATA, London, 9th ed edition, 2004.

[6] S. a. M. Janssen, A. Sharpanskykh, R. Curran, and K. G. Langendoen. AATOM - An Agent-based Airport Terminal Operations Model simulator. *SummerSim '19: Proceedings of the 2019 Summer Simulation Conference*, 2019. Publisher: ACM DL.

[7] Stef Janssen, Anne-Nynke Blok, and Arthur Knol. AATOM - An Agent-based Airport Terminal Operations Model. Technical report, 2018.

[8] Angel A. Juan, Javier Faulin, Scott E. Grasman, Markus Rabe, and Gonçalo Figueira. A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems. *Operations Research Perspectives*, 2:62–72, December 2015. doi:10.1016/j.orp.2015.03.001.

[9] Rotterdam The Hague Airport. Ons verhaal in feiten en cijfers. Technical report, 2021. URL: https://www.rotterdamthehagueairport.nl/wp-content/uploads/Ons-verhaal-in-feiten-en-cijfers-2021.pdf.

[10] Nursyuhada Taufik and Mohd Hafiz Hanafiah. Airport passengers' adoption behaviour towards self-check-in Kiosk Services: the roles of perceived ease of use, perceived usefulness and need for human interaction. *Heliyon*, 5(12):e02960, December 2019. doi:10.1016/j.heliyon.2019.e02960.

[11] Paul Pao-Yen Wu and Kerrie Mengersen. A review of models and model usage scenarios for an airport complex system. *Transportation Research Part A: Policy and Practice*, 47:124–140, January 2013. doi:10.1016/j.tra.2012.10.015.

II

Literature Study
previously graded under AE4020

1

Introduction

As of 2022, the airport services market size was valued at $97.87 billion and is projected to grow to $290.23 billion by 2029 [**?**]. These figures illustrate the immense size and growing potential of the aviation industry. This growth however also poses problems in terms of capacity of airports. A simple solution would be for airports to just keep expanding and add more terminals for example. More often than not, this is not possible resulting in the need for creating more capacity without expanding the facility. Development in self-service technology, planning and operations could be the answer for creating more capacity.

A request was made by Rotterdam The Hague Airport (RTHA) for help with finding a configuration for self-service kiosks. RTHA is the third airport of the Netherlands and serves roughly 2 million passenger every year [34]. In order to effectively evaluate the performance of the self-service kiosks by means of simulations, recommendations can be made to allow for a smooth and efficient implementations of these kiosks. The simulations will aid in determining the configuration for two types of kiosks: bag tag kiosks (before security and at the gate) and drop-off kiosks.

The aim of this literature study is to establish a good understanding of how an optimal configuration can be found for self-service baggage check-in kiosks using simulation and optimization. To achieve this understanding in chapter 2 an introduction to airport operations airport terminal processes will be given. Next, in chapter 3, the check-in process along with the state of the art of self-service technology will be discussed. To investigate how simulation and optimization can be used for the problem introduced here, in chapter 4 and chapter 5 the state of the art of airport terminal modeling and simulation optimization is established and discussed. Following this, the simulation technique and model that followed from the airport terminal review is presented in chapter 6. To conclude this literature study, a research proposal and methodology is given in chapter 7 and chapter 8.

2

Airport Operations

To start investigating how the self-service kiosks introduced in chapter 1 can be optimally configured, first a good understanding of airport operations must be established. In this chapter, first the steps of the airport terminal design process, the elements part of an airport terminal and the stakeholders will be discussed. Next, the processes in a terminal will be discussed. Last, new trends in airport terminal design in order to increase the capacity and efficiency will be looked at.

2.1. Airport Terminal Design

The airport terminal acts as bridge between the landside, where passengers arrive via personal or public transport, and the airside. A graphical representation of the airport terminal, landside and airside can be found in Figure 2.1. In this section the factors that must taken into account while designing an airport as well as the components of which an terminal exist will be discussed.

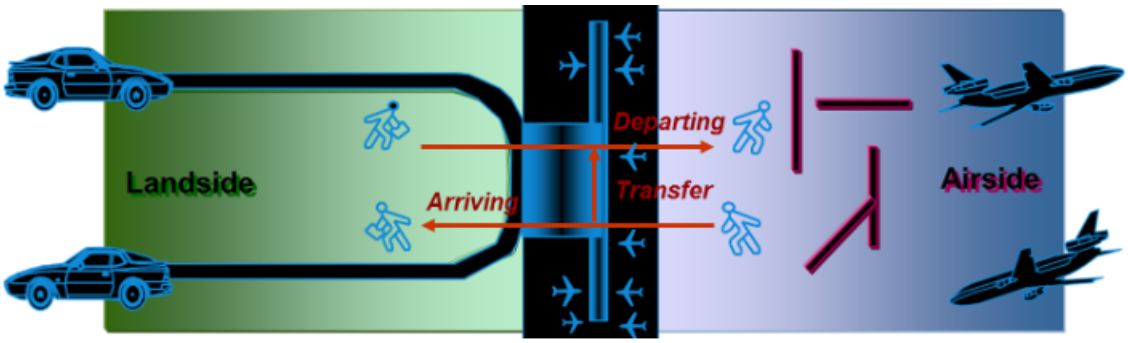


Figure 2.1: The terminal is the bridge between the landside and airside [33]

2.1.1. Airport terminal design process

As terminals are very expensive to build, errors during the design could be very costly. Therefore, design processes and standards have been developed to ensure a design that adequately designed in terms of capacity and cost. An example of a typical design process is given by Odoni and de Neufville [32]:

- Forecasting traffic levels for peak hours
- Specification of Level of Service (LoS) standards
- Flow analysis and determination of servers and space requirement
- Configuration of servers and space

The design process mostly comes down to determining the demand, then specifying the Level of Service (LoS), determining server and space requirements and configuring the server and space. The LoS refers to

the amount of space per passenger and is defined by the International Air Transport Association (IATA).

In addition to high costs when a terminal is under- or over-designed, it could lead to efficiency problems which cause delays and insufficient LoS. According to Schultz and Fricke [35], passenger and baggage handling make up 8% of the causes of delays for example.

2.1.2. Airport components

As mentioned in the previous section, the last step of the airport terminal design process is the configuration of the servers and space. To determine an optimal configuration, the components of which an airport terminal consists must be well understood. Below, an overview and description of the most important airport components is given as described by Kazda and Caves [17].

Landside kerb

The landside kerb is used by cars, taxis and other road vehicles and are often separated by kerbs. Surveillance is necessary to ensure that cars do not stay longer than permitted while dropping off passengers.

The ticketing lobby

The ticketing lobby is part of the check-in concourse and consists of queues, a counter and space behind the counter for the staff. Tickets can be bought here.

The check-in concourse

The check-in concourse is made up of elements part of the check-in process. This includes check-in and/or drop-off counters. It is also becoming more common to also provide a self-service check-in and baggage drop-off option.

The check-in counters

The check-in counters are located in the check-in concourse and are the full service option for passengers to check-in and check-in baggage.

The out-going baggage handling system

The out-going baggage handling system is often installed under the floor or out of sight of passengers. The system ensures that baggage arrives at the correct aircraft after it is checked in by the passenger. This system has the baggage that is checked-in at the full-service and self-service check-in as input.

Outbound passport control

Here the passports of passengers are checked and if all is good, the passenger can continue to the security screening.

Security screening of passengers

The security checkpoint where the passengers are screened come in many different configurations. They almost always consist of the following elements: a walk-though detection device, an X-ray machine for hand luggage checking and space for manual searches.

The departure lounge

Here passengers can wait for the flight to start boarding.

Retail facilities

These are stores in the terminal and act as entertainment and are an extra form of revenue for the airport. The activities that are not directly related to the airport terminal such as visiting these retail facilities are also referred to as discretionary activities.

2.1.3. Airport Stakeholders

When designing an airport terminal, many stakeholders are involved in the process. In Figure 2.2, a graphical representation can be found of all the stakeholders [24].

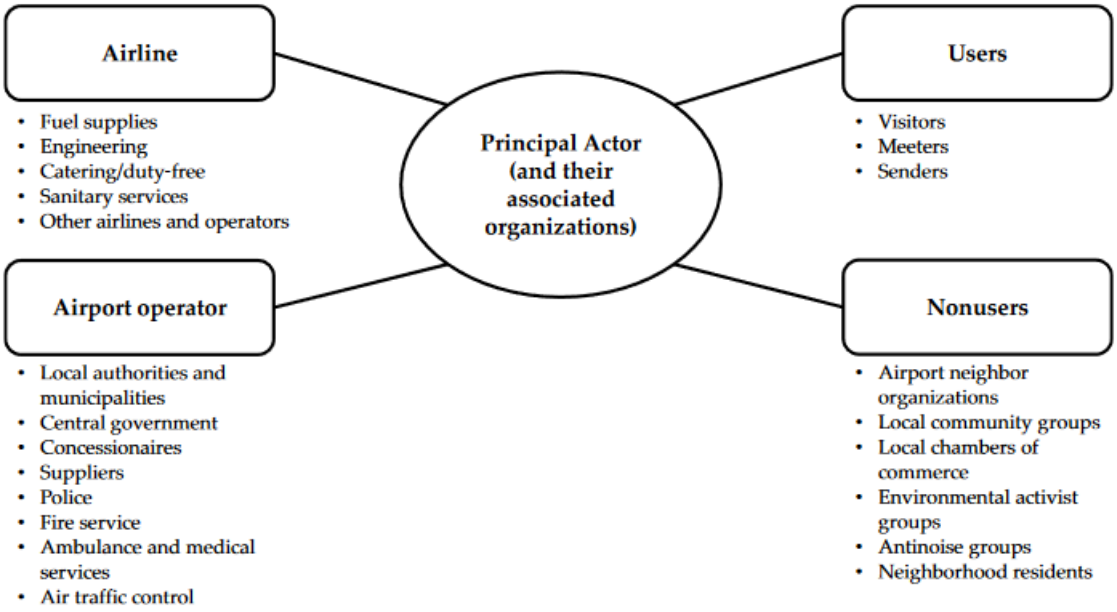


Figure 2.2: Graphical representation of the stakeholders of an airport

The airport manager is in charge of managing the stakeholders. Airport management can either be a privatized organization, a governmental body, or a mix of the two. This illustrates that many parties are involved in the decision making process at airports.

2.2. Airport Terminal Activities

Next to the design process and the airport terminal components it is important to understand the airport terminal activities. A distinction can be made between baggage and passenger handling and departing, transferring and arriving passengers. For this research, only the departing passenger process and the corresponding baggage process will be discussed as this is the only moment when self-service kiosks will be used.

Check in

For the traditional passenger check in process the air ticket and passport/ID should be submitted at the counter. Nowadays only the passport/ID is sufficient to allow the passenger to be checked-in and to hand over the boarding pass. Additionally, the luggage that has to be checked-in is weighed and, if the weight limit is not exceeded, transported away by the baggage handling system. When the weight limit is exceeded, an additional fee needs to be payed. Next, a receipt is attached to the luggage with the weight and destination on it. To speed up the process, automatic self service check-in kiosks are becoming more and more popular which will be discussed more in depth in chapter 3.

Security checkpoint

At security, passengers and their cabin luggage are checked. After security, the passengers will continue onto the airside of the terminal and go the gate when it is time to board the aircraft.

Passport control

During the passport control, the identity of the passenger is checked and it is made sure that no illegal activities take place.

Boarding the aircraft

During boarding, the boarding pass and ID of the passenger is checked one final time and most of the time, the passenger can then board the aircraft. Sometimes however, the operator responsible for boarding can ask the passenger to check-in the cabin luggage such due to a lack of cabin space. When this occurs, the boarding process is disrupted which can cause problems in terms of delays.

3

Check-In

The check-in process of which self-service kiosks are part of is incorporated in the research field introduced in chapter 1. Therefore, it is important to have a good understanding of the complete check-in process and its components. In this chapter first the design of current full-service check-in options will be discussed. Next, the self-check-in process along with the state-of-the-art of the kiosks will be discussed. Last, the next generation of self-service kiosks and the use of bag tag kiosks at the gate will be discussed.

3.1. Check-In Design

The check-in counters can be placed at public places such as train stations or in car parks but they are most commonly situated after the entry of the terminal [17]. The check-in must accommodate conveyor belts in addition to desks which results in different configurations being available. Some common configurations of check-in counters will be discussed below.

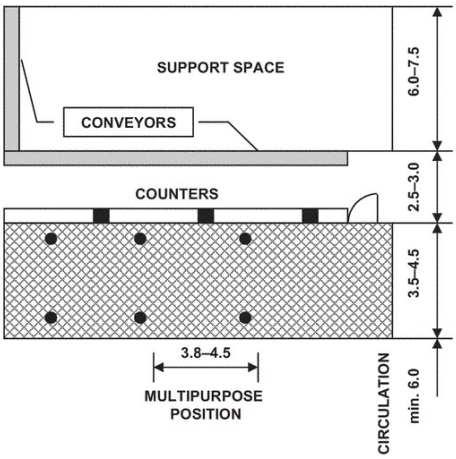


Figure 3.1: Linear facing check-in type [17]

In Figure 3.1, a linear facing type is shown which allows passengers to move through the line of desks after checking in. This lay-out is the most common configuration.

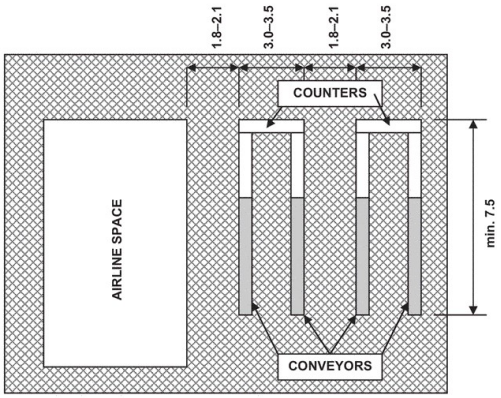


Figure 3.2: Flow-through check-in type[17]

Figure 3.2 shows the flow-through lay-out which allows the passengers to pass through to security away from the queue.

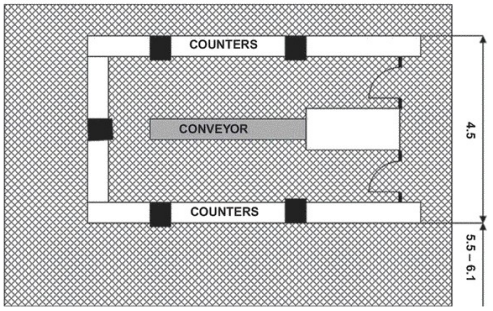


Figure 3.3: Island configuration check-in type [17]

The final lay-out option is the island configuration and is shown in Figure 3.3. For this type, the counters wrap around the belts which run in the direction of the passenger flow.

The flow-through and island configuration allow passengers to go to security more efficient and offers a more logical path. This results in less confusion and a better experience for passengers. The International Air Transport Association (IATA) [12] therefore recommends the island configuration.

3.2. Self-Service Check-In

Normal check-in remains important, it is estimated however that 33-50% of check-in counters will be self-service of which half is designed for passengers with baggage [12]. Additionally, regular check-in can create dissatisfaction when waiting times increase [47]. A solution to this is the use of more self-service devices during check-in. In 2007 the IATA launched the Fast Travel Program. In this program, the three most important aspects are considered to be self-check-in, self-flight rebooking and self-bag tagging [1].

The IATA [12] mentions a problem which could occur when airports decide to operate self-service kiosks. Operating self-service kiosks could result to a requirement to operate more regular check-in desks. To counteract this, the IATA introduced the concept of common use terminal equipment (CUTE) was introduced. CUTE enables airlines to share check-in counters and kiosks to optimally utilize the space in the terminal. The IATA also encourages the use of CUTE in terminal design. Therefore, self-service kiosks should be shared and be able to be used for multiple airlines.

Determining the location of these self-service kiosks is very important since locating the kiosks is closely related to the facility layout problem. Problems that fall under the facility layout problem are characterized by

the objective to optimally locate the facilities that make up a system [37]. Therefore, when implementing such kiosks is important to optimally place them and preferable first test configurations by means of simulation.

3.2.1. Self Check-In Options

Nowadays, a passenger has multiple choices in terms of check-in methods. A graphical representation of the different check-in methods at Zurich airport can be found in Figure 3.4.

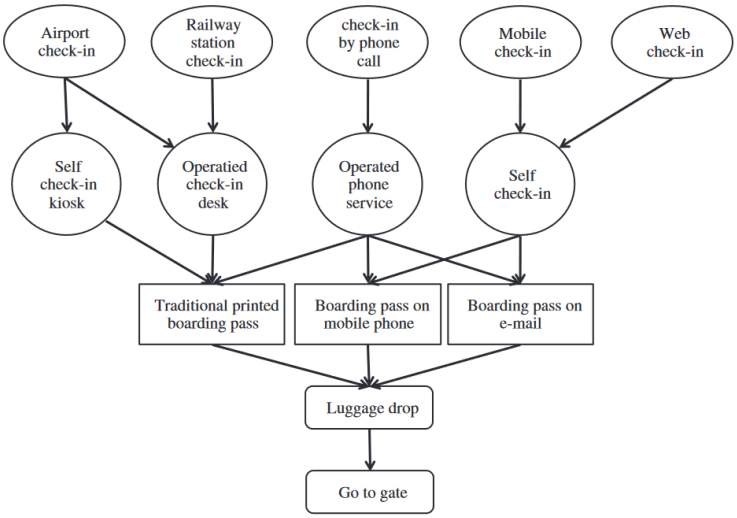


Figure 3.4: Graphical representation of the different check-in methods at Zurich airport [47]

As can be seen in the figure above, the route for a passenger to be checked-in via self-service facilities can vary. The process can be completed using a kiosk for checking-in and dropping off which can be done on the airport or at a railway station for example. The passenger can also choose to check-in by phone call, mobile check-in or web check-in. When this is the case, the passenger does not have to use a self-service kiosk anymore for checking-in, but may need it for dropping off baggage. This can be done at separate baggage drop kiosks.

3.2.2. State of the Art Self-Service Kiosks

Current self-service kiosks can completely eliminate the need for airport personnel. Below, the steps the passenger has to go through on a self-service kiosk are given according to IATA [12].

- Passenger approaches self service check-in kiosk with physical ticket or e-ticket details.
- Passenger inserts ticket or e-ticket details.
- Biometric passenger data captured. (optional)
- Biometric data analyzed. (optional)
- Airline/passenger profiling questions asked and passengers answers given.
- Optional passenger places passport on the screen.
- Passport data validated from central database.
- Baggage license plate label printed and affixed to baggage by passenger.
- Passenger asked to place baggage item(s) onto conveyor system. (optional)
- Confirmation of number of pieces of luggage. If 0-1 items of luggage per passenger step 7 else step 5.
- Process ends for baggage.

The steps above describe the entire self check-in process. In some cases, the passenger is already checked-in online or via mobile as is discussed in the previous section. When a passenger is already checked-in, the passenger must complete the following process to check-in their baggage.

- Passenger approaches self service bag tag/drop-off kiosk with physical ticket or e-ticket details.
- Passenger inserts boarding pass or e-boarding pass.
- Airline/passenger profiling questions asked and passengers answers given.
- Baggage license plate label printed and affixed to baggage by passenger.
- Optional passenger asked to place baggage item(s) onto conveyor system.
- Confirmation of number of pieces of luggage. If 0-1 items of luggage per passenger step 7 else step 5.
- Process end for baggage.

Some kiosks have extra features such payment options when bags are overweight and extra luggage buy options. Additionally, the process of receiving the bag tag and dropping off the baggage can separated by means of a separate bag tag and drop-off kiosk.

3.3. Next Generation Self-Service Kiosks

Current self-service kiosks are designed to make the check-in process completely automatic and are part of the check-in concourse. This has as a result that airports design the self-service kiosks to maximize perceived usefulness instead perceived ease of use [40].

3.3.1. Checked luggage check-in kiosks

Bagchain is a manufacturer of self-service kiosks and has as its goal to make the self-service baggage drop-off easier and more straightforward [4]. By including no extra features other than printing the bag tag at the bag tag printer and dropping off luggage at the drop-off kiosk. Bagchain aims for the self-service kiosks to have a high degree of perceived ease of use. In cases that the bag tag can not be printed, when the passenger has not bought any checked luggage for example, the passenger will be redirected to a service point. This limits the average time spent at a kiosk. Additionally, Bagchain offers mobile kiosks such that these can be placed strategically and are movable.

3.3.2. Cabin luggage check-in kiosks at gate

Next to self-service options before security, Bagchain has proposed to place bag tag kiosks at the gate to check-in cabin luggage if necessary. Bagchain [4] estimates that manually checking in cabin luggage can take between 60 and 90 seconds which can be drastically decreased by using a kiosk.

3.4. Discussion of check-in process

In light of the problem introduced in chapter 1, the following conclusions can be drawn from the literature presented in this chapter. As the trend is clearly heading towards more self-service options in the airport, it is important to extensively investigate the implementation of them. Passenger needs do not indicate more functions for example but want an easier to use kiosk. Additionally, determining the location of the kiosks must carefully be done to ensure optimal performance of the terminal. In the following two chapters, the possible methods that could aid in the efficient implementation of these kiosks will be discussed.

4

Modeling Airport Terminal Operations

In order to investigate the impact of implementing new elements in an airport terminal such as the Bagchain kiosks, modeling the airport terminal first is preferred [25]. When developing a model it is important to take into account the reusability and extensibility in order to limit reprogramming [32]. Therefore, in this chapter, the state of the art of airport terminal modeling will be analyzed and discussed.

Using the distinction between models as proposed by Wu et al. [48], this chapter will have the following outline. First, a general approach to modeling airport terminal processes will be given. Next, some early models and their capabilities will be discussed after which models will be discussed with a similar purpose as will be relevant for this research.

4.1. Approach to Modeling Airport Terminal Processes

In order to effectively develop an airport terminal model, a method is proposed by [48]. This method relies on first defining the Concept of Operations (CONOPS). As the majority of airport terminal models are developed with software, the IEEE Information Technology CONOPS Std 1362-1998 (R2007) [2] is used. This CONOPS standard provides guidelines on how to develop usage scenarios and how requirements can be drawn from them. From this CONOPS, a set of criteria is determined which can be used when designing a model [48]:

- Modeling objective
- Operational policy and constraints
- Model capabilities
- Users

When the CONOPS for a model is defined, one must investigate what data is required as input in terms of traffic forecast. An extensive list is provided by Tosic [41] and some examples are: design period number of passengers (total) and arrival/departure ratio or volume for different passenger types, etc. This illustrates large amount of data required to accurately perform simulations.

