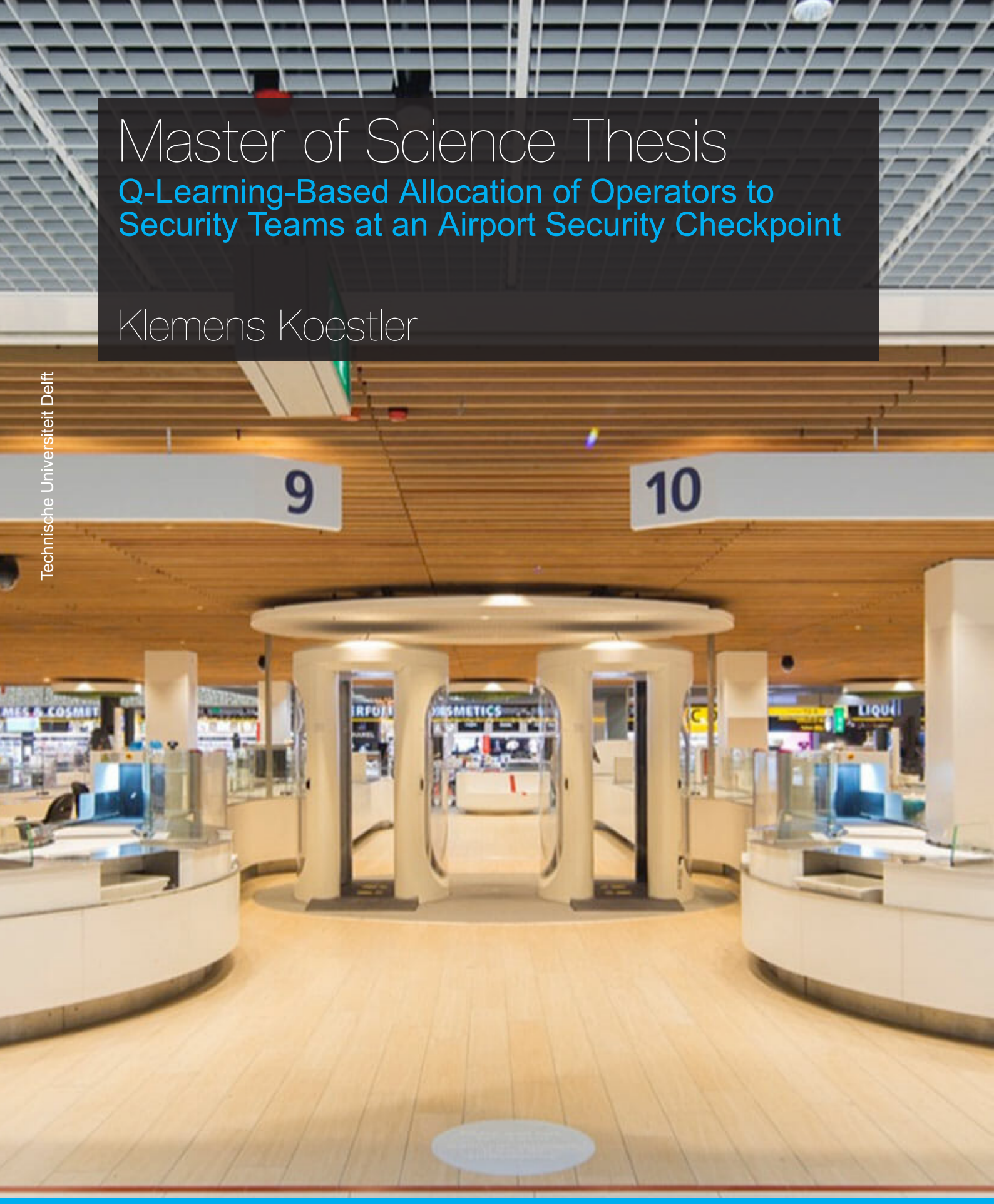
Master of Science Thesis

Q-Learning-Based Allocation of Operators to Security Teams at an Airport Security Checkpoint

Klemens Koestler

Technische Universiteit Delft



TU Delft

Schiphol

Delft University of Technology

Master Thesis

Q-Learning-Based Allocation of Operators to Security Teams at an Airport Security Checkpoint.

Author:
Klemens Koestler

Thesis Comittee:

Chairman	Dr. D. Ragni	TU Delft
Supervisor	Dr. O.A. Sharpans'kykh	TU Delft
Supervisor	J. Mutsaers	Schiphol
Supervisor	J. Veenema	Schiphol
Examiner	Dr. A. Bombelli	TU Delft

A thesis submitted in fulfillment of the requirements for the degree of Master of Science in the

Air Transport Operations Group
Aerospace Faculty

November 5, 2021

TU Delft

Acknowledgements

This thesis marks the final step in my journey to obtain a Master of Science degree in Aerospace Engineering at Delft University of Technology. It is unclear when this journey began, since, after all, there are many starting points to choose from. The reality is that this journey, regardless of starting point, has been filled with lots of laughter, some sorrow, plenty of excitement, occasional confusion and fulfillment. To all of the amazing people that have guided, accompanied and motivated me along the way: thank you!