4.2. Preliminary Requirements

In order to make an accurate assessment of the state of the art of airport terminal models which are relevant for this research, a preliminary set of requirements will be stated in this section. To generate this set of requirements the problem stated in chapter 1 must be analyzed. The objective is to find an optimal configuration for self-service kiosks in an airport terminal. Optimal can be defined with respect to several performance metrics such as queue length and waiting times for passengers. Therefore it must be possible to extract this data from the model. Additionally, since the effect of placing the self-service kiosks in the terminal is important, the airport terminal model must be holistic. Visualizing passenger flows could also aid in investigating the effect of implementing the self-service kiosks on the passenger flow dynamics. To achieve is however, passengers must be modeled individually requiring a high level of detail [48]. A final requirement can be defined for the

modeling language. Many systems both have qualitative and quantitative aspects. This is also the case for an airport terminal. Quantitative aspects are number based, for example time or number of passengers in an airport terminal. Qualitative aspects are for example reasoning and coordination of passengers. The modeling method and language must therefore be able to cope with a combination of qualitative and quantitative aspects.

4.3. State of the Art of Modeling Airport Terminal Processes

To investigate what modeling techniques are common for certain applications, the state of the art needs to established. To provide a complete overview, first early airport terminal models and the modeling techniques they rely on will be explored. Next, models which have a similar goal as the proposed research here will be discussed along with the used techniques.

4.3.1. Early Airport Terminal Processes Models

Early models were predominantly capacity models and had the purpose of determining whether a terminal could handle (future) passenger forecasts. In this section, the development of these model type is discussed.

Queue modeling

One of the first efforts made to model a process in an airport terminal made use of a queue model. This was a deterministic queue model on which many other models were later based [31]. This model elaborated upon graphically representing cumulative arrival and departure profile from the service facility. Let $A(t)$ and $D(t)$ be the arriving and departing passengers from a facility respectively for $(0, t)$. The amount of passengers in a queue at time t can then be as depicted in Equation 4.1.

$$Q(t) = A(t) - D(t) \tag{4.1}$$

This relation can also be visualized using cumulative arrival and departure profile mentioned earlier. An example graph can be found in Figure 4.1.

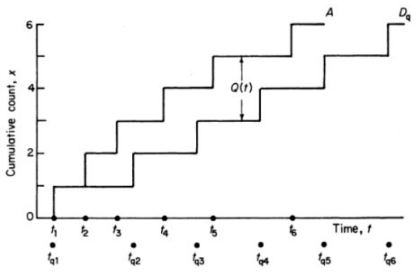


Figure 4.1: Example cumulative arrival and departure profile [31]

This model however fails to generate important metrics such as minimum and maximum waiting times for passengers. It also fails to consider uncertainty in the arrival problem of passengers.

These shortcomings can be eliminated by using a stochastic queuing model. Such a model does not assume a cumulative arrival function, but instead it uses multiple arrival functions per flight along with probabilities. A stochastic queuing model for evaluating passenger process times and resource allocation was presented by Bevilacqua and Ciarapica [6]. In this research a simulation run is performed by using Monte Carlo simulations until a steady-state is reached. This steady-state is then evaluated for common check-in configuration and a dedicated carrier check-in configuration.

To extend these models and create a more holistic airport terminal simulation, different facilities can be linked to create a complete system. Models such as described by Dunlay and Park [7] consider such linked facilities where the departure profile of one facility is used as the arrival profile of the next facility. A certain offset is taken into account to include walking from one facility to the next.

Statistical approach

Another approach taken for the development of capacity models is a statistical approach. Kim et al. [19] developed a mathematical model which used probability density functions on dwell times to estimate the number of passengers arriving at an airport terminal at various times. By making this probability function time dependent the dwell times could be accurately estimated. This let to this research providing significant insights in how dwell times differ for different airports.

Similarly, a method was proposed for analytically approximating maximum passenger delay for airport corridors and service facilities by means of a multicommodity flow model [38]. Relations were derived between flow rates and width of corridors. Multiple objective functions could applied to this system [38]. An example of a objective could be to minimize the maximum time a passenger spends in the flow model (terminal). This model is considered to be holistic since it has linked facilities, similar to the model proposed by Dunlay and Park [7].

4.3.2. Operational Planning and Design Models

Similar to the capacity models, operational planning and design models evaluate metrics such average queue length and passenger waiting times. The goal and model capabilities differ however from the previously discussed models. These types of models are used to investigate what the impact is of changing, adding or removing resources such that the daily airport operations can be evaluated. This goal requires a higher level of detail and therefore the previously discussed models can not be used for this purpose. In this section, the development of this model type is analyzed and discussed.

Agent based models

An early effort to develop a operational planning and design model was made by means of a discrete event agent based model [8]. This model evaluates passenger flows in an airport terminal while taking into account flight schedules, service rates, resources and passenger characteristics. To accomplish this, a network program called CARN (Conditional Analysis for Random Networks) was created. The passengers move through the nodes of the network as if they are facilities with each node having a certain service rate distribution. As a result, it was found that passenger delays resulting in passengers missing flights was well correlated to congestion. Based on the method proposed Eilon and Mathewson [8] a great effort was made to develop Airport Landside Simulation Model (ALSIM) reviewed by Tosic [41]. Additionally, Lui et al. [27] also adopts the method proposed by Eilon and Mathewson [8]. The goal of this model was to evaluate physical and operational plans to ensure that the New York JFK could accommodate a large new type of aircraft.

All these models adopt a macroscopic approach which results in simulations that can be evaluated by the performance of the global system. Therefore, to evaluate the influence of operational and resource changes is for individual passengers, a microscopic approach is necessary. A simulation of Ataturk airport used the ProModel simulation package to take a more microscopic approach [20]. A similar microscopic simulation for Kansai airport [39]. Using the microscopic approach, passenger flows can be visualized as can be seen in Figure 4.2.

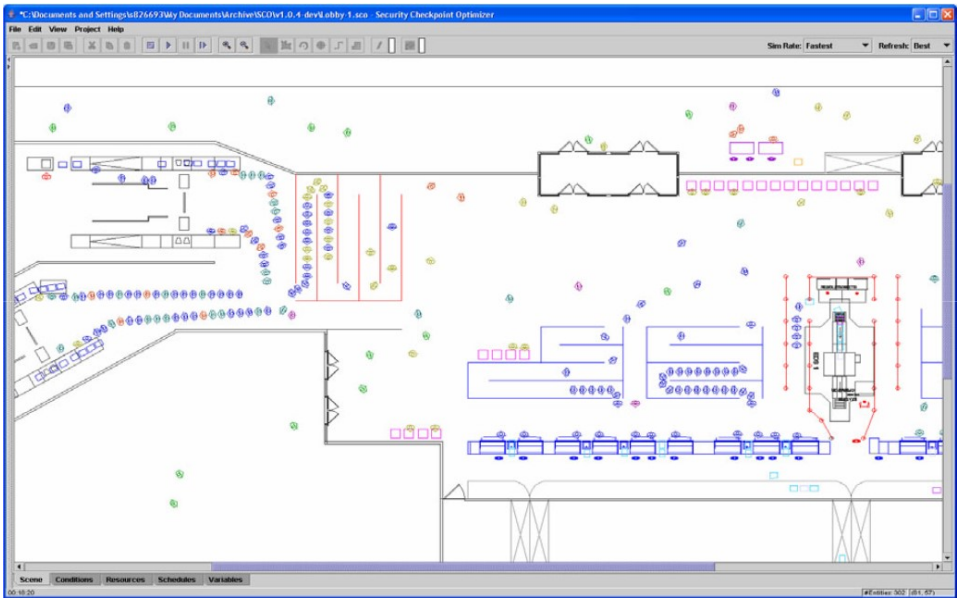


Figure 4.2: Example of visualizing passenger flows [46]

Next to visualizing the passenger flow, which allowed for emergence of flow patterns, the use of agent based modeling for airport terminal process modeling was taken a step further by evaluating different performance metrics separately [46]. This has as an advantage that the performance of facilities can be evaluated separately while modeling the complete terminal.

It is argued that the entire airport terminal process is based on individual passenger behavior [35]. Therefore, in order to optimize this, the individual passenger behavior must be modeled accurately. Schultz and Fricke [35] proposed a stochastic approach to agent based modeling to model the processes in an airport terminal. This stochastic approach was implemented for passenger movements at the operational level and at the tactical level. This allows agents to act with environmental anticipation and enables the capability of passengers making tactical decisions and route finding [35]. The results of the simulation pointed out that there is a need for a significant change of the common flow-oriented design method of airport terminals.

The final model that will be discussed is an agent based model that aims to evaluate different operational policies [45]. The policies that are evaluated concern changing the passenger flows through the entrance of the terminal and setting up self-service check-in kiosks. For this research it was argued that inter-passenger relations did not have to be modeled since only the system behavior was relevant resulting in a simpler model [45]. The passenger flows were visualized with a GUI displayed in Figure 4.3 which aided in the evaluation of the different policies.

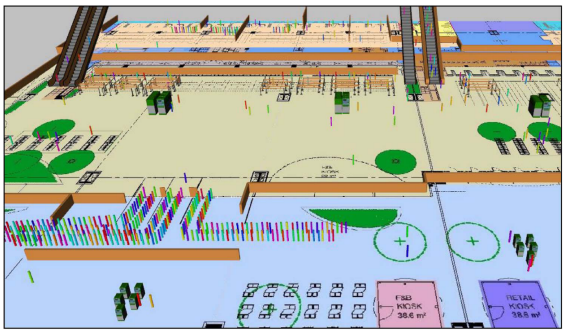


Figure 4.3: Snapshot of the GUI of the model developed by Verma et al. [45]

Other approaches

Next to agent based approaches, other approaches have been explored. Manataki and Zografos [29] have proposed a system dynamics based method for example. The model proposed was used to effectively answer the same question as the models discussed above which includes aiding with making decisions on operational concepts. The method used relies on nodes representing different airport processes and the links between them representing the passenger flows. The nodes or facilities are modeled using the building blocks of system dynamics modeling: stocks, flow and converters. This type of modeling approach has as a main advantage that it can be easily applied to a multitude of airports since (almost) all airports have the same service facilities. Manataki and Zografos [29] therefore argue that not the layout, but the topology or the way the facilities are connected are important.

Another approach that has been taken to evaluate the airport terminal is the use of a fuzzy model.Kyildi and Karashin [23] have proposed a model which uses this approach to perform an analysis for the check-in unit for a airport. The fuzzy network establishes the link between performance metrics and the variables on which they are dependent using IF-THEN statements. Since a relationship is established, one can easily trace back a delay to the main contributing factor.

4.4. Discussion of Model Overview

Taking into account the aim of this research, an operational planning and design type and therefore the models and techniques used for that purpose are most relevant. Wu and Mengersen [48] concluded that the main differences between operational planning and design models and capacity planning type models are the detail level which results in the following differences:

- Individual passenger versus passenger group interactions
- Spatial detail and passenger movement interaction differences
- Visualization differences

As discussed earlier, Verma et al. [45] has developed model with a similar objective as this research. However, contrary to that research, there must be the opportunity to investigate the effect of placing the kiosks at different on a passenger level. Therefore, to investigate this as accurately as possible, individual passenger interactions must be modeled and visualized. These requirements lead to an agent based model with stochastic approach and microscopic detail level being favorable. The main downside to this high level of detail approach is that it can provide misleading results which requires detailed validation to be prevented.

This approach must be paired with modeling language that satisfies the earlier defined requirement. This requirement stated that the language must cope with both qualitative and quantitative aspects. An example of such a language is Temporal Trace Language (TTL) [36].

Developing a model that satisfies all requirements from scratch is a lengthy and complex process when carried out properly. Therefore it is out of the scope of this research. Expert validation and comparisons between predicted and recorded traffic patterns is for example needed [48]. Therefore it is recommended to extend an existing easily extendable model which is available. In chapter 6 a detailed overview will be given of such a model and how it can be extended for the purpose of this research.

5

Simulation Optimization

Optimization of a certain problem can be performed in many different ways depending on the type of model. When finding an optimal solution for a mathematical model for example, an objective function can easily be evaluated and derivative information is available which allows for efficient exploration and exploitation of the parameter space. For an agent-based model with a high degree of stochasticity as will be the case for this research, this derivative information may not be available or hard to estimate [3]. Therefore, a derivative free optimization or black-box optimization method needs to be used. These methods are also referred to as simulation optimization and will be discussed in depth in this chapter. A general introduction to simulation optimization will first be given after which different simulation optimization techniques will be discussed.

Simulation optimization is often applied to discrete event simulations and systems of stochastic nonlinear and/or differential equations [3]. To get a better understanding of the simulation optimization problem it can be categorized by the following categories [43]:

- Continuous vs. discrete: whether or not the variables considered by the problem are discrete or continuous.
- Constrained vs. unconstrained: whether or not constraints apply to the decision variables.
- Mono-objective vs. multi-objective: whether the problem considered aims to optimize a single or multiple objective functions. Methods exist that allow for rewriting multiple objective functions to one weighted objective function. These methods are Pareto and naive methods [28].
- Status vs. dynamic: For static optimization the possible solutions are evaluated at the end of each iteration. For Dynamic optimization the possible solutions can be tuned during the evaluation process.

Different simulation optimization approaches have different application. In Figure 5.1, different approaches and their characteristics can be found.

Algorithm class	Discrete	Continuous	Local	Global
Ranking and selection	×			×
Metaheuristics	×	×		×
Response surface methodology		×	×	×
Gradient-based methods		×	×	
Direct search	×	×	×	
Model-based methods	×	×	×	×
Lipschitzian optimization		×		×

Figure 5.1: Different simulation optimization approaches and their characteristics [3]

For this literature review, gradient-based methods and Lipschitizian will not be discussed due to these methods only having the ability to solve continuous optimization problems and requiring derivative information

(or an estimation of it). Response surface methodology however will be covered since, with some modifications, response surface methodology can be used for discrete simulation optimization problems [5].

5.1. Metaheuristics

In order to provide a comprehensive overview of the metaheuristic algorithms used for simulation optimization, a distinction will be made between finite and large or infinite parameter spaces. These metaheuristic algorithms are often the driving force behind simulation optimization. Additionally, a simheuristic approach to simulation optimization will be discussed in this section.

5.1.1. Finite Parameter Spaces

For finite parameter spaces the number of solutions is relatively small and fixed. The main goal is to structure or rank the different simulation runs [3]. Due to the finite parameter space, there is no emphasis on exploration and solutions simply need to be compared to find the optimal one.

An optimization technique which can be used when dealing with a relatively small and finite parameter space is ranking and selection. A review of different ranking and selection procedures was done by Kim et al. [18]. The basic principle behind ranking and selection is to minimize the number of simulation runs to select while trying to guarantee that the selected solution is the best with a certain probability.

5.1.2. Large or Infinite Parameter Spaces

When the parameter space becomes larger or infinite it is not sufficient to essentially compare all solutions to find the best solution as was done in ranking and selection. Methods that can be used in case of a large or infinite parameter space must strike a balance between exploration and exploitation. The algorithm must explore different areas while still exploiting promising areas. This must be done while being efficient resulting in a not excessive computational burden.

Algorithms

The algorithms that fall under metaheurisics for a large or infinite parameter space are simulated annealing, ant colony optimization and tabu search. An extensive review of metaheuristic optimization methods was done by van der Horst [43]. The results from this overview can be found in Table 5.1.

Table 5.1: Overview of metaheuristic techniques and their application [43].

Metaheuristic technique	Application	Algorithms
Single solution based	Continuous and discrete optimization, Combinatorial optimization	Simulated annealing, Tabu search, Variable neighbourhood search, Iterated local search, GRASP method
Evolutionary computation	Combinatorial optimization, Constrained optimization, Multi and single objective optimization, Continuous optimization	Genetic algorithm, Evolution strategy, Differential evolution, Coevolutionary algorithms, Scatter search with path relinking,
Swarm intelligence	Global optimization, Combinatorial optimization	Particle swarm optimization, Ant colony optimization

The single solution based algorithms rely on a starting point (an initial solution) and moving away from this starting point. The algorithms that fall under single solution based technique can be applied to both continuous and discrete problems. The next technique displayed in Table 5.1 is evolutionary computation which also falls under model-based methods. The search space is explored and exploited using pairing, mutating, crossing and selection which are evolutionary principles [43]. This technique can be used for both multi and single objective optimization. Moreover, this technique is mostly used on continuous and constrained optimization problems. The final technique is swarm intelligence which also falls under model-based methods and metaheuristics just as evolutionary computation. This technique uses principles which can be found in nature and specifically large insect swarms and distincts itself from evolutionary computation by not resam-

pling [43]. This technique is able to find a global optimum. All discussed metaheuristic techniques can be applied to combinatorial problems.

5.1.3. Simheuristics

When choosing a metaheuristic algorithm the computational power required must be taken into account. It could for example be beneficial to choose a less accurate algorithm which requires less computational power if the problem is very complex. This is especially the case when the simulation model contains stochasticity. To further decrease the computational power required Juan et al. [16] proposed a method to reduce this computational burden with a method called simheuristics. Additionally, using a simheuristic framework allows for deterministic optimization techniques to be used for optimization problems where stochasticity is involved.

This method can be paired with many metaheuristic optimization algorithms and relies on using a simplified (deterministic model). The metaheuristic algorithm is used to find a good solutions when evaluating the objective function with results from the deterministic model. It is argued that a solution that performs well when evaluated with the deterministic model, will also be a good solution when using the stochastic model [16]. A pool of 'elite' solutions can be formed using a metaheuristic algorithm from the determinstic model until a stopping criterion is reached. This pool of 'elite' solutions is then intensively evaluated in order accurately rerank the solutions [16]. In Figure 5.2, schematic representation of the simheuristic approach can be found.

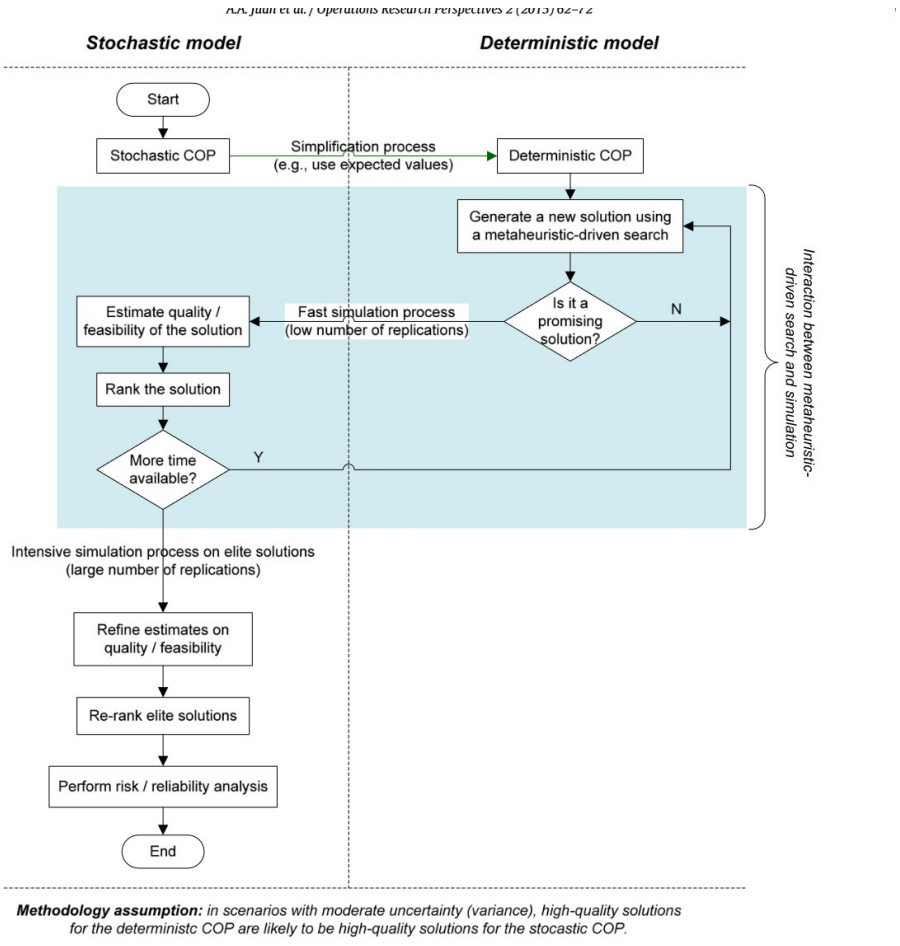


Figure 5.2: Schematic representation of simheuristic simulation optimization approach [16]

Simheuristics have been applied by Grasas et al. [9] in order to use iterated local search on a stochastic combinatorial problem. Usually, this metaheuristic is only applicable to deterministic problems but with simheuristics it could be extended to a stochastic problem. The research presents several applications of the

framework such as: vehicle routing, scheduling and inventory routing [9].

5.2. Other Simulation Optimization Techniques

Next to the metaheuristic simulation optimization approaches discussed earlier, there are several other simulation optimization techniques. In this section response surface methodology, model-based methods and direct search methods will be discussed.

5.2.1. Response Surface Methodology

Response surface methodology is also a method to use for simulation optimization. This method is typically useful in continuous optimization problems but can be applied to discrete optimization problems as well [5]. This method relies on approximating the underlying simulation by focusing on the input-output relationship [3]. This is also known as meta or surrogate modeling. Pseudocode for a classic response surface methodology algorithm can be found in Figure 5.3.

Algorithm 2 Basic RSM procedure

Require: Initial region of approximation $\mathcal{X}$, choice of regression surface r
1: **while** under simulation budget and not converged **do**
2: Perform a design of experiments in relevant region, using k data points
3: $t_i \leftarrow \text{simulate}(x_i), \quad i = \{1, \ldots, k\}$ {Evaluate noisy function $f(x_i, \omega)$}
4: $\lambda^* \leftarrow \arg\min_\lambda \sum (t_i - r(x_i, \lambda))^2$ {Fit regression surface r through points using squared loss function}
5: $x^* \leftarrow \{\arg\min_x r(x, \lambda^*) : x \in \mathcal{X}\}$ {Optimize surface}
6: Update set of available data points and region of approximation
7: **end while**

Figure 5.3: Psuedocode for a classic response surface methodology [3]

A response surface methodology can be applied in two ways for simulation optimization approaches [3]:

- Multiple surrogate models are built for certain regions which are then used to guide the search.
- One surrogate model is built for the entire parameter space which is then used to pick solutions.

5.2.2. Model-based methods

Model-based methods rely on building a probability distribution for the solution space. This is done to guide the search through the parameter space for an optimal solution [3]. An example of such a method is ant colony optimization discussed in subsection 5.1.2. This method both falls under model-based methods and metaheuristics. This algorithm relies on the real-world behavior of ants and their pheromones. When a path or solution is used many times, pheromones build up resulting in this solution being more likely to be used. The pheromone build up results in the aforementioned probability distribution. Pseudo code for an ant colony optimization algorithm can be found in Figure 5.4.

Algorithm 9 Ant Colony Optimisation

$C \leftarrow (C_1, ..., C_n)$ components
$popsize \leftarrow$ number of trails to build at once

$\overrightarrow{p} \leftarrow [p_1, ..., p_n]$ ▷ Pheromones of components, initially zero
$Best \leftarrow \square$
repeat
    $P \leftarrow popsize$ trails built by iteratively selecting components based on pheromones and costs or values
    for $P_i \in P$ **do**
        $P_i \leftarrow$ optionally perform local search P_i
        if $Best = \square$ or $fitness(P_i) > fitness(Best)$ **then**
            $Best \leftarrow P_i$
        end if
    end for
    Update $\overrightarrow{p}$ for components based on the fitness results for each $P_i \in P$
until $Best$ is best solution or time-out

Figure 5.4: Psuedocode for ant colony optimization algorithm [43]

Other model based methods fall under estimation of distribution algorithms which belong to the evolutionary computation field [3]. For this method, instead of generating a new solution from a genetic operator, a new solution is generated by sampling from the inferred probability solution over the solution space [3].

5.2.3. Direct Search Methods

The last simulation optimization technique that will be discussed in this chapter fall under direct search methods. These methods can be defined as an examination of trial solutions [3]. These trial solutions are generated by a certain strategy. This strategy determines which solution will be evaluated next as a function of earlier results [11]. Direct search methods build on the derivative free optimization ideas which is beneficial to simulation optimization since derivative information is often no available. An extensive review of direct search methods is given by Kolda et al. [22].

The two most popular direct search methods are pattern search and Nelder-Mead simplex method. These methods are very interesting due their simplicity and ability to not be affected by derivative information not being present. Pattern search was elaborated upon by Trosset [42] and more information on the Nelder-Mead simplex method is given by Nelder and Mead [30]. Additionally, the convergence for these methods in a simulation optimization setting is also investigated and described by Lucidi and Sciandrome [26]. Pseudo code for Nelder-Mead simplex method can be found in Figure 5.5.

Algorithm 4 Basic Nelder–Mead simplex procedure for SO

Require: A set of $n-1$ points in the parameter space to form the initial simplex
1: **while** under simulation budget and not converged **do**
2: Generate a new candidate solution, x_i, through simplex centroid reflections, contractions or other means
3: $t_i^{j_i} \leftarrow \text{simulate}(x_i), \quad i = \{i-n+1, \ldots, i\}, \; j_i = \{1, \ldots, N_i\}$ {Evaluate noisy function $f(x, \omega)$ N_i times, where N_i is determined by some sampling scheme}
4: Calculate $\frac{\sum_{j_i} t_i^{j_i}}{N_i}$, or some similar metric to determine which point (i.e., with the highest metric value) should be eliminated
5: **end while**

Figure 5.5: Psuedocode for a Nelder-Mead simplex method [3]

Since the trial solutions are generated by a certain strategy and then compared, this method is difficult to scale and use with large parameter spaces. Also direct search methods can be affected by noise, effective sampling schemes to control the noise are required [3].

5.3. Discussion of Simulation Optimization Methods

None of the simulation optimization methods discussed above is better than the other. Some however will be better for certain applications depending on the simulation optimization problem that needs to be solved. As

discussed earlier, a simulation optimization problem can be categorized using the following categories [43]:

- Continuous vs. discrete
- Constrained vs. unconstrained
- Mono-objective vs. multi-objective
- Static vs. dynamic

Additionally, the computational time must also be taken into account. When complexity is high of the to be solved problem, it could be beneficial to choose a less accurate method which requires less computational power. A less accurate solution for an accurate real world model is more valuable than an optimal solution for a oversimplified model. For a metaheuristic approach, an algorithm can be paired with simheuristics to decrease the computational power required. A preliminary trade-off will be performed in chapter 7 using the results from the review performed in this chapter. This trade-off will determine whether a ranking and selection, metaheuristic, response surface methodology, model-based method or direct search method will be used for this research. A metaheuristic algorithm with a simheuristic framework will also be taken into account for the trade-off.

6

Agent-based Airport Terminal Operations Model

Following from chapter 4, an agent based model with a stochastic and microscopic approach was deemed to be the best approach. Additionally, the usage of the Temporal Trace Language was a viable option. Developing such a model from scratch however is very complex. Therefore it is preferred to extend an existing and proven model. The Agent-Based Airport Terminal (AATOM) Operations Model simulator is such a model which is extendable and available. This model utilizes the LEADSTO language which can be defined as sub-language of TTL [36]. In this chapter, an introduction will be given on agent-based modeling along with the advantages and disadvantages. An introduction to AATOM and its architecture will be given.

6.1. Introduction to Agent Based Modeling

There is a large difference between macroscopic modeling approaches and microscopic agent based modeling approaches. Instead of describing the overall global system characteristics, the global system system characteristics are generated from the actions of the individual agents [21]. These agents are the active components or decision makers. In this section, first the basic principle and architecture utilized by agent based models will be discussed. After that, the advantages and disadvantages of agent based modeling will be discussed.

Agent based models almost always use a structure where the following three elements must be defined: the agents, the environment and the interactions [21]. The agents are the active components or decision makers. Heterogeneity of agents is possible when designing these agents meaning that they can have different characteristics such as sex, age, etc. These agents are simulated in an environment which comprises of other elements which are not active, for example resources. In order to accurately simulate real life processes agents must be able to interact with each other and the environment [21]. These interactions result in the overall outcome and carefully designing these is therefore crucial.

The architecture described above comes with its advantages and disadvantages [21]. The bottom-up approach allows for analysis on agent level as well as the macroscopic level instead of just the macroscopic level. Additionally, having the possibility to define all three elements gives a high degree of freedom. This freedom however poses also a problem. It could be unclear what should included in the model and what should be abstracted by a probability distribution [21]. Problems have also been found with reproduction, the large size of a simulation and validation.

6.2. AATOM Architecture

The model deploying the agent modeling approach discussed above is the AATOM simulator developed by Janssen et al. [13]. As this simulator will be used for this research, an introduction to the architecture, agents, environment and interactions will be given in this section.

The three different layers of which the simulator comprises are: the strategic layer, the tactical layer and the operational layer [14]. The strategic layer is the first layer and consists of the goal, belief and reasoning module. The tactical layer takes care of the actuation activities. It consists of a lower level belief module, which is connected to the belief module in the strategic layer, the interpretation module, activity and navigation module. The last layer is the operational layer which interacts with the environment and other agents. This layer therefore has the perception module and actuation module. The exact structure and connections between the different module is visualized in Figure 6.1.

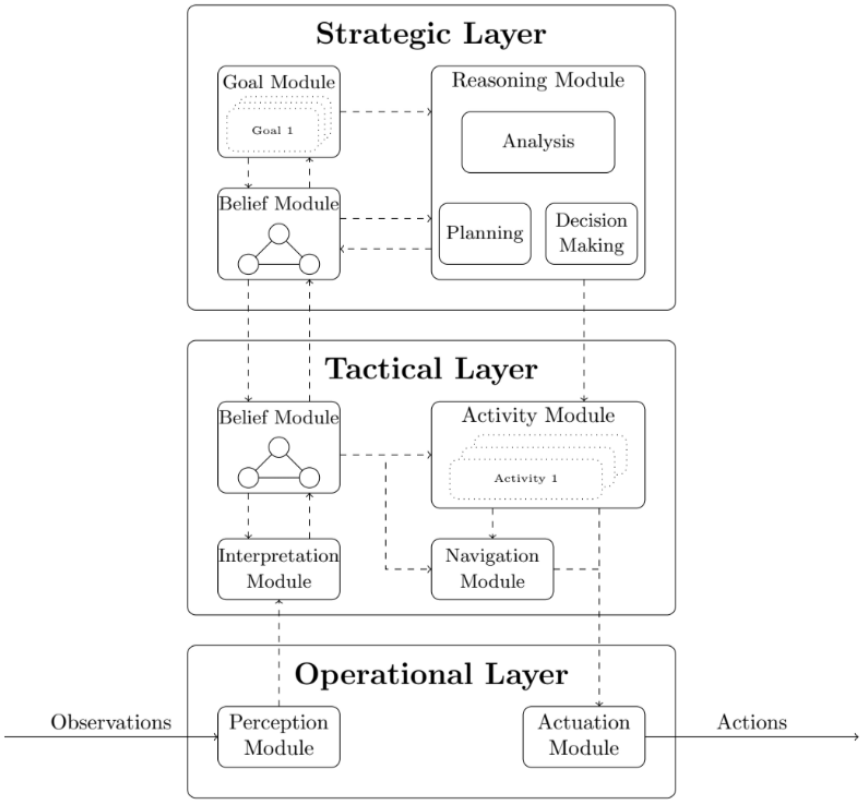


Figure 6.1: Schematic representation of the AATOM architecture[14]

As was discussed in the introduction to agent based modeling, AATOM utilizes agents, environment and interactions. In the following sections these will be discussed in more detail.

6.2.1. Agents

In AATOM the following three agents are used to simulate: passengers, operators and an orchestrating agent [14]. These agents can have characteristics such as inbound or outbound passengers and a certain flight. The same holds for operators which can be security or check-in operators. Next to these characteristics, agents have different attributes which for example is, in the passenger case, a boolean which tracks whether or not a passenger is checked in. The passenger and operator agent are so called human agents. The orchestrating agent does not have to be a human agent and can be implemented to steer other agents and make adjustments in the environment [14].

6.2.2. Environment

The environment also consists of multiple types similar to the agents. These types are: area, flight and physical object [14]. Areas are accessible by agents and are used to separate the terminal in a check-in area, security, etc. Another important environment type is the flight type. This type is used to define different flights that can be linked to passengers, gates and check-in desks among others. The last type of environment is the physical object. The physical object includes walls, desks and all other physical objects which can typically be found in an airport terminal. The self-service kiosks which will be developed for this research will also be of the physical object type for example.

6.2.3. Interactions

The last element but arguably the most important are the interactions between agent and environment and between agents. For AATOM the following interactions are defined between agents and the environment [14]:

- Check-in operator (agent) and flight (environment)
- Security operator (agent) and sensor (environment)

And the following interactions are defined between agents [14]:

- Operator and passenger
- Operator and operator

7

Research Proposal

At this point, an introduction to the many different airport terminal processes and specifically the process check-in has been given. Additionally, the state-of-the-art of airport terminal modeling and simulation optimization techniques was given. From this, a problem definition can be derived along with a research objective, question and sub questions. These will discussed and elaborated upon in this chapter.

7.1. Problem Definition

Following from the literature presented earlier in this report a research objective can formed. In this section first a research gap will be identified after which a research objective will be presented.

From chapter 3 it follows that self-service options are becoming more and more important and is encouraged to be implemented. It has also become clear that user experience and location of the self-service kiosks is critical to these kiosks being a success. As a result, new types of kiosks were developed which are movable and are easier to use making it attractive for airports. To efficiently implement these kiosks, simulations can greatly aid the process of investigating the impact on the performance of the terminal as has been shown in chapter 4. In research done so far, little models emphasize on the influence of self-service kiosks on airport terminal performance resulting in little knowledge on how to configure these kiosks. Combining such simulations with optimization could result in a powerful tool for airports to determine optimal configurations.

The following research gap was established from the literature review: A new generation of self-service kiosks are more intuitive to use and more flexible in terms of locating them. Using a microscopic agent based simulation approach together with optimization techniques could aid airports in determining an optimal configuration.

7.1.1. Research Objective

With the problem definition and research gap established the following research objective can be formed:

To determine an optimal configuration of self-service luggage kiosks for checked luggage before security and cabin luggage at the gate using a simulation optimization approach.

7.1.2. Preliminary Optimization Technique Trade-Off

Now that the problem is defined, a preliminary trade-off of optimization techniques can be performed. Since the exact number of locations where the self-service kiosks can be located is currently unknown, the size of the parameter space is also not known. Therefore an effort will be made to score the techniques discussed in chapter 5 although the knowledge on how different algorithms compare for simulation optimization is lacking [3]. The emphasis during this trade-off will lie on the scalability and adaptability of the algorithms due to the not known size of the parameter space.

Following from chapter 5, ranking and selection can automatically be disregarded as it is only viable for finite/small parameter spaces. This is the case since comparing and ranking is too slow for large parameter

spaces Therefore, the options that remain and efficiently explore and exploit the parameter space are: metaheuristics, response surface methodology, model-based methods and direct search methods.
In order to perform an effective trade-off, scores (1,2,3) will be awarded to the different techniques for different categories. These categories include: efficiency, adaptability, scalability and simplicity. Metaheuristic approaches get a good score for adaptability and scalability since many different algorithms are available. Additionally, metaheuristic approaches can be paired with a simheuristic framework further increasing the efficiency. Response surface methodology excels in efficiency once a metamodel is created to map the input to output relation. This has as a consequence however that it is difficult to adapt which would require retraining the metamodel. Model based methods are not very simple to implement and difficult to scale due to these methods being very dependent on the parameter space being known. This has a benefit however in terms of efficiency. The final method is direct search method which is easy implement and can be easily tailored to different problems as it is a sequential examination of trial solutions. The strategy the generates these solutions however may not be easily scalable which leads to a low score for this category. An overview of the scores given can be found in Table 7.1.

Table 7.1: Trade-off table for optimization techniques

Technique	Adaptability	Scalability	Efficiency	Simplicity	Total
Metaheuristics	3	3	2	3	**11**
Model-based methodology	1	2	3	1	**7**
Response Surface Methodology	1	2	3	2	**8**
Direct search methods	3	1	2	3	**9**

From this trade-off can be concluded that a metaheuristic approach would strike a good balance between the different categories. Metaheuristics with a simheuristic framework would further increase the efficiency although this decreases the simplicity since a deterministic model must be developed. Therefore, a metaheuristic approach with a simheuristic framework seems to be the preferred optimization method.

7.2. Research Questions

Following from the problem description and the identified research gap, a research question can be formed. In this section the main research question along with the sub questions will be presented.

7.2.1. Main Research Question

How can an optimal configuration be determined for self-service kiosks in an airport terminal for checking in luggage before security and checking in cabin luggage at the gate?

7.2.2. Sub Questions

How are self-service kiosks involved in airport terminal processes?

- How are self-service kiosks before security involved in the airport terminal process?
- How are self-service kiosks at the gate involved in the airport terminal process?

How can self-service kiosks be modeled in an airport terminal simulator?

- What is the concept of operations for self-service kiosks before security?
- What is the concept of operations for self-service kiosks at the gate?
- Which passenger flows must be modeled?
- How can the interactions resulting from the addition of self-service kiosks be modeled?
- How can the modeled self-service kiosk be verified?

What data is necessary for modeling self-service kiosks?

- Which regulations are imposed on self-service kiosks?

- What parameters and data must collected in order to model the self-service kiosks?

Which metaheuristic algorithm is appropriate for the problem?

- How is the parameter space defined resulting from the case study?
- How can an appropriate optimization method be selected?
- How does a simheuristic approach influence the accuracy and computational burden of the solution?
- What accuracy is required?

How is an optimal configuration defined?

- Which performance metrics must be optimized?
- What weights should be given to the different performance metrics in the objective function?

What is the effect of adding self-service kiosks in an airport terminal on passenger flow dynamics?

- Which performance metrics can be used to understand the effect of self-service kiosks on the passenger flow dynamics?
- What can be understood from the simulation visualization in terms of passenger flow dynamics?

What recommendations can be given airports wanting to implement self-service kiosks in terminals?

- What recommendations can be given for the configuration of self-service kiosks before security?
- What recommendations can be given for the configuration of self-service kiosks at the gate?

8

Research Methodology

Following the research proposal, the research methodology will be presented. First, the case study for Rotterdam The Hague Airport (RTHA) will be discussed. After that, the research methodology will be concluded with the work packages of which this research will comprise along with the corresponding planning.

8.1. Case Study RTHA

As mentioned in chapter 1, a request was made by Rotterdam The Hague Airport (RTHA) for help with finding a good configuration for self-service kiosks. RTHA is the third airport of the Netherlands and serves roughly 2 million passenger every year [34]. In this section a preliminary case study will be described.

RTHA recently implemented a new lay-out of the terminal where the passenger check-in self-service kiosks are removed and a new split self-service baggage check-in system will be implemented. A split self-service baggage check-in system refers to a separate bag tag and drop-off kiosk. In order to place these new kiosks tactically, the TU Delft and RTHA will collaborate using simulation and optimization together. Using the experience RTHA has configuring airport terminals possible locations will be identified beforehand to reduce the problem size and determine the parameter space. The possible locations will be indicated by a green square in the current terminal lay-out as illustrated by Figure 8.1. These locations are determined yet however.

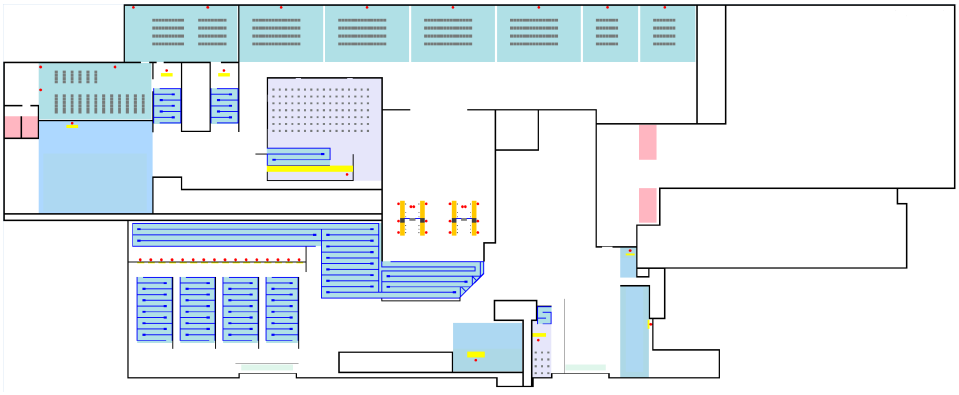


Figure 8.1: Lay-out of RTHA in AATOM where possible kiosk locations would be indicated by a green square.

As mentioned before, next to the possible locations, the performance metrics which should be taken into account by the optimization algorithm have to be determined. These metrics will also determined together with RTHA. Examples of performance metrics include: average queue length, average queue time and average walking distance per passenger.

8.2. Work Packages

In this section the work packages that will form the thesis will be given.

- **Work package 1 - Development of self-service kiosks in AATOM:** In this work package the self-service kiosks (bag tag kiosks and drop-off kiosk before security and bag tag kiosks at the gate) will be developed in AATOM. This will be done in a test environment which can be compiled and tested quickly.
- **Work package 2 - Implementing self-service kiosks in RTHA layout:** Once the kiosks are implemented as objects that can be placed in AATOM along with the interactions, the kiosks can be added to the RTHA lay-out. This is necessary since the kiosks have to be tested in order to determine whether they operate as expected in a more elaborate simulation scenario. The final activity for this work package is defining the traffic scenario. Only departing passengers will be considered during when defining the traffic scenario
- **Work package 3 - Implementation of metaheuristic optimization algorithm:** Next a trade-off will be performed to determine the appropriate metaheuristic algorithm. Once the metaheuristic algorithm is determined, it can be implemented .
- **Work package 4 - Implementation of simheuristic framework:** In order to implement a simheuristic framework, a deterministic version of AATOM must be developed. This is necessary due to the stochasticity being located in the objective function of the optimization problem. Constructing the deterministic model will consist of determining where the stochasticity is located in the model and replacing it with a deterministic value. This deterministic copy must then be saved such that it can be accessed by the simulation optimization algorithm during the optimization process.
- **Work package 5 - Integration of simulation optimization:** The last step before preliminary simulation results can be presented is the integration of the metaheuristic optimization algorithm with the AATOM simulator in the simheuristic framework. The use of the deterministic and stochastic version of AATOM must carefully controlled to ensure the correct version is used at each point in the optimization process.
- **Work package 6 - Data collecting:** This work package in which the data necessary for the simulations will be collected, can be executed in parallel to the previous packages. This is the case since the service rate can take any value for the development and testing of the simulation and optimization. To better understand which data is necessary for AATOM to simulate successfully the research by Janssen et al. [15] can be used.
- **Work package 7 - Analysis of simulation optimization results:** Once the AATOM simulator with self-service kiosks and simulation optimization integration is completed, the optimization runs can be started. The analysis of the results of the optimization runs will include validating and preparing the results for presentation to RTHA.
- **Work package 8 - Reporting:** Reporting will be done during and after every work package in order to accurately document the process and findings.
- **Work package 9 - Thesis defense preparation:** Once the thesis is handed-in and green light has been given, the defense of thesis preparation will start. This also includes preparing a thesis presentation.

8.3. Planning

In Figure 8.2 a Gantt chart of the planning of the research can be found. A certain time period is assigned to reporting in the Gantt chart in Figure 8.2. This time can be used throughout the whole process to ensure that everything is accurately documented.