Thank you to my supervisor, Dr. Alexei Sharpanskykh. Not only did you guide and support me during my thesis, but also for my bachelor group project and honors research project. Through your supervision, you taught me how to be a better researcher. Through your expertise and enthusiasm, you inspired me to learn about programming, data science and artificial intelligence. Your influence has opened many opportunities for me, which I am very grateful for.

Thank you to my team at Amsterdam Airport, Jasper Mutsaers and Jornt Venema, for an unforgettable work experience that I will never forget. Your technical insight into the simulator have pushed this research to the next level.

Thank you to Didier for the thesis collaboration. From the countless hours conducting security measurements at the airport to all the days spent developing and debugging the security checkpoint simulator, you have pushed and motivated me to always give it my all. I am glad to have found a new friend.

Thank you to Andrea, Jacopo and Jasper for making the hard times a lot easier and the good times even better. Our shenanigans and jokes are something I will cherish forever.

Thank you to the *Castle*, *IDEA League*, *OGs*, *Wien Überbleibsl* for being amazing friends and celebrating all of the exciting moments of my life with me.

Thank you to Monika for always being there for me and never failing to put a smile on my face.

Last but not least, thank you to my family for the endless love and support. I could not have achieved this accomplishment without you all.

Klemens Koestler
Vienna, November 2021

Contents

Introduction

Airports are nodes that connect transportation between the ground and air. They accommodate departure, arrival and transfer flows of passengers, where each flow consists of several sub-processes and facilities. Their aim is to offer outstanding quality of service to passengers along each of these processes, focusing on safe, efficient and memorable travel experiences. An essential airport process is the security checkpoint, which controls access between public and restricted areas. Every passenger along with their luggage must be screened to avoid the carrying of banned articles into restricted areas within an airport and on board the aircraft. During this process, there are many interactions between passengers, operators and technological assets. Consequently, airport managers are not able to predict the expected performance of a security checkpoint for different scenarios. Therefore, research has focused on simulating airport security checkpoint to aid airport managers in planning and decision making [87].

A critical part to the operation of a security checkpoint is the shift scheduling of operators. Most airports plan shifts based on the number of teams needed to operate a lane such that the demand arising from the departure profile is accommodated. Research has focused into developing algorithms in combination with a simulation model of a security checkpoint that are able to optimize the shift schedules [73]; [38]; [68]. However, these studies do not consider adjusting the size of the teams for a security lane in order to improve its operational efficiency.

The purpose of this investigation is to *develop and investigate a q-learning based approach for allocation of individual operators to security teams in order to improve operational efficiency of a security checkpoint*. The research approach consists of two parts. First, an agent-based model is developed to simulate the security checkpoint. The model is calibrated and validated in collaboration with Amsterdam Airport Schiphol. Second, a learning agent is introduced into the model, whose goal is to allocate individual operators to security teams during the operations.

The thesis is composed of three parts. Part I presents the scientific paper investigating the research objective. Part II consists of a literature study that supports the purpose of the research. Lastly, Part III provides supporting work related to the scientific paper, which includes: model elaboration

I

Scientific Paper

Q-Learning-based Allocation of Operators to Security Teams At an Airport Security Checkpoint

Klemens Koestler*

Delft University of Technology, Delft, The Netherlands

Abstract

In this paper, we propose and analyze a q-learning-based approach for allocation of operators to security teams in order to improve operational efficiency of an airport security checkpoint. The research is composed of two parts. First, we develop an agent-based model capable of simulating an airport security checkpoint. Second, we introduce learning agents into the model, whose goal is to allocate operators to a security team during operations, to improve operational efficiency of the security checkpoint. We propose two learning activities of these agents. Activity 1 allocates operators to the recheck process, where operators are responsible for reexamining luggages that have been rejected and decide if they are safe or not. Activity 2 allocates operators to the CT process, where operators are responsible for examining CT images of luggages and decide if a luggage should be rechecked or not. We demonstrate that introducing a learning agent with either activity increases the throughput of the security checkpoint. Furthermore, activity 1 and activity 2 decrease the time spent in critical operations for the recheck process and CT process, respectively. The behavioural strategies learned by the agents were to add an operator when there is excess luggage waiting and remove an operator when there are excess operators available. Policy evolution between two different learning agents was compared by determining the similarity in their state transition networks per episode. The similarity was computed using the DeltaCon method and proved to be a promising technique for identifying differences in agent behaviour. This study appears to be the first to dynamically-schedule security operator shifts using a reinforcement learning approach. Insights gained from this study demonstrate that dynamically allocating operators to a security lane improves its operational efficiency, which opens the possibility of dynamic-scheduling security operators for entire terminals. Furthermore, it may aid airport managers in creating more resilient security checkpoints.