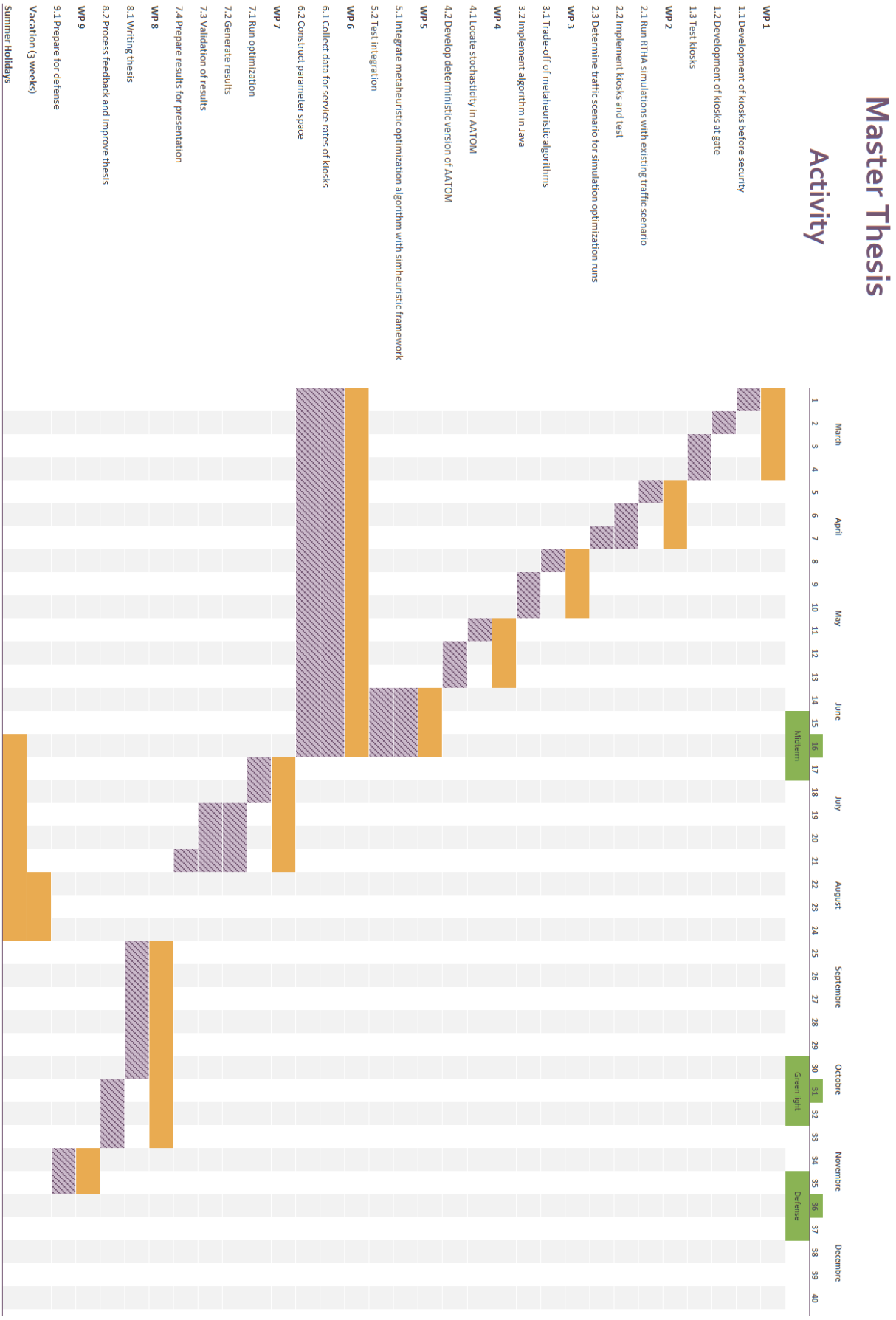


Figure 8.2: Gantt chart of the planning of the work packages.

III

Supporting work

1 Extension of AATOM

In this appendix, the steps taken to extend AATOM with the self-service luggage check-in system are discussed. These steps include the implementation of the self-service process in AATOM and the RTHA layout and the verification process.

1.1. Model Development

In this section the model development is more elaborated upon. A more technical insight will be give on how the implementation was programmed.

1.1.1. Development of PhysicalObjects

The first step of developing the self-service kiosks in AATOM is to define the kiosks as physical objects. A physical object can be observed and interacted with by the passengers. The two types of kiosks which are implemented are bag tag kiosks and drop-off kiosks.

Bag tag kiosks

The bag tag kiosk is used to provide bag tags with information such as the flight. There are two types of bag tag kiosks: bag tag kiosks before security and bag tag kiosks at the gates. These kiosks can respectively be used to bag tag luggage that can not be taken into the cabin and carry-on luggage. The main difference between these kiosks in the simulator is that the bag tag kiosk at the gate has an extra attribute: the gate where it is located. Both bag tag kiosks have the following attributes:

- servingPosition: The position where the passenger needs to be to operate the bag tag kiosk.
- agentAtKiosk: The passenger which is at the kiosk. Is null if no passenger is using the bag tag kiosk.
- isOpen: A boolean which indicates whether a kiosk is open or not.

Drop-off kiosks

The second kiosk which is part of the self-service baggage check-in process, is the drop-off kiosk. This kiosk can be used once the passenger has attached a bag tag to the to be checked in piece of baggage. The drop-off kiosk has the following attributes:

- servingPosition: The position where the passenger needs to be to operate the bag tag kiosk.
- agentAtKiosk: The passenger which is at the kiosk. Is null if no passenger is using the bag tag kiosk.
- isOpen: A boolean which indicates whether a kiosk is open or not.
- luggageInSystem: The baggage which is currently on the belt.
- toBeRemoved: The baggage that reached the end of the belt and needs to be removed.
- moveLuggageVector: The vector indicating the direction of travel for the baggage on the belt.

As can be seen above, the first three attributes are identical to the bag tag kiosk. The last three enable the visualization of luggage being transported on the belt. Once a piece of baggage reaches the end of the belt, the piece of luggage is destroyed by AATOM. In reality, the luggage will have entered the luggage handling system.

Once a piece baggage is placed on the belt, it is not in possession of the passenger anymore. This entails that the piece luggage is removed from the luggage list of the passenger. The visualization of luggage on the belt is handled by the update function of the drop-off kiosk. The pseudo code for this function can be found below:

Algorithm 1 Update function drop-off kiosk

```
for Luggage in luggage on belt do
    if Luggage is not close to end position of belt then
        Update luggage position
    else if Luggage is close to end position of belt then
        Set luggage to be removed          ▷ Luggage will be removed once it reached the end of the belt
    end if
end for
for Luggage that needs to be removed do
    Remove luggage from luggage on belt
    Destroy luggage                        ▷ Luggage now in luggage system
end for
```

1.1.2. Development of Self-Service Process Activities

Next to the physical objects, the corresponding activities must be defined. The activities are part of the goal module which ensures that the passengers perform all necessary tasks before boarding the aircraft. Two new activities were added: the bag tag and drop-off activity.

Bag tag activity

The bag tag activity is performed at the bag tag kiosk. Both the bag tag activity before security and at the gate are the same with the one difference that the activity before security bag tags checked luggage and the activity at the gate bag tags carry on baggage. The bag tag activity has the following attributes to ensure correct functionality:

- servingKiosk: The kiosk which will be used to bag tag the baggage.
- bagTagTimeDist: Statistical distribution from which the service rate will be sampled.

When the next activity for the passenger is a bag tag activity, first the nearest queue for a bag tag kiosk is found. The passenger must choose which queue to go to as there are multiple bag tag locations. The passenger then enters the queue and waits until a kiosk becomes available. At the kiosk the the pieces of baggage are bag tagged in the update function. Pseudo code for this function can be found below:

Algorithm 2 Update function bag tag activity

```
if Distance from passenger to kiosk ≤ 0.3 then
    for luggage in passenger.luggage do
        if luggage type = checked AND not bag tagged then    ▷ Only baggage that is not carry on
            Waiting time ← value drawn from bagTagTimeDist   ▷ Service rate
            Luggage ← bag tagged = True
        end if
    end for
    if Waiting time is over then
        Free up kiosk for next passenger
        End activity
    end if
end if
```

Another important process within the bag tag activity is the queue selection process. There are three possible locations identified at RTHA for kiosks. Each of these locations will have their own queue which the passenger needs to go through before the kiosk can be utilized. The passenger needs to consider the distance to the queue and number of passengers in the queue in order to make a decision. At first, the passenger will choose the closest queue with less than five passengers in it. If a queue has more five passengers in it, the passenger will not consider the queue as an option. It could be possible however that all queues have five or more passengers in them. In that case the passenger will select the queue with the least passengers in it.

An extra check has been added to ensure that the to be evaluated queue is not a queue for the drop-off kiosks. Due to the close proximity of the kiosks and queues in terminal, a passenger could choose the queue at the drop-off kiosk as the bag tag kiosk queue. This resulted in passengers entering and eventually blocking the drop-off queue rendering the simulation unusable. Therefore this check is important as it ensures that this error can not occur.

Pseudo code of the queue selection function can be found in Algorithm 3.

Drop-off activity

The drop-off activity is set up in a similar way as the bag tag activity. The passenger enters a queue and waits until a kiosk is free to drop off the baggage. The attributes this activity has are also similar to the ones for the bag tag activity and can be found below:

- servingKiosk: The kiosk which will be used to bag tag the baggage.
- dropOffTimeDist: Statistical distribution from which the service rate will be sampled.

The pseudo code for the update function for the drop off activity can be found below:

Algorithm 4 Update function drop-off activity

```
1:  if Distance from passenger to kiosk ≤ 0.3 then
2:      for luggage in passenger.luggage do
3:          if luggage type = checked AND bag tagged AND not checked in then   ▷ Only baggage that is not
    carry on
4:              Waiting time ← value drawn from dropOffTimeDist                 ▷ Service rate
5:              Luggage ← checked in = true
6:              Add luggage to luggage on the belt collection
7:          end if
8:      end for
9:      if Waiting time is over then
10:         Free up kiosk for next passenger
11:         End activity
12:     end if
13: end if
```

1.1.3. Integration with Existing Model

The next step is to integrate the self-service luggage check-in process into the existing model. To achieve this, the activities must be added into the goal and planning module. Additionally, the gate operator activity must also be updated. The functionality for deciding whether or not the passenger must check in their cabin luggage must be added. The final step taken to integrate the self-service luggage check-in system is the updated agent generator.

Goal and planning module

The goal and planning module are updated such that the newly developed activities can also completed by passengers. When a passenger can use the self-service system, the bag tag and drop-off activity must be added to the goal and planning module in that particular order. The order of the goal and planning module is important as is this is also the order passengers will perform activities.

Algorithm 3 Queue selection function

```
List of bag tag kiosks ← bag tag kiosks
List of queues ← all queues in terminal
List of drop-off kiosks ← drop-off kiosks

Closest queuing area ← null
Number of passengers in closest queue ← 1e6
Distance to closest queue ← 1e6
                                  ▷ Check for queues which have less to five passengers in them
for Bag tag kiosk in list of bag tag kiosks do
    for Queue in list of all queues in terminal do
        Get distance kiosk to queue
        Get distance passenger to queue
        Get number of passengers in Queue
        if Distance kiosk to queue < 3 AND Queue is not drop-off queue then
            if Distance passenger to queue < distance to closest queue AND Number passengers in Queue < 5 then
                Closest queuing area ← Queue
                Number of passengers in closest queue ← Number of passengers in Queue
                Distance to closest queue ← Distance passenger to Queue
            end if
        end if
    end for
end for
                                  ▷ No queue with less than five passengers is found
if Closest queuing area == null then
    for Bag tag kiosk in list of bag tag kiosks do
        for Queue in list of all queues in terminal do
            Get distance kiosk to queue
            Get distance passenger to queue
            Get number of passengers in Queue
            if Distance kiosk to queue < 3 AND Queue is not drop-off queue then
                if Distance passenger to queue < distance to closest queue AND Number passengers in Queue < Number of passengers in closest queue then
                    Closest queuing area ← Queue
                    Number of passengers in closest queue ← Number of passengers in Queue
                    Distance to closest queue ← Distance passenger to Queue
                end if
            end if
        end for
    end for
end if
```

Passengers can only utilize the self-service option when they are checked in online and have luggage with them which requires check-in. The goal and planning modules are created per passenger when the passenger is generated. During the generation of the agent a check is performed to determine which passenger can utilize the self-service system. During this check the attributes of the passenger are checked and the correct goals are assigned to the passenger.

The bag tag activity at the gate required another approach. During the generation of a passenger it is not yet known if a passenger must check-in their carry-on luggage. Whether or not a passenger must check-in its carry-on luggage is dependent on the number of passengers on the flight bringing carry-on luggage and the capacity of the aircraft. Ideally, the bag tag activity at the gate is added to the goal and planning module of a passenger during the simulation. To do so is very computationally expensive however. An alternative method was found by giving every passenger the bag tag activity at the gate. The gate operator, which will be discussed in the next section, then decides whether or not the goal must fulfilled. The bag tag activity is located in the goal module after the gate activity and before the boarding activity.

Gate operator activity

The gate operator is responsible for performing a final check of the documents before the passenger can board the aircraft. In some cases however, the passenger must check-in their carry-on luggage due to the limited amount space available on the aircraft. The gate operator will count the number of pieces of carry-on luggage that are brought on board and will start checking these in when the maximum capacity is reached. The attributes of this activity are:

- agentToCheck: The passenger the operator will check.
- agentHasLargeCarryOn: Tracker whether a passenger has large carry on luggage.
- alreadyInstructed: List of agents which are already instructed by the operator.

The pseudo code for this process can be found in Algorithm 5.

Algorithm 5 Update function gate operator activity

```
1: if Flight leaves < 10 minutes then
2:     Stop the boarding process
3: end if
4:
5: if agentToCheck == null then
6:     agentToCheck ← random agent
7:     agentHasLargeCarryOn = false                    ▷ Tracker to determine size of carry on luggage
8: end if
9:
10: if agentToCheck != null then
11:     if agentToCheck not in alreadyInstructed then                    ▷ Agent not instructed
12:         if agentToCheck not sitting AND agentToCheck is close to operator then
13:             if agentToCheck.getLuggage.getSizeCabinLuggage == true then
14:                 this.numLargeCarryOnBoarded += 1                ▷ Update large carry on luggage counter
15:                 agentHasLargeCarryOn ← true
16:             end if
17:         end if
18:
19:         if waiting time of agenToCheck is over AND agentToCheck already instructed then   ▷ Manual bag
    tagging complete
20:             agentToCheck ← boarded = true
21:             agentToCheck ← null
22:         end if
23:
24:         if numLargeCarryOnBoarder > capacity AND agentHasLargeCarryOn AND agentToCheck not al-
    ready instructed then
25:             if kiosk is initialized then
26:                 if kiosk is open then                                ▷ Bag tagging with kiosk
27:                     Send agent to kiosk for bag tagging
28:                     Add agentToCheck to alreadyInstructed
29:                     agentToCheck = null
30:
31:                 else                                                 ▷ Manual bag tagging
32:                     Bag tag manually by communicating to wait
33:                     Add agentToCheck to alreadyInstructed
34:
35:                 end if
36:
37:             else
38:                 Bag tag manually by communicating to wait
39:                 Add agentToCheck to alreadyInstructed
40:             end if
41:
42:         else
43:             agentToCheck ← boarded = true
44:             Add agentToCheck to alreadyInstructed
45:             agentToCheck ← null
46:         end if
47:     end if
48:
49: else                                                    ▷ No luggage to be checked in
50:     boarding = true                                     ▷ Agent has boarded the aircraft
51:     agentToCheck = null
52: end if
```

Agent Generator

The previous version of the agent generator was only able to generate passengers with checked luggage if the passengers were not checked in online. With the self-service luggage check-in system implemented however, passengers can check in luggage when the passenger is checked in online.

With the agent generator updated, the following combinations of passenger check-in and luggage status are possible:

- Passenger checked in online with luggage that needs to be check in.
- Passenger checked in online without luggage that needs to be check in.
- Passenger not checked in online with luggage that needs to be check in.
- Passenger not checked in online without luggage that needs to be check in.

1.2. Implementation of Self-Service Luggage Check-In System in the RTHA Layout

To perform experiments in the RTHA layout, this layout must first be prepared for these the kiosks. In this section the reconfiguration of the RTHA layout and the new agent generator.

1.2.1. Reconfiguring RTHA Layout to Accommodate Kiosks

Reconfiguring the layout of RTHA in the simulator consisted of changing the queuing areas and finding a suitable location for the kiosks. In Figure 1.1 the initial layout can be found.

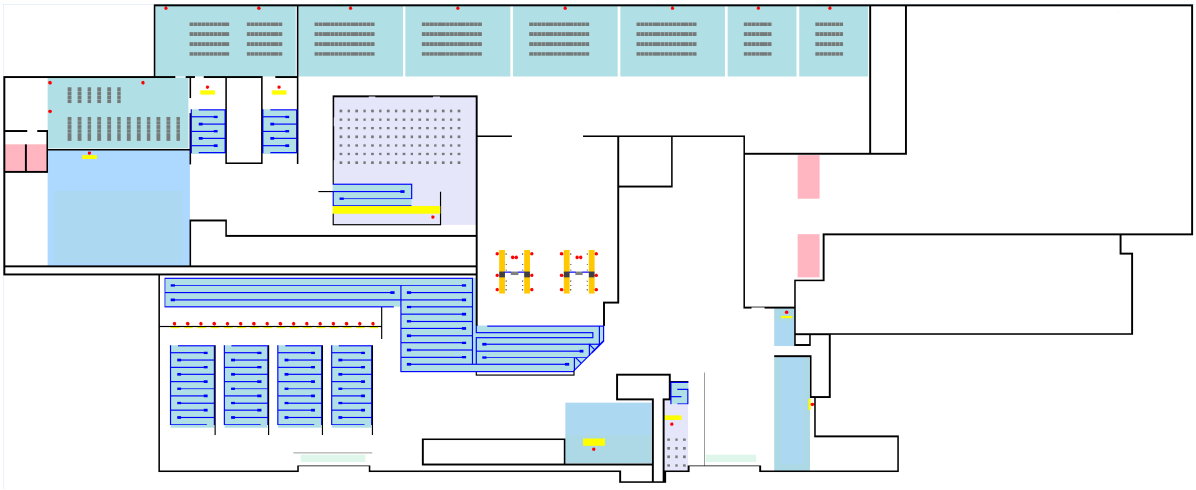


Figure 1.1: RTHA layout without kiosks

After reconfiguration the queue areas and placing the kiosks in the RTHA layout the simulator is as depicted in Figure 1.2. Important to note is that the simulator is also extended with an outside area to be able to place kiosks there as well.

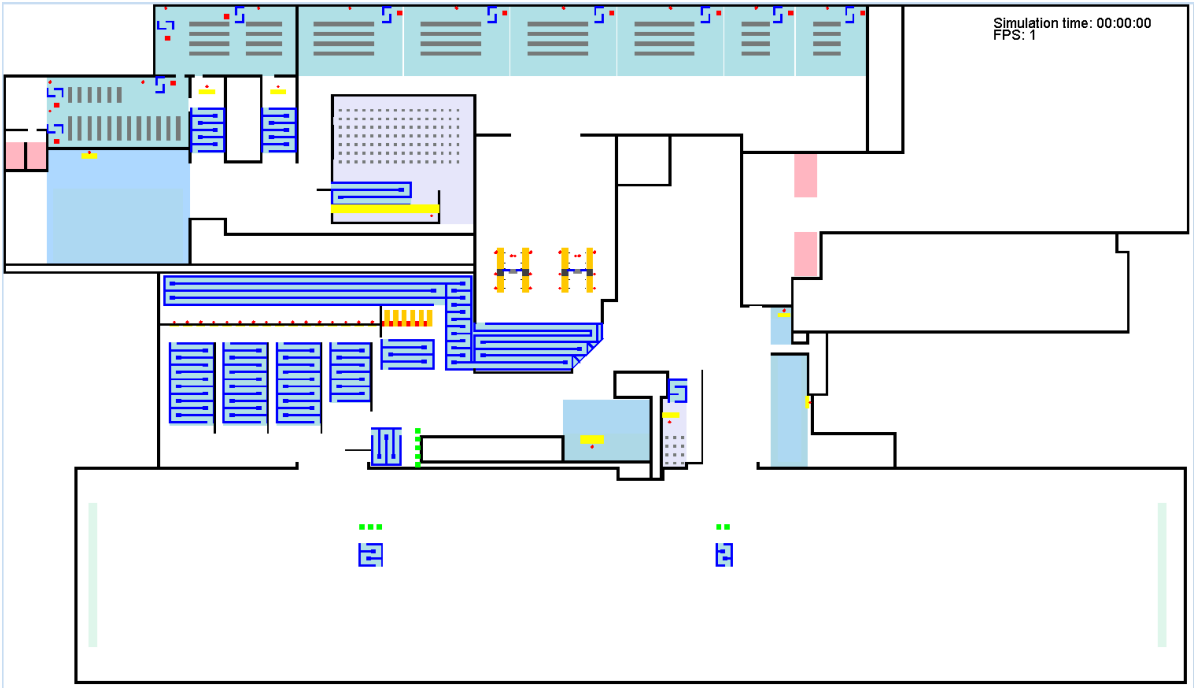


Figure 1.2: RTHA layout with kiosks

The gate kiosks also have a small queue in case that multiple passenger must utilize the kiosk right after each other.

1.3. Verification

To verify AATOM with the newly added facilities, multiple tests have been carried out. The verification was performed on two levels: facility level and terminal level. The verification process utilized two testing methods: testing via the GUI and manual testing. The GUI was used to verify that the correct passenger flows emerged for example. Manual testing allowed verifying that the variables were updated to the correct values. In this section first the verification of each facility is discussed. After that, the verification of the complete terminal will be discussed. This section will be concluded with a section on the changes which resulted from the verification.

Verification of bag tag process

The type of test will be denoted after each aspect that is tested. For the verification process of the bag tag kiosks and the corresponding activity the following aspects where tested:

- The agent correctly identifies and finds the closest free kiosk (GUI)
- The agent initiates the bag tagging process and after the waiting time is over, the bag tag status of the luggage is correctly updated (Manual testing)
- The agent leaves the kiosk which becomes free again (Manual testing)
- The agent queues correctly (GUI)
- The agent does not get stuck (GUI)

Verification of drop-off process

The verification of the drop-off kiosks and the corresponding activity is done by testing the aspects listed below:

- The agent correctly identifies and finds the closest free kiosk (GUI)

- The agent initiates the luggage drop-off and after the waiting time is over, the checked in status of the luggage is correctly updated (Manual testing)
- The agent leaves the kiosk which becomes free again (Manual testing)
- The agent queues correctly (GUI)
- The agent does not get stuck (GUI)

Verification of gate kiosks

In order to verify the bag tag kiosks at the gate and the corresponding activity the following aspects where tested:

- The agent correctly identifies and finds the closest free kiosk (GUI)
- The agent initiates the bag tagging process and after the waiting time is over, the bag tag status of the luggage is correctly updated (manual test)
- The agent leaves the kiosk which becomes free again (manual test)
- The agent correctly queues (GUI)
- The agent does not get stuck (GUI)
- Manual bag tagging works correctly (GUI)

Additionally, the added decision making process of the gate operator was tested. The aspects which were tested can be found below:

- The gate operator correctly tracks the number of large pieces of cabin luggage (manual test)
- The gate operator sends passengers to kiosk (if present) when capacity of aircraft is reached (GUI and manual test)
- The gate operator correctly bag tags and checks in luggage when no kiosk is present (manual test)

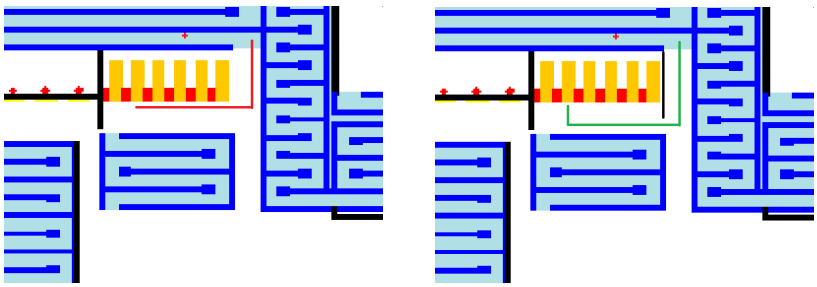
Verification of terminal

Now that the facilities are thoroughly tested separately, the whole terminal can be tested. Below the aspects can be found which were tested:

- No agents get stuck (GUI)
- The desired passenger flow emerge. The flows are: entrance - check-in - security, entrance - bag tag kiosk - drop-off kiosk - security, entrance - security (GUI)
- The agents choose the correct bag tag queue (GUI and manual test)

Changes due to bugs found during verification

Following from the verification process, some bugs were found which required fixing. The first main bug which arose was the bug which caused agents to queue incorrectly. Agents were passing to queue instead of going through as this was the shortest path. By setting a temporary or intermediate goal, the agents are forced to go through the queue. Another bug that was found during verification was agents running into each other when trying to leave the kiosk. By adding a wall this could be fixed since it physical objects affect the path planning of agents. In Figure 1.3 the old and new situation can be found.



(a) Without extra wall (b) With extra wall

Figure 1.3: Drop off kiosk without and with extra wall

The final bug that was found occurred at the gate. The gate operator calls each agent one by one in order to let the agent board. To determine which agent to call, the gate operator activity has a function which chooses a random agent in the gate area. This function however did not work correctly since it tried to call an agent which was going to the kiosk and thus already had visited the gate operator. By building in an extra check which prevents trying to call an agent twice, this bug was fixed.

2

Simulation Optimization with Simheuristics

In this appendix, the Simheuristic simulation optimization procedure is elaborated upon. First the algorithm trade-off is presented where the metaheuristic algorithm is chosen. Next, Simheuristics is discussed more in depth. This includes the integration of the Simheuristic framework with AATOM and the construction of the deterministic version. Last, the optimization software is discussed.

2.1. Algorithm Trade-Off

To determine the algorithm that will be used for optimizing the configuration of the kiosks, a trade-off is performed. This trade-off is similar to the trade-off performed in chapter 7 and will be applied to the algorithms that fall under metaheuristics. The criteria that will be used during the trade-off are the same as used in chapter 7: adaptivity, scalability, efficiency and simplicity. Efficiency and simplicity refer to how efficient the algorithm can navigate the parameter space and how easy it is to implement the algorithm. The adaptivity refers to whether the algorithm can be used to be more exploring or exploiting the solution space and scalability if the algorithm can easily be applied to problems with different solutions space sizes.

In Table 2.1, the results of the trade-off can be found. As was also identified in the trade-off done by van der Horst [43], the single solution based methods are the most simple and scalable. The genetic algorithm on the other hand excels in efficiency but is much more challenging to implement successfully. Finally, tabu search is more efficient and more adaptable than the other single solution based methods due to the tabu list. This tabu list prevents previously visited solutions to be visited again making it more efficient whilst also providing control of the exploration and exploitation characteristics.

Table 2.1: Trade-off table for optimization techniques

Technique	**Adaptability**	**Scalability**	**Efficiency**	**Simplicity**	**Total**
Simulated Annealing	2	3	1	3	**9**
Tabu Search	3	3	2	3	**10**
Iterated Local Search	2	3	1	3	**9**
Variable Neighborhood Search	1	3	1	3	**8**
GRASP	1	3	1	3	**8**
Genetic Algorithms	2	2	3	1	**8**

From the trade-off above it can be concluded that tabu search is the best option for this research. Due to the computational resources required for one simulation a Simheuristic framework will be used. The implementation of this framework will be discussed in the next section.

2.2. Simheuristics

The agent-based model used for simulating the passengers in the airport terminal has a stochastic nature. Therefore multiple simulation runs need to be performed to determine the average fitness of a certain solution. This requires substantial computational resources. Therefore a reduction in number of simulation runs while still finding a good enough solution is desired. A simheuristic framework could aid in reducing the necessary number of simulation runs while optimizing the solution. The Simheuristic framework has been discussed before in chapter 5. In this section the implementation of this framework for the kiosk location optimization problem will be discussed. First the integration of the tabu search algorithm with the framework will be discussed. The process of constructing the deterministic version of AATOM is discussed will also be discussed in this section.

2.2.1. Integration of tabu search with simheuristic framework

The tabu search algorithm will be used to identify promising solutions when evaluated with the deterministic version of AATOM. This deterministic version of AATOM will be explained in the next section.

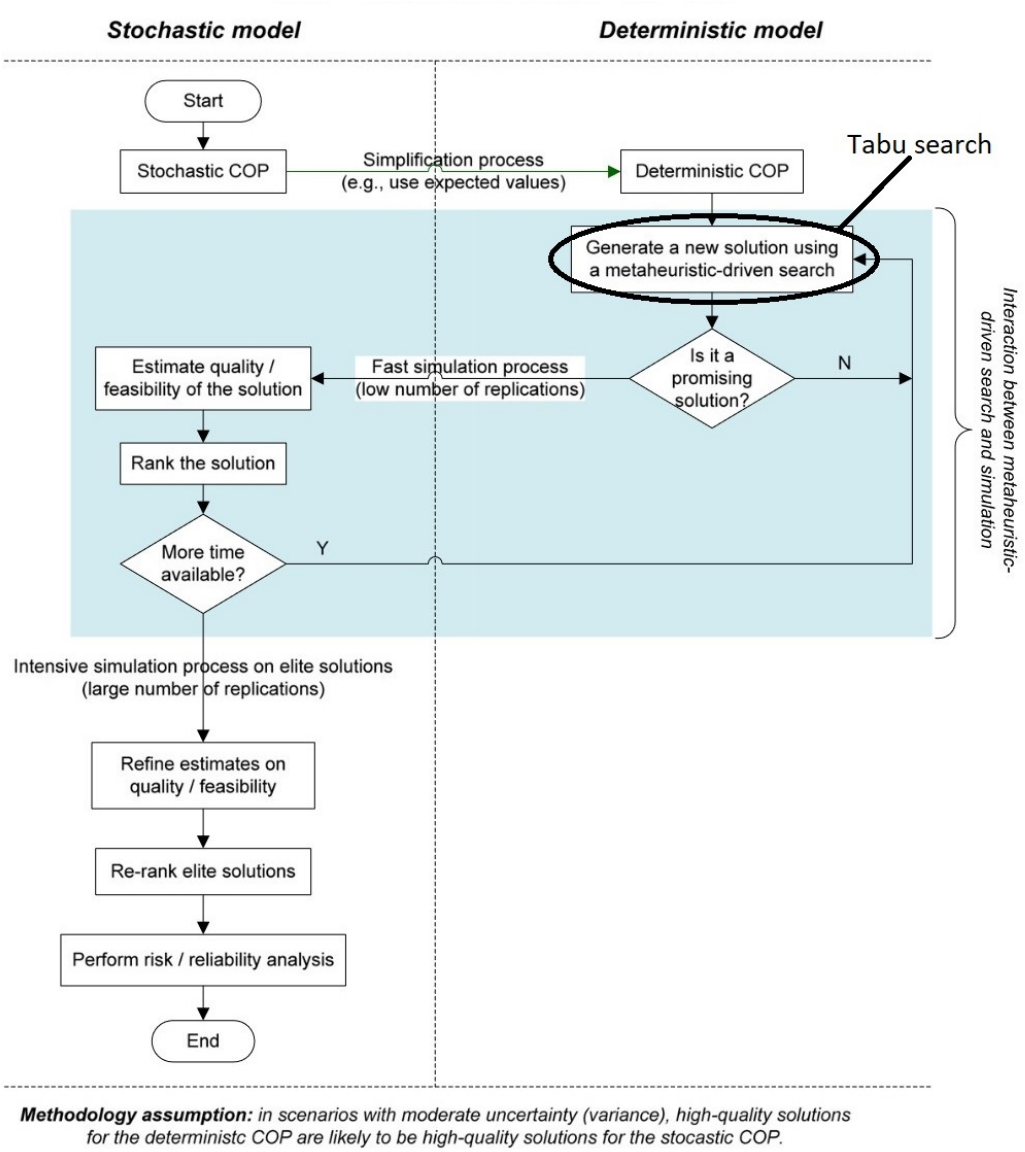


Figure 2.1: Simheuristics with tabu search

As can be seen in Figure 2.1, the tabu search algorithm is responsible for identifying promising solutions using the deterministic version. A solution is promising when it deemed to be a local optimum.

2.2.2. Constructing the deterministic version of AATOM

The Simheuristic framework requires the development of a deterministic version of AATOM. As suggested by Juan et al. [16], when no analytical model is available, a deterministic version of the model can be constructed by replacing the random variables with their expected values. AATOM contains many random variables to model stochastic processes. Additionally, the generation of agents is of stochastic nature and needs to be made deterministic. In this section, the steps taken to construct the deterministic version of AATOM will be discussed as well as the verification.

Replacing the random variables

Before the random variables can be replaced, first the random variables must be identified. This is done by going through all classes of the project and identifying which values are sampled from a distribution. In Table 2.2, an overview of the identified random variables can be found.

Table 2.2: Random variables present in AATOM and their expected value

Class	Variable	Distribution	Expected Value
Passenger	Checkpointdroptime	Normal(μ,σ)	μ
Passenger	Checkpointcollecttime	Normal(μ,σ)	μ
BasicPassengerDecisionMaking	t-ps_dist	Normal(μ,σ)	μ
BasicPassengerDecisionMaking	t-r_dist	Gamma(a,b)	$\frac{a}{b}$
BasicPassengerDecisionMaking	t-s_dist	Gamma(a,b)	$\frac{a}{b}$
BasicPassengerDecisionMaking	t-us_dist	Normal(μ,σ)	μ
BasicOperatorShopActivity	waitingtime	Normal(μ,σ)	μ
BasicOperatorGateTDCActivity	waitingtime	Normal(μ,σ)	μ
BasicOperatorGateTDCActivity	manualBagTagtime	Normal(μ,σ)	μ
BasicOperatorCheckInActivity	waitingtime	Normal(μ,σ)	μ
BasicOperatorBorderControlActivity	waitingtime	Normal(μ,σ)	μ
BasicLuggageCheckActivity	timeDistribution	Normal(μ,σ)	μ
BasicETDCheckActivity	waitingDistribution	Normal(μ,σ)	μ
BasicETDCheckActivity	etdDistribution	Normal(μ,σ)	μ
BasicToiletActivity	maleTime	LogNormal(μ,σ)	$e^{\mu+\frac{1}{2}\sigma}$
BasicToiletActivity	femaleTime	LogNormal(μ,σ)	$e^{\mu+\frac{1}{2}\sigma}$
BasicShopBrowseActivity	shopTimeDist	Normal(μ,σ)	μ
BasicRegionalCheckpointActivity	dropDist	Normal(μ,σ)	μ
BasicRegionalCheckpointActivity	collectDist	Normal(μ,σ)	μ
BagTagPrinter	bagTagTimeDist	Normal(μ,σ)	μ
BagTagPrinterGate	bagTagGateTimeDist	Normal(μ,σ)	μ
DropOffKiosk	dropOffTimeDist	Normal(μ,σ)	μ

Constructing a deterministic agent generator

The agent generator does not allow for simply replacing all random variables with their expected value. The agent generation is responsible for the calculation of the arrival times and the determination of the passenger characteristics. The arrival times are calculated according to a Poisson distribution and the passenger types are generated uniformly according to a distribution defined per flight. Additionally, luggage types and location where passengers are generated are also determined randomly according to a uniform distribution. To steps taken to make this process deterministic are discussed below.

In the stochastic version of AATOM the passenger types and arrival times are calculated before the simulation is started. The passenger types refer to whether the passenger is traveling for leisure, business, etc. Based on the passenger type, the passenger arrives in one of three time windows between the earliest and latest time a passenger can arrive. These time windows are all the same size. Currently, for each passenger type, 20% arrives in the first window, 60% arrives in the second window and the remaining 20% arrives in the third window. Knowing this distribution and the size of the flight for which passengers need to be generated, the passengers per arrival window can easily be calculated. Since the inter arrival times are exponentially distributed as the generation of the passengers is a Poisson process, the inter arrival times are expected to

have a value of the time window in seconds divided by the number of passengers to be generated in this time window. Using this method to calculate the inter arrival times, the times at which passengers are generated is made deterministic.

In order to determine the luggage types and checked in status per passenger, the stochastic version of AATOM must be investigated. In order to determine whether or not a passenger is checked in online, the generator first checks if passenger's flight does not leave within an hour. If the passenger's flight does leave within an hour, the passenger must be checked in online otherwise it will not make the flight. If the passenger arrives earlier, a random variable is generated using a uniform distribution between 0 and 1. If this value is lower than the threshold of 0.6, the passenger is not checked in online. In other words, of the passengers that arrive at the airport with more than an hour until the flight leaves, 60% is not checked in online.

In a similar fashion, the luggage of the passenger is generated using a uniformly distributed random variable. A threshold value is again used to control the ratio of passenger bringing checked luggage for example.

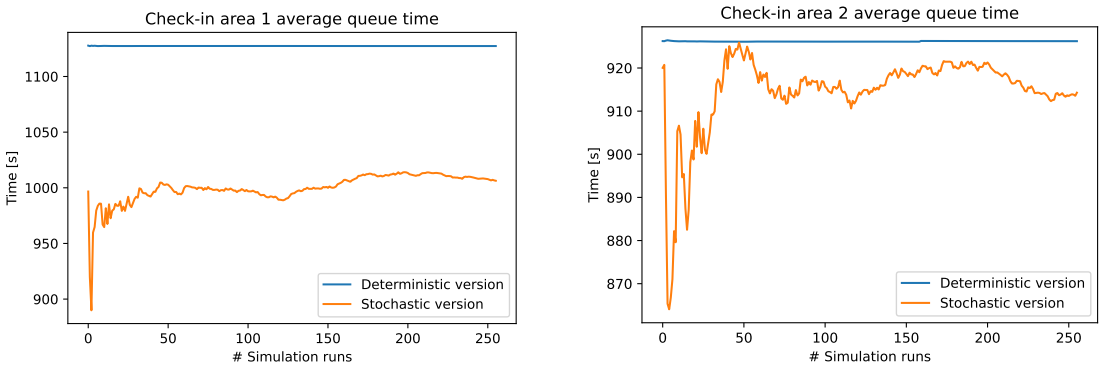
By determining the luggage types and checked in status per bucket and then generating them with the appropriate inter arrival times, this process can be made deterministic. The passenger type will also be taken into account and will be generated deterministically while adhering to the passenger distributions. Arrays are constructed containing the following information per passenger: passenger type, luggage type and checked in status using the predetermined distributions. These arrays are then shuffled deterministically (using a certain fixed seed) and can be used for passenger generation. This is done per passenger bucket to ensure that every passenger status combination is represented appropriately.

2.2.3. Verification of the deterministic version of AATOM

In addition to the verification steps explained in the paper, the moving average of the separate facilities and the CoV of the facilities was when the whole terminal was simulated were checked. Additionally, the verification steps of the deterministic agent generator is discussed here.

Verification of deterministic facilities

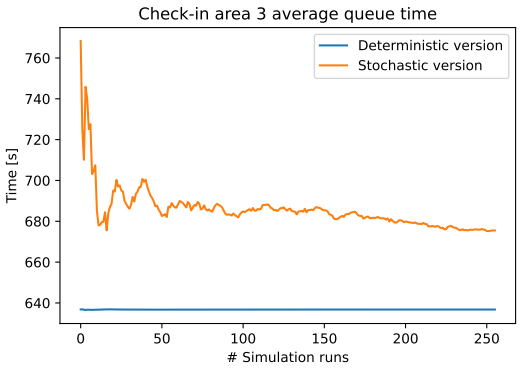
The moving average of resulting from the deterministic version is expected to be a straight line as (almost) the same output should be given each time. The output of the stochastic version is expected fluctuate more. In Figure 2.2, Figure 2.3 and Figure 2.4 this can be seen.



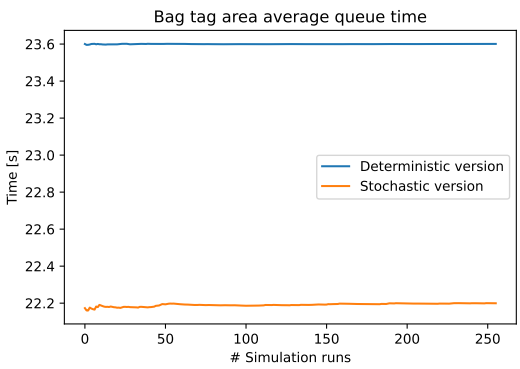
(a) Moving average of CI1 queue times

(b) Moving average of CI2 queue times

Figure 2.2: Moving average graphs

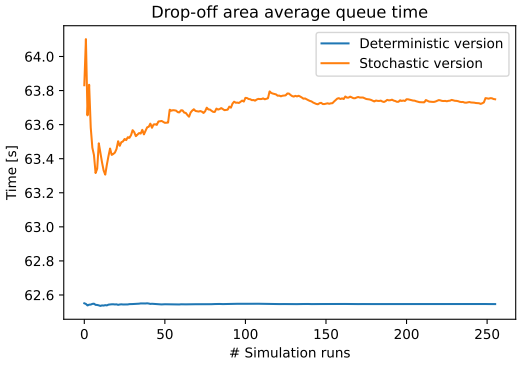


(a) Moving average of CI3 queue times

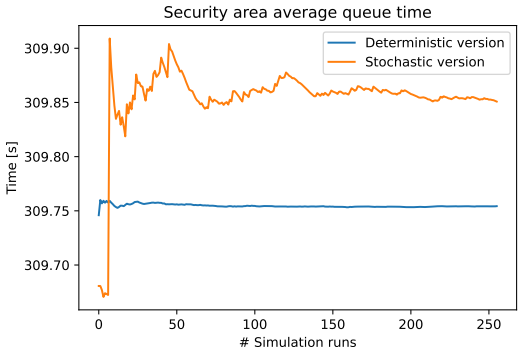


(b) Moving average of BT area queue times

Figure 2.3: Moving average graphs



(a) Moving average of DO area area queue times



(b) Moving average of Security area queue times

Figure 2.4: Moving average graphs

The CoV of the separate facilities when the whole terminal is simulated resulting fro the deterministic version can be found in Table 2.3. The CoV of the facilities resulting from the whole terminal simulation is similar to the CoV of the facilities resulting from the separate facility simulations.

Table 2.3: CoV of the average queue times for the whole terminal for the deterministic version after 64 runs

CI1	CI2	CI3	BT1	BT2	BT3	DO	Security
0.0032	0.0012	0.0009	0.00056	0.0033	0.011	0.0021	0.011

2.3. Optimization Software

Due to the missing of real-world data and to investigate the effect of the input variables on the optimal configuration, a sensitivity analysis is performed. To do this efficiently and to utilize the available resources on the server, it is desired that the optimization procedures are run in parallel. This reduces the number of simulation runs on the server from 600 to 50. In this section the program developed for this tabu search in parallel will be discussed.

The optimization program is developed using Python. A method must therefore be found and developed to communicate to AATOM using Python. To achieve this, AATOM is stored as a Jar file which can be run from Python using the subprocess package. Inputs can be passed onto AATOM using a text file which is constructed before a simulation run is started. Multiple simulations can be run parallel using the built in experimenter in AATOM. This experimenter checks the number of sets of inputs which are given in the input text file and assigns one simulation with a set of inputs to each computational core. If two sets of inputs are given for

example, two simulations will be run using two cores in total. Once a simulation is finished, AATOM outputs the results in a text file which can be read using Python.

2.3.1. Tabu search class

The tabu search class is developed to perform a single tabu search optimization procedure for a certain setting. This class has attributes to track the best candidates, elite solutions and evaluated solutions. An overview of the most important attributes can be found below:

- Best candidate: Tracker of the best candidate
- Setting: Demand level and service rates of the kiosks
- Tabu list: The tabu list
- Fitness log: Fitness for each solution evaluated with the deterministic version
- Elite solutions: Found elite solutions
- Evaluated solutions: All evaluated solutions
- Neighborhoods: List of all evaluated neighborhoods
- Best candidates log: List of the best candidates for each iteration
- Elite fitness log: Fitness for each solution evaluated with the stochastic version

By defining the attributes above per tabu search object, a structured method can be used to used link the correct results to a setting. Additionally, by looping through the list of tabu search objects, all steps of a tabu search iteration can be performed per setting. This will be elaborated upon in subsection 2.3.2.

The tabu search class has functions which can be used in the main tabu search program. These functions are used to perform the steps of a tabu search iteration. The functions are discussed in more detail below.

get neighborhood solutions

This function is used to get the neighborhood of a certain solution. The neighborhood is determined and added to the list of solutions which need to be evaluated.

add parameters best candidate

In the case that a local minimum is found and the solution needs to be perturbed or the initial solution needs to be evaluated, this function is used. This function writes the inputs of the current best candidate solution to a text file which can used as input by AATOM.

add parameters neighborhood

This function writes the inputs for all solutions in the to be evaluated neighborhood to the input text file. For each neighborhood a maximum of eight sets of inputs is written.

get fitness

This function is used to update the fitness log after an iteration. It checks if a new solutions have been evaluated for which no fitness has been calculated yet. If such a solution is found, this function will calculate the fitness and append it to the fitness log. The fitness log consists of dictionaries which are used as a link between solution and fitness.

get new best candidate

At the end of an iteration, this function is used to determine if a solution in the neighborhood is better than the current best candidate. If so, the best candidate is updated. If this is not the case, a local minimum is found and a new random best candidate is chosen.

add parameters elite solutions

Once the tabu search algorithm has identified the elite solutions, four of these elite solutions are then evaluated using the stochastic version of AATOM. This function is used to create the text file which contains the inputs of the to be evaluated elite solution.

get fitness elite solutions
This is the final function which is part of the tabu search class. This function is used to update the elite fitness log after an evaluation run. It checks which solution has been evaluated and updates the elite fitness log. The log consists of dictionaries which form the link between the input solution and corresponding list of fitness values. A fitness value list is used as an elite solution is evaluated 128 times.

2.3.2. Main Simheuristic tabu search procedure

As the tabu search class is now explained, the main code for the Simeheuristic tabu search procedure can be discussed. The purpose of this code is to initialize a tabu search object for each setting, run the tabu search optimization procedure with the deterministic version of AATOM and evaluate the resulting elite solutions with the stochastic version of AATOM. Therefore the Simheuristic tabu search procedure can be divided into three phases: the initialization phase, the iterative optimization phase and elite solution evaluation phase.

During the initialization phase, for each of the to be optimized settings, an optimization procedure in the form of a tabu search object is created. Then, an initial solution is randomly generated and set as the best solution for each optimization procedure. To conclude the the initialization phase, the initial solutions of all optimization procedures are evaluated simultaneously.

Next, the iterative optimization phase starts. During each iteration of this phase, the following steps are taken. First, for each optimization procedure, the neighborhood of the best solution is determined and the inputs for the solutions in neighborhood are added to the input file. Once the inputs of the neighborhood solutions for all optimization procedures are added to the input text file, the deterministic version of AATOM can evaluate all neighborhoods in parallel. After the simulations are finished, the fitness is determined for all evaluated neighborhood solutions. Per optimization procedure, the best neighborhood solution is compared to the best solution to determine whether a better solution is found. If for one of the optimization procedures a local minimum is found, the best solution is perturbed and evaluated for that optimization procedure.

The final phase starts once the iterative optimization phase is finished. From the iterative optimization phase a set of elite solutions follow. The four best elite solutions will be evaluated with the stochastic version of AATOM in order to accurately determine which solution is the best. Once the four elite solutions are evaluated per optimization procedure, the Simheuristic tabu search procedure is finished resulting in an optimal solution per optimization procedure.

The pseudocode of the parallel Simheuristic simulation optimization can be found below in Algorithm 6.

Algorithm 6 Parallel Simheuristic tabu search

```
Settings ← the 12 predefined settings
for setting in settings do
    ts ← tabu search object initialized with setting
    optimization procedures ← ts
end for
                                                    ▷ Evaluation of initial solutions
for setting in settings do
    Optimization procedure ← tabu search object initialized with setting
end for

for Optimization procedure in optimization procedures do
    Add parameters of best candidate to the input text file
end for

Run the deterministic AATOM simulator using the input text file

for Optimization procedure in optimization procedures do
    Get the fitness of each evaluated initial solution
end for
                                                    ▷ Start of tabu search
while not max number of iterations reached do
    for Optimization procedure in optimization procedures do
        Get the neighborhood of current best solution
        Add parameters of the neighborhood to the input text file
    end for

    Run the deterministic AATOM simulator using the input text file

    for Optimization procedure in optimization procedures do
        Get the fitness of each evaluated solution
        Get the next best candidate
    end for
                                    ▷ In case of a perturbation of the best candidate
    if No new best candidate is found for one of the optimization procedures then
        for Each optimization procedure where no new best candidate is found do
            Add parameters of perturbed solution
        end for

        Run the deterministic AATOM simulator using the input text file

        for Optimization procedure in optimization procedures do
            Get the fitness of each evaluated initial solution
        end for
    end if
end while
                                                    ▷ Evaluation of elite solutions
for Optimization procedure in optimization procedures do
    Add parameters of two best elite solutions 128 times to input text file (total 256)

    Run the stochastic AATOM simulator using the input text file

    Add parameters of third and fourth best elite solutions 128 times to input text file (total 256)

    Run the stochastic AATOM simulator using the input text file

    Get the fitness of each evaluated elite solution
end for
```

3

Additional Results

In this appendix additional results and the results from the statistical tests for both the configuration and gate experiment can be found.

3.1. Results

Complimentary results to the results in the paper can be found in this section. The optimal configuration is first discussed in more detail. After that, additional results on the impact of the optimal configuration on airport terminal performance will be shown.

3.1.1. Configuration optimization experiment

From the results in Table 3.1, it becomes clear that the outdoor kiosks are favored as discussed in the paper. A check has been performed to verify that all kiosk locations are used at some point in the simulation. This means that there is always a moment during the simulation that the queues of location two and three reach the threshold resulting in passengers deciding to go to location one. The flight schedule however is most likely not busy enough to result in a sustained flow of passengers to location one. Therefore the number of bag tag kiosks at location one is a result of the constraint for the number of kiosks at location one.

Table 3.1: Optimal number of self-service kiosks per setting

Setting	Optimal number of bag tag kiosks				Optimal number of drop-off kiosks
	Loc. 1	Loc. 2	Loc. 3	Total	
1	2	3	4	9	3
2	2	0	5	7	3
3	2	4	3	9	3
4	2	0	4	6	3
5	2	2	3	7	3
6	2	0	5	7	4
7	2	3	5	10	3
8	2	0	4	6	3
9	2	2	5	9	3
10	2	2	2	6	3
11	2	2	2	6	3
12	2	0	5	7	4

In Table 3.2, the waiting times for each passenger type and kiosk location can be found for both the optimized and base configuration. As can be seen, the waiting times for passenger types one and two increase for the base configuration as the demand level increases. The demand level is defined as the percentage of passengers bringing checked luggage. This increase in waiting time is as expected. For the optimized configuration, this increase in waiting is less apparent which is logical as passengers are spread out across multiple facilities.

Table 3.2: Average waiting times per passenger type per configuration for all settings

Setting	Base configuration average waiting times per passenger type [s]			Optimized configuration average waiting times per passenger type [s]		
	Type 1	Type 2	Type 3	Type 1	Type 2	Type 3
DL:0.4, BT sr:20, DO sr:30	951	695	951	788	691	863
DL:0.6, BT sr:20, DO sr:30	977	696	977	771	684	855
DL:0.8, BT sr:20, DO sr:30	1019	698	1019	791	683	851
DL:0.4, BT sr:20, DO sr:50	950	689	950	771	680	850
DL:0.6, BT sr:20, DO sr:50	964	682	964	816	658	830
DL:0.8, BT sr:20, DO sr:50	1027	698	1027	793	653	828
DL:0.4, BT sr:40, DO sr:30	940	682	940	768	672	840
DL:0.6, BT sr:40, DO sr:30	1006	688	1006	756	667	837
DL:0.8, BT sr:40, DO sr:30	1027	698	1027	784	674	847
DL:0.4, BT sr:40, DO sr:50	942	687	942	784	681	849
DL:0.6, BT sr:40, DO sr:50	971	687	971	811	656	823
DL:0.8, BT sr:40, DO sr:50	1022	701	1022	800	665	833

In the boxplots below, the waiting times resulting from 256 runs for each passenger type, configuration and setting can be found.

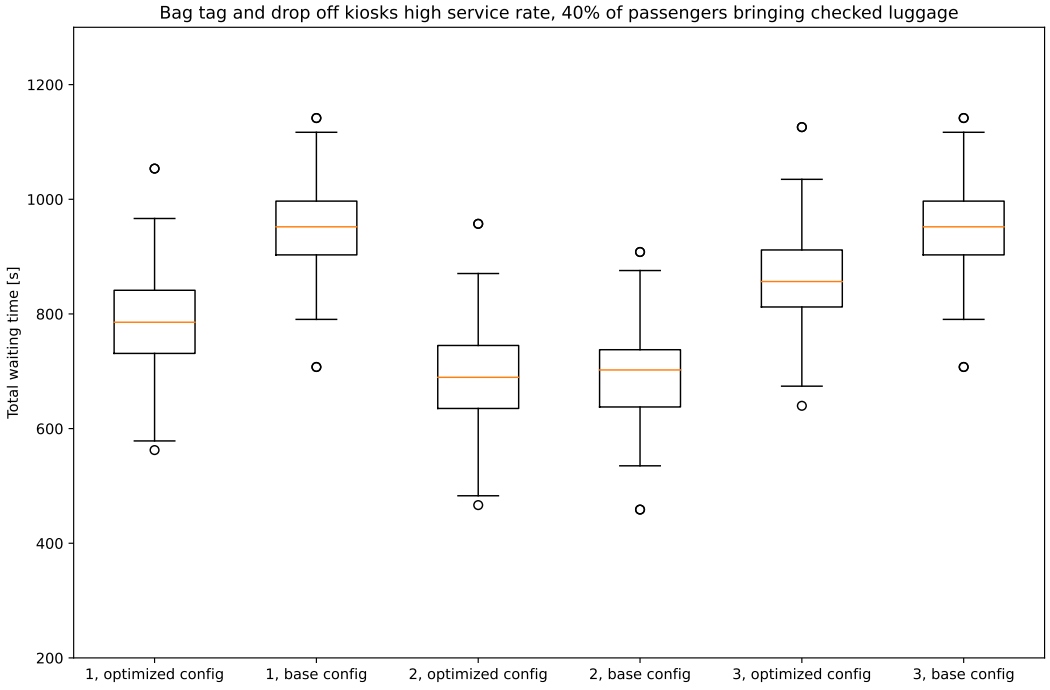


Figure 3.1: Waiting times per passenger type for optimized and base config and setting: (dl:0.4, bt sr: 20 s, do sr 30)

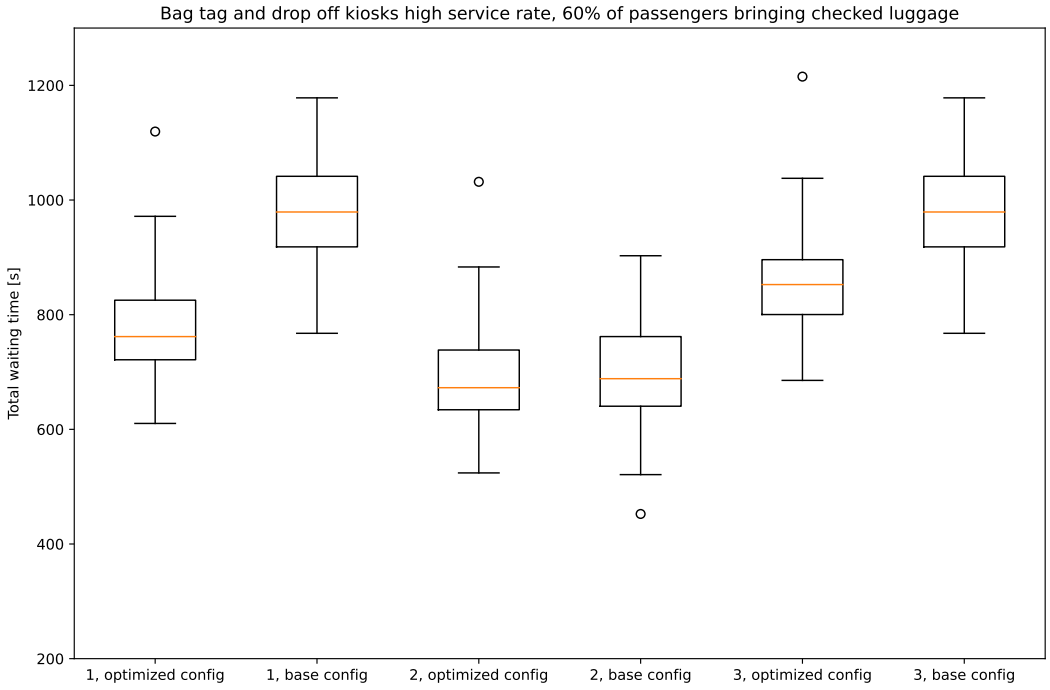


Figure 3.2: Waiting times per passenger type for optimized and base config and setting: (dl:0.6, bt sr: 20 s, do sr 30)

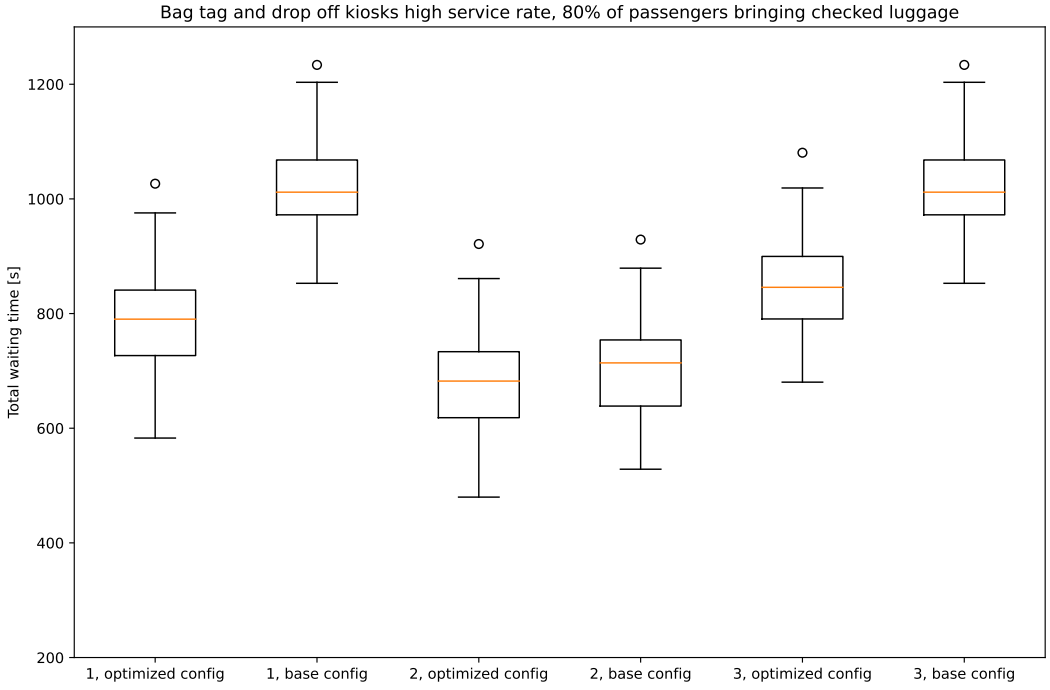


Figure 3.3: Waiting times per passenger type for optimized and base config and setting: (dl:0.8, bt sr: 20 s, do sr 30)

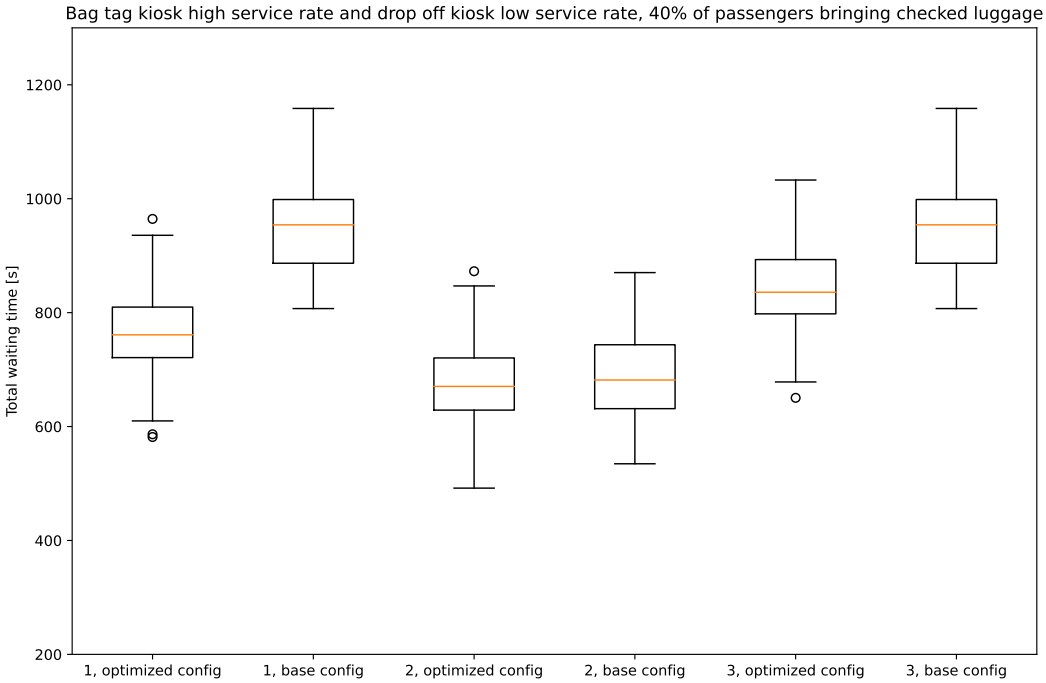


Figure 3.4: Waiting times per passenger type for optimized and base config and setting: (dl:0.4, bt sr: 20 s, do sr 50)

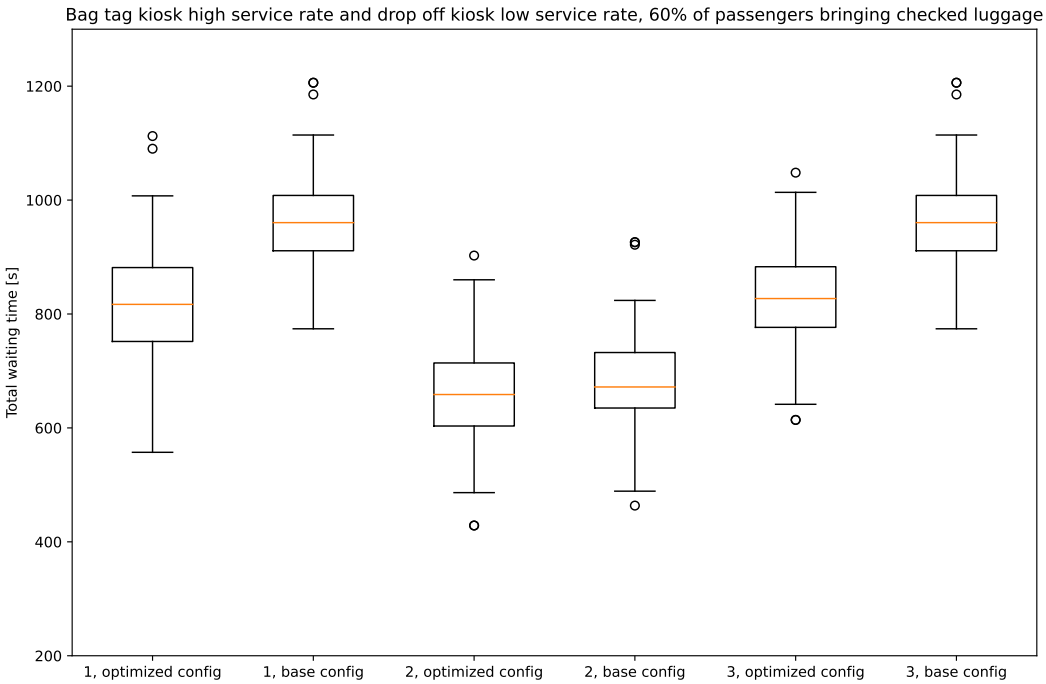


Figure 3.5: Waiting times per passenger type for optimized and base config and setting: (dl:0.6, bt sr: 20 s, do sr 50)

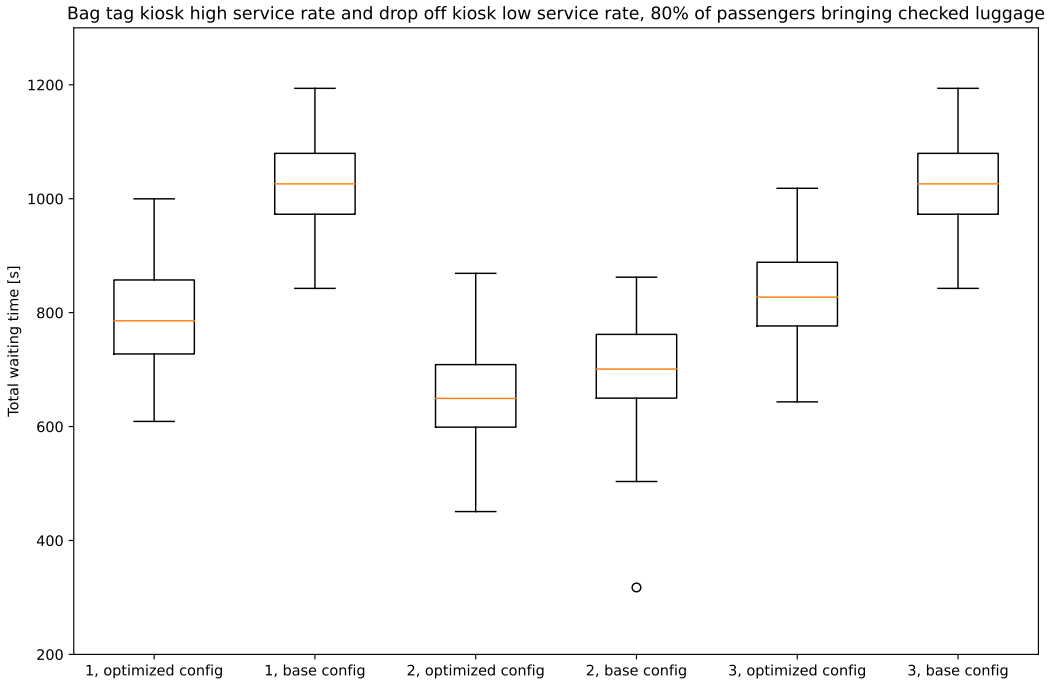


Figure 3.6: Waiting times per passenger type for optimized and base config and setting: (dl:0.8, bt sr: 20 s, do sr 50)

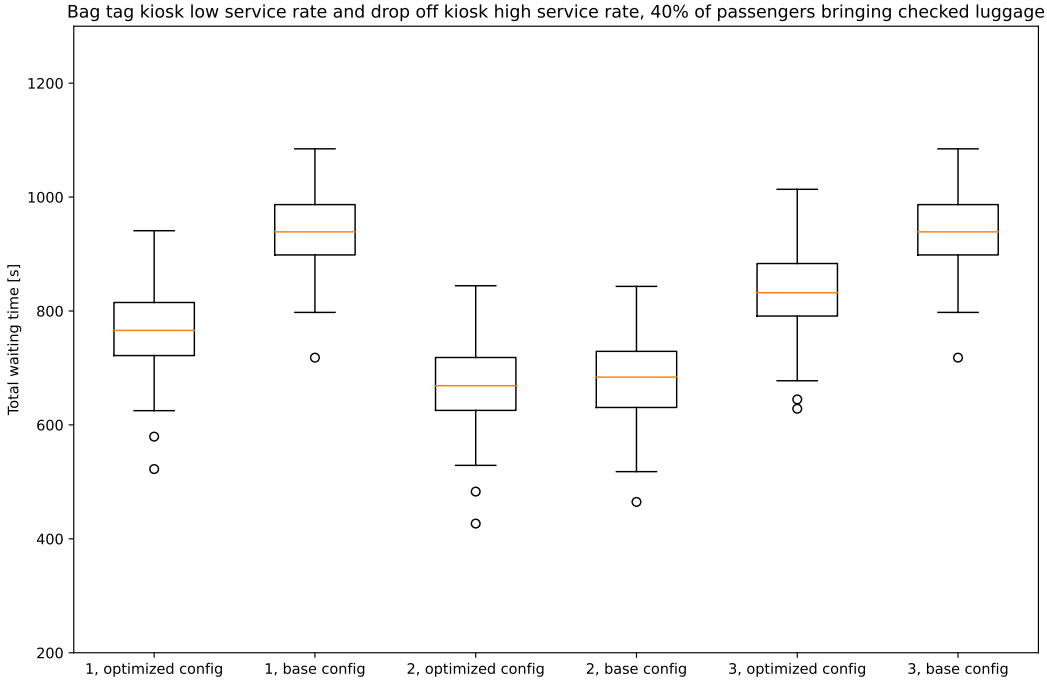


Figure 3.7: Waiting times per passenger type for optimized and base config and setting: (dl:0.4, bt sr: 40 s, do sr 30)

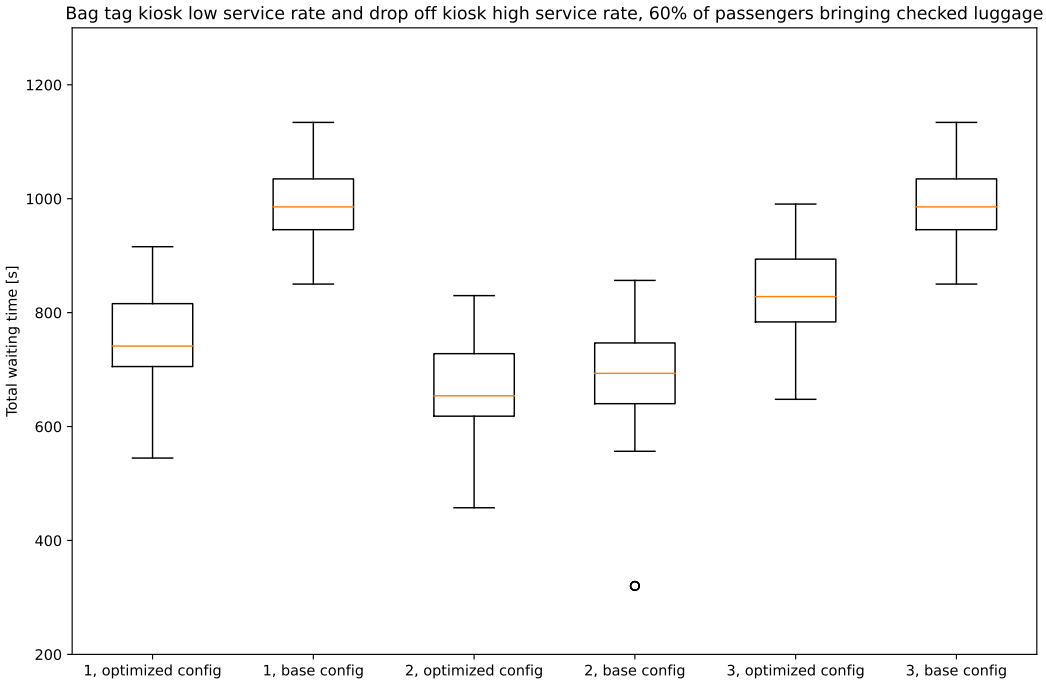


Figure 3.8: Waiting times per passenger type for optimized and base config and setting: (dl:0.6, bt sr: 40 s, do sr 30)

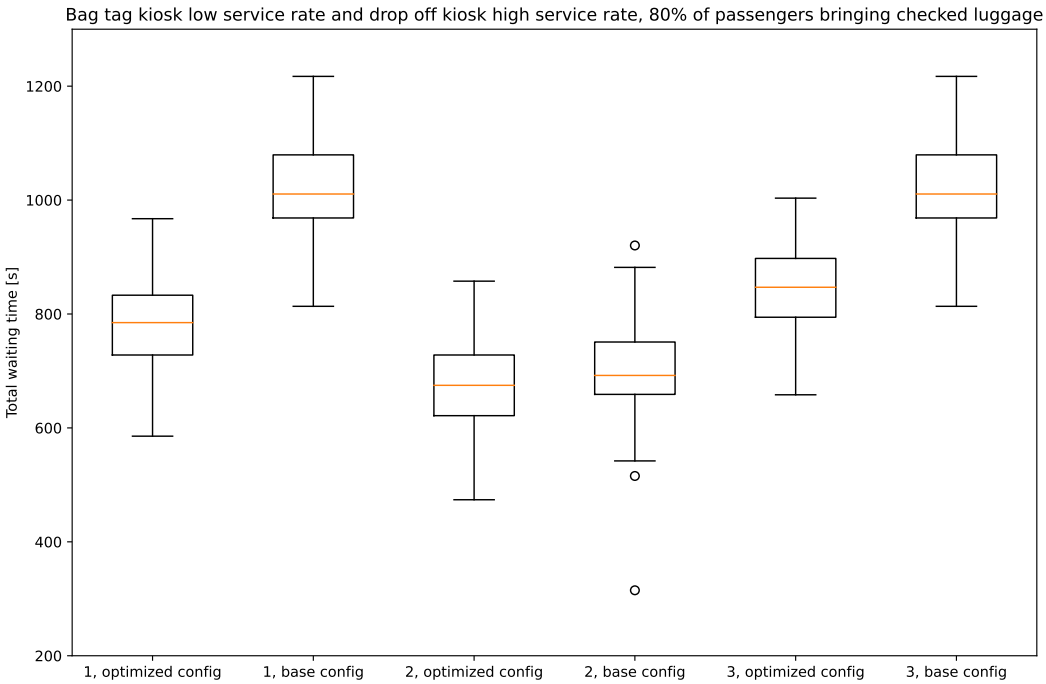


Figure 3.9: Waiting times per passenger type for optimized and base config and setting: (dl:0.8, bt sr: 40 s, do sr 30)

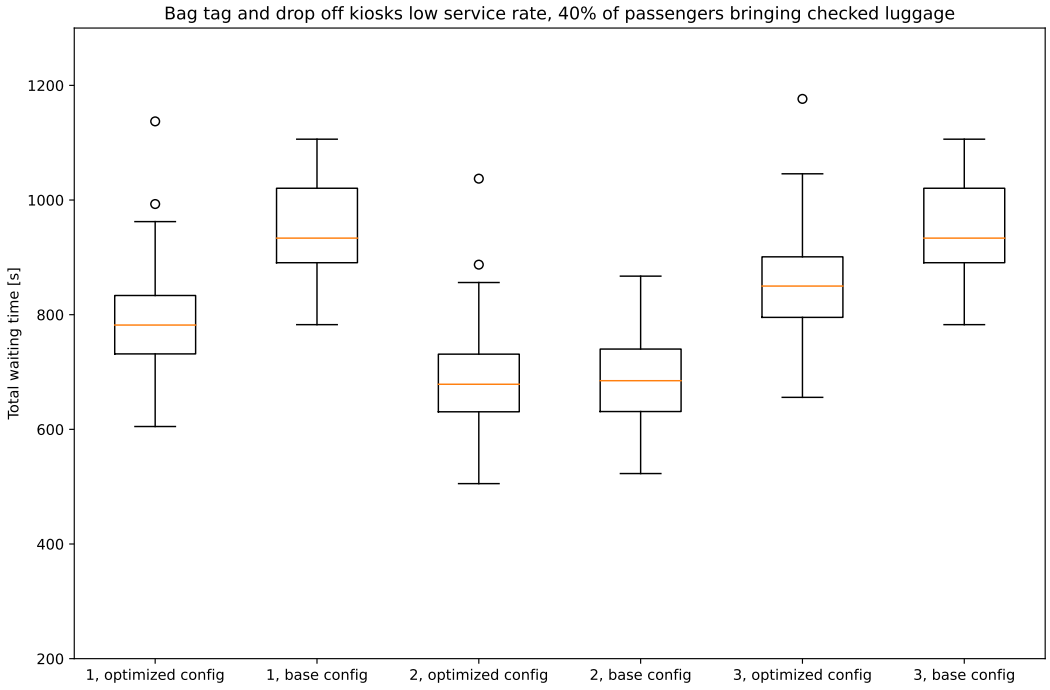


Figure 3.10: Waiting times per passenger type for optimized and base config and setting: (dl:0.4, bt sr: 40 s, do sr 50)

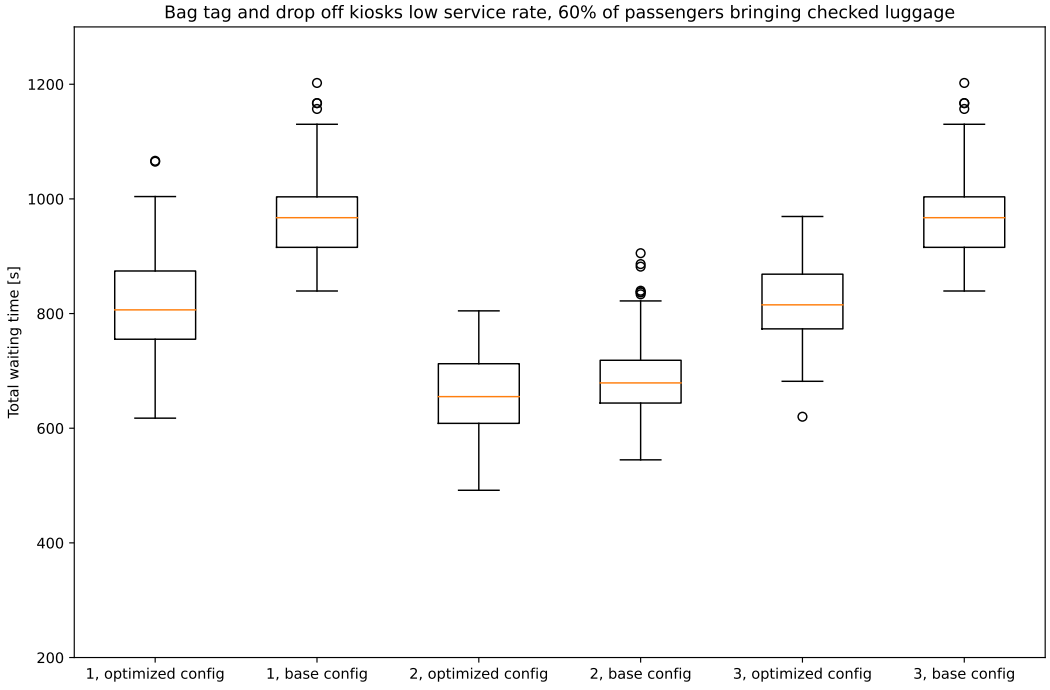


Figure 3.11: Waiting times per passenger type for optimized and base config and setting: (dl:0.6, bt sr: 40 s, do sr 50)

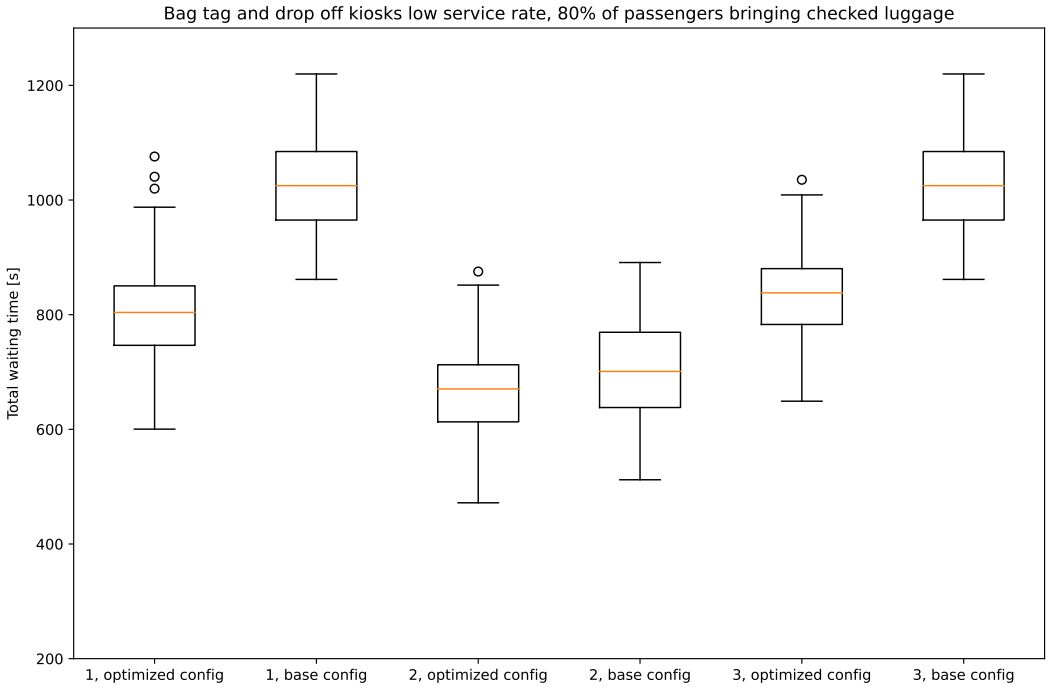


Figure 3.12: Waiting times per passenger type for optimized and base config and setting: (dl:0.8, bt sr: 40 s, do sr 50)

3.1.2. Simheuristic optimization performance

In the tables below, the ranking resulting from the stochastic and deterministic version can be found for each evaluated setting. As discussed in the paper, when the fitness values are close together the ranking resulting from the deterministic version is not the same as the ranking resulting from the stochastic version. For larger differences in fitness value, the ranking is the same indicating that the deterministic version can be used for roughly estimating the fitness of a solution.

Table 3.3: Ranking comparison between deterministic and stochastic version for setting: (dl:0.4, bt sr: 20 s, do sr: 30 s)

Input	Rank from Deterministic Version	Rank from Stochastic Version
[2,0,5,3]	1 (fitness 0.1005)	4 (fitness 0.1388)
[2,3,4,3]	2 (fitness 0.1016)	1 (fitness 0.1389)
[2,5,2,3]	3 (fitness 0.1017)	3 (fitness 0.1399)
[2,2,3,3]	4 (fitness 0.1017)	2 (fitness 0.1405)

Table 3.4: Ranking comparison between deterministic and stochastic version for setting: (dl:0.6, bt sr: 20 s, do sr 30)

Input	Rank from Deterministic Version	Rank from Stochastic Version
[2,0,4,3]	1 (fitness 0.1016)	3 (fitness 0.1388)
[2,2,3,3]	2 (fitness 0.1016)	4 (fitness 0.1388)
[2,3,5,3]	3 (fitness 0.1018)	2 (fitness 0.1397)
[2,0,5,3]	4 (fitness 0.1018)	1 (fitness 0.1374)

Table 3.5: Ranking comparison between deterministic and stochastic version for setting: (dl:0.8, bt sr: 20 s, do sr 30)

Input	Rank from Deterministic Version	Rank from Stochastic Version
[2,2,4,3]	1 (fitness 0.1021)	3 (fitness 0.1392)
[2,4,3,3]	2 (fitness 0.1022)	1 (fitness 0.1391)
[2,0,4,3]	3 (fitness 0.1022)	2 (fitness 0.1391)
[2,3,5,3]	4 (fitness 0.1028)	4 (fitness 0.1403)

Table 3.6: Ranking comparison between deterministic and stochastic version for setting: (dl:0.4, bt sr: 20 s, do sr 50)

Input	Rank from Deterministic Version	Rank from Stochastic Version
[2,0,4,3]	1 (fitness 0.1016)	1 (fitness 0.1384)
[2,4,2,3]	2 (fitness 0.1022)	3 (fitness 0.1392)
[2,3,3,3]	3 (fitness 0.1023)	4 (fitness 0.1397)
[2,5,0,3]	4 (fitness 0.1041)	2 (fitness 0.1389)

Table 3.7: Ranking comparison between deterministic and stochastic version for setting: (dl:0.6, bt sr: 20 s, do sr 50)

Input	Rank from Deterministic Version	Rank from Stochastic Version
[2,3,5,3]	1 (fitness 0.1075)	2 (fitness 0.1446)
[2,2,3,3]	2 (fitness 0.1077)	1 (fitness 0.1436)
[2,1,0,3]	3 (fitness 0.1092)	3 (fitness 0.1455)
[3,1,3,3]	4 (fitness 0.1377)	4 (fitness 0.1750)

Table 3.8: Ranking comparison between deterministic and stochastic version for setting: (dl:0.8, bt sr: 20 s, do sr 50)

Input	Rank from Deterministic Version	Rank from Stochastic Version
[2,0,4,4]	1 (fitness 0.1613)	4 (fitness 0.1991)
[2,0,5,4]	2 (fitness 0.1618)	1 (fitness 0.1967)
[2,2,4,4]	3 (fitness 0.1619)	2 (fitness 0.1979)
[2,1,3,4]	4 (fitness 0.1619)	3 (fitness 0.1981)

Table 3.9: Ranking comparison between deterministic and stochastic version for setting: (dl:0.4, bt sr: 40 s, do sr 30)

Input	Rank from Deterministic Version	Rank from Stochastic Version
[2,2,4,3]	1 (fitness 0.1015)	2 (fitness 0.1384)
[2,3,5,3]	2 (fitness 0.1016)	1 (fitness 0.1384)
[2,4,3,3]	3 (fitness 0.1018)	4 (fitness 0.1395)
[2,5,4,3]	4 (fitness 0.1019)	3 (fitness 0.1385)

Table 3.10: Ranking comparison between deterministic and stochastic version for setting: (dl:0.6, bt sr: 40 s, do sr 30)

Input	Rank from Deterministic Version	Rank from Stochastic Version
[2,3,5,3]	1 (fitness 0.1017)	3 (fitness 0.1384)
[2,0,5,3]	2 (fitness 0.1038)	2 (fitness 0.1384)
[2,0,4,3]	3 (fitness 0.1038)	1 (fitness 0.1395)
[2,2,0,3]	4 (fitness 0.1041)	4 (fitness 0.1398)

Table 3.11: Ranking comparison between deterministic and stochastic version for setting: (dl:0.8, bt sr: 40 s, do sr 30)

Input	Rank from Deterministic Version	Rank from Stochastic Version
[2,5,5,3]	1 (fitness 0.1021)	3 (fitness 0.1402)
[2,2,5,3]	2 (fitness 0.1021)	1 (fitness 0.1392)
[2,3,2,3]	3 (fitness 0.1038)	1 (fitness 0.1396)
[3,3,4,3]	4 (fitness 0.1320)	4 (fitness 0.1696)

Table 3.14: Ranking comparison between deterministic and stochastic version for setting: (dl:0.8, bt sr: 40 s, do sr 50)

Input	Rank from Deterministic Version	Rank from Stochastic Version
[2,5,4,4]	1 (fitness 0.1617)	4 (fitness 0.1989)
[2,3,5,4]	2 (fitness 0.1621)	3 (fitness 0.1985)
[2,2,3,4]	3 (fitness 0.1633)	2 (fitness 0.1981)
[2,0,5,4]	4 (fitness 0.1635)	1 (fitness 0.1960)

Table 3.12: Ranking comparison between deterministic and stochastic version for setting: (dl:0.4, bt sr: 40 s, do sr 50)

Input	Rank from Deterministic Version	Rank from Stochastic Version
[2,5,4,3]	1 (fitness 0.1023)	3 (fitness 0.1395)
[2,3,5,3]	2 (fitness 0.1027)	2 (fitness 0.1394)
[2,2,4,3]	3 (fitness 0.1029)	4 (fitness 0.1405)
[2,2,2,3]	4 (fitness 0.1033)	1 (fitness 0.1382)

Table 3.13: Ranking comparison between deterministic and stochastic version for setting: (dl:0.6, bt sr: 40 s, do sr 50)

Input	Rank from Deterministic Version	Rank from Stochastic Version
[2,4,3,3]	1 (fitness 0.1076)	3 (fitness 0.1446)
[2,4,4,3]	2 (fitness 0.1078)	2 (fitness 0.1442)
[2,4,0,3]	3 (fitness 0.1087)	4 (fitness 0.1449)
[2,2,2,3]	4 (fitness 0.1094)	1 (fitness 0.1438)

3.2. Statistical tests

In this section, the results from the statistical tests performed for the configuration and gate experiment can be found. For both experiments, first the normality of each to be tested data set is determined. If two normally distributed data sets are compared, a t-test is used, otherwise a Wilcoxon signed rank test is used. As a final test, a Vargha-Delaney magnitude test is done to test what the probability is that a randomly selected observation from one sample is larger than a randomly selected observation from the other sample [44].

For the normality, t-test and Wilcoxon signed rank test a threshold p-value of 0.05 is used. For the normality test, if the resulting p-value is lower than the threshold, the null hypothesis is rejected. The null hypothesis states that the data set is normally distributed. For the t-test and Wilcoxon signed rank test, the null hypothesis states that both data sets are sampled from the same statistical distribution.

The Vargha-Delaney magnitude test is interpreted using the guidelines as proposed by Hess and Kromrey [10]. The guidelines can be found in Table 3.15.

Table 3.15: Guideline for interpreting Vargha-Delaney magnitude

test result <0.147	0.147 <test result <0.33	0.33 <test result <0.474	0.474 <test result
Large	Medium	Small	Negligible

3.2.1. Configuration optimization experiment

For the configuration experiment, first the results from the normality tests are shown. Based on this test result, a t-test or Wilcoxon signed rank test is conducted. Last a magnitude test is done to get a complete idea of the statistical significance of the results. The goal of these tests is to investigate whether the impact of the kiosks on airport terminal performance is significant.

Normality test

For the normality test a p-value threshold of 0.05 is used.

Table 3.16: Results of normality test configuration experiment setting:(dl:0.4, bt sr: 20 s, do sr: 30 s)

Passenger type	P-value normality test optimized kiosks configuration	P-value normality test base configuration
1	0.076	0.083
2	0.074	0.43
3	0.048	0.083

Table 3.17: Results of normality test configuration experiment setting:(dl:0.6, bt sr: 20 s, do sr: 30 s)

Passenger type	P-value normality test optimized kiosks configuration	P-value normality test base configuration
1	0.0015	0.41
2	0.0012	0.81
3	0.000046	0.41

Table 3.18: Results of normality test configuration experiment setting:(dl:0.8, bt sr: 20 s, do sr: 30 s)

Passenger type	P-value normality test optimized kiosks configuration	P-value normality test base configuration
1	0.67	0.88
2	0.64	0.83
3	0.73	0.88

Table 3.19: Results of normality test configuration experiment setting:(dl:0.4, bt sr: 20 s, do sr: 50 s)

Passenger type	P-value normality test optimized kiosks configuration	P-value normality test base configuration
1	3.6 e-38	0.25
2	2.1 e-38	0.14
3	7.8 e-39	0.25

Table 3.20: Results of normality test configuration experiment setting:(dl:0.6, bt sr: 20 s, do sr: 50 s)

Passenger type	P-value normality test optimized kiosks configuration	P-value normality test base configuration
1	0.39	0.030
2	0.75	0.068
3	0.91	0.030

Table 3.21: Results of normality test configuration experiment setting:(dl:0.6, bt sr: 20 s, do sr: 50 s)

Passenger type	P-value normality test optimized kiosks configuration	P-value normality test base configuration
1	0.39	0.030
2	0.75	0.068
3	0.91	0.030

Table 3.22: Results of normality test configuration experiment setting:(dl:0.4, bt sr: 40 s, do sr: 30 s)

Passenger type	P-value normality test optimized kiosks configuration	P-value normality test base configuration
1	0.53	0.52
2	0.56	0.51
3	0.53	0.52

Table 3.23: Results of normality test configuration experiment setting:(dl:0.6, bt sr: 40 s, do sr: 30 s)

Passenger type	P-value normality test optimized kiosks configuration	P-value normality test base configuration
1	0.59	2.7 e-28
2	0.62	4.3 e-11
3	0.43	2.7 e-28

Table 3.24: Results of normality test configuration experiment setting:(dl:0.8, bt sr: 40 s, do sr: 30 s)

Passenger type	P-value normality test optimized kiosks configuration	P-value normality test base configuration
1	0.99	7.1 e-34
2	0.97	1.3 e-6
3	0.58	7.1 e-34

Table 3.25: Results of normality test configuration experiment setting:(dl:0.4, bt sr: 40 s, do sr: 50 s)

Passenger type	P-value normality test optimized kiosks configuration	P-value normality test base configuration
1	0.0068	0.0023
2	0.0038	0.12
3	0.034	0.0023

Table 3.26: Results of normality test configuration experiment setting:(dl:0.6, bt sr: 40 s, do sr: 50 s)

Passenger type	P-value normality test optimized kiosks configuration	P-value normality test base configuration
1	0.21	9.1 e-5
2	0.14	0.00062
3	0.92	9.1 e-5

Table 3.27: Results of normality test configuration experiment setting:(dl:0.6, bt sr: 40 s, do sr: 50 s)

Passenger type	P-value normality test optimized kiosks configuration	P-value normality test base configuration
1	0.21	9.1 e-5
2	0.14	0.00062
3	0.92	9.1 e-5

Statistical significance test

Based on the results from the normality tests, a t-test or Wilcoxon signed rank test is chosen. For each passenger type, the p-value has been calculated for the sets of waiting times from the optimized and base configuration. For each setting it becomes apparent that the decrease in waiting time for passenger type one and three

is significant. The waiting time for passenger type two does not experience a significant change in value. A p-value threshold of 0.05 is used for these tests.

Table 3.28: Results of significance test configuration experiment setting:(dl:0.4, bt sr: 20 s, do sr: 30 s)

Passenger type	Test	P-value
1	t-test	1.2 e -31
2	t-test	0.72
3	Wilcoxon signed rank test	2.35 e-13

Table 3.29: Results of significance test configuration experiment setting:(dl:0.6, bt sr: 20 s, do sr: 30 s)

Passenger type	Test	P-value
1	Wilcoxon signed rank test	2.2 e-22
2	Wilcoxon signed rank test	0.31
3	Wilcoxon signed rank test	1.05 e-17

Table 3.30: Results of significance test configuration experiment setting:(dl:0.8, bt sr: 20 s, do sr: 30 s)

Passenger type	Test	P-value
1	t-test	5.75 e-47
2	t-test	0.12
3	t-test	5.63 e-36

Table 3.31: Results of significance test configuration experiment setting:(dl:0.4, bt sr: 20 s, do sr: 50 s)

Passenger type	Test	P-value
1	Wilcoxon signed rank test	3.77 e-21
2	Wilcoxon signed rank test	0.074
3	Wilcoxon signed rank test	1.00 e-16

Table 3.32: Results of significance test configuration experiment setting:(dl:0.6, bt sr: 20 s, do sr: 50 s)

Passenger type	Test	P-value
1	Wilcoxon signed rank test	1.32 e-18
2	Wilcoxon signed rank test	0.029
3	Wilcoxon signed rank test	6.15 e-20

Table 3.33: Results of significance test configuration experiment setting:(dl:0.8, bt sr: 20 s, do sr: 50 s)

Passenger type	Test	P-value
1	Wilcoxon signed rank test	1.88 e-22
2	Wilcoxon signed rank test	8.90 e-6
3	Wilcoxon signed rank test	2.11 e-22

Table 3.34: Results of significance test configuration experiment setting:(dl:0.4, bt sr: 40 s, do sr: 30 s)

Passenger type	Test	P-value
1	t-test	1.72 e-37
2	t-test	0.27
3	t-test	8.34 e-20

Table 3.35: Results of significance test configuration experiment setting:(dl:0.6, bt sr: 40 s, do sr: 30 s)

Passenger type	Test	P-value
1	Wilcoxon signed rank test	9.74 e-23
2	Wilcoxon signed rank test	0.0082
3	Wilcoxon signed rank test	8.92 e-22

Table 3.36: Results of significance test configuration experiment setting:(dl:0.8, bt sr: 40 s, do sr: 30 s)

Passenger type	Test	P-value
1	Wilcoxon signed rank test	1.10 e-22
2	Wilcoxon signed rank test	0.012
3	Wilcoxon signed rank test	4.46 e-22

Table 3.37: Results of significance test configuration experiment setting:(dl:0.4, bt sr: 40 s, do sr: 50 s)

Passenger type	Test	P-value
1	Wilcoxon signed rank test	3.39 e-20
2	Wilcoxon signed rank test	0.56
3	Wilcoxon signed rank test	6.23 e-14

Table 3.38: Results of significance test configuration experiment setting:(dl:0.6, bt sr: 40 s, do sr: 50 s)

Passenger type	Test	P-value
1	Wilcoxon signed rank test	7.59 e-21
2	Wilcoxon signed rank test	0.0026
3	Wilcoxon signed rank test	6.61 e-22

Table 3.39: Results of significance test configuration experiment setting:(dl:0.8, bt sr: 40 s, do sr: 50 s)

Passenger type	Test	P-value
1	t-test	6.53 e-44
2	t-test	0.00025
3	t-test	2.82 e-39

Magnitude test

Below the results from the Vargha-Delaney magnitude test can be found. The guidelines used to interpret these results can be found Table 3.15. For each setting, the waiting times of passenger types one and three experience a large to medium to large difference in magnitude. For passenger type two, this difference is small or negligible.

Table 3.40: Results of magnitude test configuration experiment setting:(dl:0.4, bt sr: 20 s, do sr: 30 s)

Passenger type	Result magnitude test
1	0.079
2	0.48
3	0.20

Table 3.41: Results of magnitude test configuration experiment setting:(dl:0.6, bt sr: 20 s, do sr: 30 s)

Passenger type	Result magnitude test
1	0.038
2	0.46
3	0.13

Table 3.42: Results of magnitude test configuration experiment setting:(dl:0.8, bt sr: 20 s, do sr: 30 s)

Passenger type	Result magnitude test
1	0.023
2	0.43
3	0.063

Table 3.43: Results of magnitude test configuration experiment setting:(dl:0.4, bt sr: 20 s, do sr: 50 s)

Passenger type	Result magnitude test
1	0.041
2	0.44
3	0.15

Table 3.44: Results of magnitude test configuration experiment setting:(dl:0.6, bt sr: 20 s, do sr: 50 s)

Passenger type	Result magnitude test
1	0.11
2	0.42
3	0.11

Table 3.45: Results of magnitude test configuration experiment setting:(dl:0.8, bt sr: 20 s, do sr: 50 s)

Passenger type	Result magnitude test
1	0.031
2	0.33
3	0.038

Table 3.46: Results of magnitude test configuration experiment setting:(dl:0.4, bt sr: 40 s, do sr: 30 s)

Passenger type	Result magnitude test
1	0.045
2	0.46
3	0.15

Table 3.47: Results of magnitude test configuration experiment setting:(dl:0.4, bt sr: 40 s, do sr: 30 s)

Passenger type	Result magnitude test
1	0.045
2	0.46
3	0.15

Table 3.48: Results of magnitude test configuration experiment setting:(dl:0.8, bt sr: 40 s, do sr: 30 s)

Passenger type	Result magnitude test
1	0.012
2	0.41
3	0.047

Table 3.49: Results of magnitude test configuration experiment setting:(dl:0.4, bt sr: 40 s, do sr: 50 s)

Passenger type	Result magnitude test
1	0.086
2	0.48
3	0.20

Table 3.50: Results of magnitude test configuration experiment setting:(dl:0.6, bt sr: 40 s, do sr: 50 s)

Passenger type	Result magnitude test
1	0.075
2	0.39
3	0.053

Table 3.51: Results of magnitude test configuration experiment setting:(dl:0.8, bt sr: 40 s, do sr: 50 s)

Passenger type	Result magnitude test
1	0.030
2	0.37
3	0.039

3.2.2. Gate experiment

For the gate experiment, first the results from the normality tests are shown. Based on this test result, a t-test or Wilcoxon signed rank test is conducted. Last a magnitude test is done to get a complete idea of the statistical significance of the results.

Normality test

In Table 3.52 the results from the normality test for each setting can be found.

Table 3.52: Results of normality test for gate experiment

	P-value of normality test kiosk setup	P-value of normality no kiosk setup
60% large carry on luggage	0.11	0.70
70% large carry on luggage	0.076	0.000020
80% large carry on luggage	0.55	0.83

Statistical significance test

Using the results from the normality tests, a t-test or Wilcoxon signed rank test is used. This tests are used to determine whether there is a significant difference in boarding time for the configuration with and without kiosk. For the 60% setting no significant difference can be found in terms of boarding time. For the 70% and 80% setting a significant difference is found.

Table 3.53: Results of significance test for gate experiment

	Test	P-value
60% large carry on luggage	t-test	0.78
70% large carry on luggage	Wilcoxon signed-rank test	6.2 e-23
80% large carry on luggage	t-test	3.0 e-120

Magnitude test

The results of the Vargha-Delaney magnitude test can be found in Table 3.54. The guidelines used to interpret these results can be found Table 3.15. From the results it becomes clear that if the percentage of passengers carrying carry-on luggage increases, the result of the Vargha-Delaney magnitude decreases.

Table 3.54: Results of magnitude test for gate experiment

	Result magnitude test
60% large carry on luggage	0.50
70% large carry on luggage	0.24
80% large carry on luggage	0.00032

Bibliography

[1] Overcoming hurdles as Fast Travel moves forward apace | Airlines. URL https://airlines.iata.org/analysis/overcoming-hurdles-as-fast-travel-moves-forward-apace.

[2] IEEE Guide for Information Technology - System Definition - Concept of Operations (ConOps) Document. *IEEE Std 1362-1998*, pages 1–24, December 1998. doi: 10.1109/IEEESTD.1998.89424. Conference Name: IEEE Std 1362-1998.

[3] Satyajith Amaran, Nikolaos V. Sahinidis, Bikram Sharda, and Scott J. Bury. Simulation optimization: a review of algorithms and applications. *Annals of Operations Research*, 240(1):351–380, May 2016. ISSN 1572-9338. doi: 10.1007/s10479-015-2019-x.

[4] Bagchain. bagchain gate kiosk. URL https://bagchain.aero/index.php/bagchain-gate/.

[5] Russell R. Barton and Martin Meckesheimer. Chapter 18 Metamodel-Based Simulation Optimization. In Shane G. Henderson and Barry L. Nelson, editors, *Handbooks in Operations Research and Management Science*, volume 13 of *Simulation*, pages 535–574. Elsevier, January 2006. doi: 10.1016/S0927-0507(06)13018-2.

[6] M. Bevilacqua and F. E. Ciarapica. Analysis of check-in procedure using simulation: A case study. In *2010 IEEE International Conference on Industrial Engineering and Engineering Management*, pages 1621–1625, December 2010. doi: 10.1109/IEEM.2010.5674286. ISSN: 2157-362X.

[7] William J. Dunlay and Chang-Ho Park. Tandem-Queue Algorithm for Airport User Flows. *Transportation Engineering Journal of ASCE*, 104(2):131–149, March 1978. doi: 10.1061/TPEJAN.0000697. Publisher: American Society of Civil Engineers.

[8] Samuel Eilon and Stephen Mathewson. A Simulation Study for the Design of an Air Terminal Building. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3(4):308–317, July 1973. ISSN 2168-2909. doi: 10.1109/TSMC.1973.4309241. Conference Name: IEEE Transactions on Systems, Man, and Cybernetics.

[9] Alex Grasas, Angel A Juan, and Helena R Lourenço. SimILS: a simulation-based extension of the iterated local search metaheuristic for stochastic combinatorial optimization. *Journal of Simulation*, 10(1):69–77, February 2016. ISSN 1747-7778. doi: 10.1057/jos.2014.25. Publisher: Taylor & Francis _eprint: https://doi.org/10.1057/jos.2014.25.

[10] Melinda Hess and Jeffrey Kromrey. Robust Confidence Intervals for Effect Sizes: A Comparative Study of Cohen's d and Cliff's Delta Under Non-normality and Heterogeneous Variances. *Paper Presented at the Annual Meeting of the American Educational Research Association*, January 2004.

[11] Robert Hooke and T. A. Jeeves. "Direct Search" Solution of Numerical and Statistical Problems. *Journal of the ACM*, 8(2):212–229, April 1961. ISSN 0004-5411, 1557-735X. doi: 10.1145/321062.321069.

[12] International Air Transport Association. *Airport development reference manual*. IATA, London, 9th ed edition, 2004. ISBN 978-92-9195-086-7.

[13] S. a. M. Janssen, A. Sharpanskykh, R. Curran, and K. G. Langendoen. AATOM - An Agent-based Airport Terminal Operations Model simulator. *SummerSim '19: Proceedings of the 2019 Summer Simulation Conference*, 2019. Publisher: ACM DL.

[14] Stef Janssen, Anne-Nynke Blok, and Arthur Knol. AATOM - An Agent-based Airport Terminal Operations Model. Technical report, 2018.

[15] Stef Janssen, Régis van der Sommen, Alexander Dilweg, and Alexei Sharpanskykh. Data-Driven Analysis of Airport Security Checkpoint Operations. *Aerospace*, 7(6):69, June 2020. ISSN 2226-4310. doi: 10.3390/aerospace7060069. Number: 6 Publisher: Multidisciplinary Digital Publishing Institute.

[16] Angel A. Juan, Javier Faulin, Scott E. Grasman, Markus Rabe, and Gonçalo Figueira. A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems. *Operations Research Perspectives*, 2:62–72, December 2015. ISSN 2214-7160. doi: 10.1016/j.orp.2015.03.001.

[17] Antonín Kazda and Robert E. Caves. *Airport design and operation*. Advances in Mergers and Acquisitions ; Volume 14. Emerald, Bingley, third edition. edition, 2015. ISBN 978-1-78441-869-4 1-78441-869-2 978-1-5231-1347-7 1-5231-1347-2.

[18] Seong-Hee Kim and Barry L. Nelson. Recent advances in ranking and selection. In *2007 Winter Simulation Conference*, pages 162–172, December 2007. doi: 10.1109/WSC.2007.4419598. ISSN: 1558-4305.

[19] Wonkyu Kim, Yonghwa Park, and Byung Jong Kim. Estimating hourly variations in passenger volume at airports using dwelling time distributions. *Journal of Air Transport Management*, 10(6):395–400, November 2004. ISSN 0969-6997. doi: 10.1016/j.jairtraman.2004.06.009.

[20] A.S. Kiran, T. Cetinkaya, and S. Og. Simulation modeling and analysis of a new international terminal. In *2000 Winter Simulation Conference Proceedings (Cat. No.00CH37165)*, volume 2, pages 1168–1172 vol.2, December 2000. doi: 10.1109/WSC.2000.899081.

[21] Franziska Klügl and Ana L. C. Bazzan. Agent-Based Modeling and Simulation. *AI Magazine*, 33(3):29–29, September 2012. ISSN 2371-9621. doi: 10.1609/aimag.v33i3.2425. Number: 3.

[22] Tamara G. Kolda, Robert Michael Lewis, and Virginia Torczon. Optimization by Direct Search: New Perspectives on Some Classical and Modern Methods. *SIAM Review*, 45(3):385–482, 2003. ISSN 0036-1445. Publisher: Society for Industrial and Applied Mathematics.

[23] R. Koray Kyld and M. Karasahin. The capacity analysis of the check-in unit of Antalya Airport using the fuzzy logic method. *Transportation Research Part A: Policy and Practice*, 42(4):610–619, May 2008. ISSN 0965-8564. doi: 10.1016/j.tra.2008.01.004.

[24] Claudia Lehmann. *Exploring Service Productivity: Studies in the German Airport Industry*. Springer Fachmedien Wiesbaden, Wiesbaden, 2019. ISBN 978-3-658-23035-7 978-3-658-23036-4. doi: 10.1007/978-3-658-23036-4.

[25] Andrew C. Lemer. Measuring performance of airport passenger terminals. *Transportation Research Part A: Policy and Practice*, 26(1):37–45, 1992. Publisher: Elsevier.

[26] Stefano Lucidi and Marco Sciandrone. On the Global Convergence of Derivative-Free Methods for Unconstrained Optimization. *SIAM Journal on Optimization*, 13(1):97–116, January 2002. ISSN 1052-6234. doi: 10.1137/S1052623497330392. Publisher: Society for Industrial and Applied Mathematics.

[27] Rogers Lui, Ravinder Nanda, and James J. Browne. International passenger and baggage processing at john f. kennedy international airport. *IEEE Transactions on Systems, Man, and Cybernetics*, (2):221–225, 1972. Publisher: IEEE.

[28] Sean Luke. *Essentials of Metaheuristics*. 2009.

[29] Ioanna E. Manataki and Konstantinos G. Zografos. A generic system dynamics based tool for airport terminal performance analysis. *Transportation Research Part C: Emerging Technologies*, 17(4):428–443, August 2009. ISSN 0968-090X. doi: 10.1016/j.trc.2009.02.001.

[30] J. A. Nelder and R. Mead. A Simplex Method for Function Minimization. *The Computer Journal*, 7(4):308–313, January 1965. ISSN 0010-4620. doi: 10.1093/comjnl/7.4.308.

[31] G.F. Newell. *Applications of queueing theory*. Monographs on statistics and applied probability. Chapman and Hall, London, 2nd ed edition, 1982. ISBN 978-0-412-24500-8. Section: xvi, 303 p. : illustrations ; 22 cm.

[32] Amedeo R. Odoni and Richard de Neufville. Passenger terminal design. *Transportation Research Part A: Policy and Practice*, 26(1):27–35, January 1992. ISSN 0965-8564. doi: 10.1016/0965-8564(92)90042-6.

[33] Paul Roling. Landside capacity, February 2021.

[34] Rotterdam The Hague Airport. Ons verhaal in feiten en cijfers. Technical report, 2021. URL https://www.rotterdamthehagueairport.nl/wp-content/uploads/Ons-verhaal-in-feiten-en-cijfers-2021.pdf.

[35] Michael Schultz and Hartmut Fricke. Managing Passenger Handling at Airport Terminals. *Ninth USA/Europe Air Traffic Management Research and Development Seminar (ATM2011)*, 2011.

[36] Alexei Sharpanskykh and Jan Treur. A Temporal Trace Language for Formal Modelling and Analysis of Agent Systems. In *Specification and Verification of Multi-agent Systems*, pages 317–352. Springer New York, NY, July 2010. ISBN 978-1-4419-6983-5. doi: 10.1007/978-1-4419-6984-2_11.

[37] S. P. Singh and R. R. K. Sharma. A review of different approaches to the facility layout problems. *The International Journal of Advanced Manufacturing Technology*, 30(5-6):425–433, September 2006. ISSN 0268-3768, 1433-3015. doi: 10.1007/s00170-005-0087-9.

[38] Senay Solak, John-Paul B. Clarke, and Ellis L. Johnson. Airport terminal capacity planning. *Transportation Research Part B: Methodological*, 43(6):659–676, July 2009. ISSN 0191-2615. doi: 10.1016/j.trb.2009.01.002.

[39] Takakuwa and Oyama. Simulation analysis of international-departure passenger flows in an airport terminal. In *Proceedings of the 2003 Winter Simulation Conference, 2003.*, volume 2, pages 1627–1634 vol.2, December 2003. doi: 10.1109/WSC.2003.1261612.

[40] Nursyuhada Taufik and Mohd Hafiz Hanafiah. Airport passengers' adoption behaviour towards self-check-in Kiosk Services: the roles of perceived ease of use, perceived usefulness and need for human interaction. *Heliyon*, 5(12):e02960, December 2019. ISSN 2405-8440. doi: 10.1016/j.heliyon.2019.e02960.

[41] Vojin Toi. A review of airport passenger terminal operations analysis and modelling. *Transportation Research Part A: Policy and Practice*, 26(1):3–26, January 1992. ISSN 0965-8564. doi: 10.1016/0965-8564(92)90041-5.

[42] Michael W. Trosset. On the Use of Direct Search Methods for Stochastic Optimization. June 2000. Accepted: 2018-06-18T17:48:14Z.

[43] Didier van der Horst. An improved Tabu Search for optimising the configuration of an agent-based simulation model of a novel security checkpoint. Master's thesis, TU Delft, 2021.

[44] András Vargha and Harold D. Delaney. A Critique and Improvement of the CL Common Language Effect Size Statistics of McGraw and Wong. *Journal of Educational and Behavioral Statistics*, 25(2):101–132, June 2000. ISSN 1076-9986. doi: 10.3102/10769986025002101. URL https://doi.org/10.3102/10769986025002101. Publisher: American Educational Research Association.

[45] Ashish Verma, Divyakant Tahlyan, and Shubham Bhusari. Agent based simulation model for improving passenger service time at Bangalore airport. *Case Studies on Transport Policy*, 8(1):85–93, March 2020. ISSN 2213-624X. doi: 10.1016/j.cstp.2018.03.001.

[46] Diane Wilson, Eric K. Roe, and S. Annie So. Security Checkpoint Optimizer (SCO): An Application for Simulating the Operations of Airport Security Checkpoints. In *Proceedings of the 2006 Winter Simulation Conference*, pages 529–535, December 2006. doi: 10.1109/WSC.2006.323126. ISSN: 1558-4305.

[47] Andreas Wittmer. Acceptance of self-service check-in at Zurich airport. *Research in Transportation Business & Management*, 1(1):136–143, August 2011. ISSN 2210-5395. doi: 10.1016/j.rtbm.2011.06.001.

[48] Paul Pao-Yen Wu and Kerrie Mengersen. A review of models and model usage scenarios for an airport complex system. *Transportation Research Part A: Policy and Practice*, 47:124–140, January 2013. ISSN 0965-8564. doi: 10.1016/j.tra.2012.10.015.