Keywords: Airport, Security Checkpoint, Agent-Based Modelling, Reinforcement Learning, Q-Learning, Operator Allocation

1 Introduction

Air transportation is one of the backbones to modern society and an important enabler for globalization. In 2018, the expenditure on air transport alone was 1% of the worlds GDP, equivalent to $845 billion [1]. In this critical infrastructure, airports make up central nodes, where the circulation of people, resources and capital are managed. Airports are constantly evaluating their terminal procedures to ensure safety at the airport and on-board the aircraft, while maintaining a high level of efficiency. A high level of efficiency implies minimal waiting times for each airport terminal process, thereby creating constant flow of passengers/luggage from terminal entrance to gates. Since the processes are affected by human behaviour, an airport is considered to be a complex system. Consequently, one of the greatest challenges for airport managers is predicting the expected performance of each airport terminal process for future scenarios. Predicting performance of a process for different scenarios is important for planning capacity and infrastructure demands. To address this challenge, simulation has emerged as powerful tool to aid airport managers in planning and decision making [2]. Hence, the last decades have seen a growing trend in literature towards airport modelling and simulation.

Existing landside airport models have focused on three aspects of the airport terminal: capacity, efficiency and security [2]. Capacity models are used in strategic planning to answer whether airport infrastructure satisfies future demand [3, 4]. Efficiency models are used in operational planning to develop optimal solutions to operational problems [5, 6]. Lastly, security models are used in investigations of security effectiveness, where the capability to identify and mitigate threats of an activity or subprocess is determined [2, 7]. Early research into landside airport models has focused on entire airport terminals. However, recent advancements in computational power has seen the rapid development of more detailed models that simulate micro interactions in individual airport processes (e.g., check-in, security checkpoint, etc.). The modelling paradigm used for simulating airports

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

varies between research since the choice depends on the research question [8]. Common modelling paradigms for capacity and efficiency models are discrete-event simulation and agent-based simulation [9, 10]. On the contrary, probabilistic methods and fuzzy logic are common modelling paradigms for security models [11, 12].

To date, most research on security checkpoints has neglected the efficiency aspect and instead focused on safety and security. Therefore, very little is currently known about factors influencing the efficiency of a security checkpoint. The few studies that have developed efficiency models of the security checkpoint focus on strategies for the management of security checkpoint operations, such as operator scheduling [13–15]. Research on operator scheduling develops optimization algorithms in combination with an efficiency model of a security checkpoint that establish shift schedules for operators. Shift schedules are based on the number of teams needed to operate a lane such that the demand arising from the departure profile is accommodated. Current research on operator scheduling has focused on static-scheduling, where schedules are made before simulation [14, 15]. Therefore, they require complete prior knowledge about the checkpoint characteristics, which is difficult to obtain in an uncertain environment. By contrast, the use of dynamic-scheduling for operator shift schedules has not been investigated. Dynamic-scheduling focuses on adapting schedules to an evolving task scenario [16]. For example, adjusting the size of the operators teams in a security lane during operations to improve the lanes operational efficiency. Reinforcement Learning has emerged as a suitable paradigm to solve dynamic-scheduling problems due to its ability to learn and cope with uncertain, dynamic environments [17].

This research aims to address the issues in operator scheduling by *developing and investigating a q-learning-based approach for the allocation of operators to security teams at an airport security checkpoint in order to improve the operational efficiency.* In order to accomplish our objective, the following methodological steps are taken. We develop an efficiency model for the security checkpoint using agent-based simulation. The reasons for selecting an agent-based simulation approach were that models can be developed without having knowledge about system interdependencies and that agents with learning capability can be added. Next, we introduce a learning agent into the model, whose goal is to allocate operators to a security team during operations. Data collected from Amsterdam Airport Schiphol is used to calibrate the model. Model validation is performed using a two-step approach. First, security experts from Amsterdam Airport Schiphol evaluate the realism of the agent interactions and emerging phenomenon, such as the pattern of bottleneck transitions during operations. Second, simulated performance of the security checkpoint is compared to day-to-day operations at Amsterdam Airport Schiphol. The contributions of our research are as follows: introduction of an agent-based efficiency model that is capable of simulating different airport security checkpoint configurations; advancement of knowledge on factors affecting the efficiency of a security checkpoint; and introduction of dynamic-scheduling approach for security operators using reinforcement learning.

The paper is structured as follows. Section 2 discusses the related work on landside airport terminal modelling. Section 3 describes the agent-based efficiency model of the security checkpoint. Section 4 introduces learning agents into the model, whose goals are to allocate operators to a security team in order to improve operational efficiency of a security checkpoint. Section 5 presents the experiments and results. Section 6 discusses the results and limitations of the research. Lastly, Section 7 summarizes key findings and suggests possible future work.

2 Related Work

Airports are nodes that connect transportation between the ground and air. They accommodate departure, arrival and transfer flows of passengers, where each flow consists of several processes and facilities. Their aim is to offer excellent quality of service to passengers for each of these processes, focusing on safe and efficient travel experiences. Since these facilities and processes are affected by human behaviour, an airport is considered to be a socio-technical complex system. Therefore, airport terminals are inherently difficult to model, which has led to a plethora of airport terminal research.

Section 2.1, Section 2.2 and Section 2.3 review capacity, efficiency and security airport terminal models respectively. Section 2.4 summarizes the research gap in airport terminal literature.

2.1 Capacity Models

Early examples of research into airport terminals include capacity models, which determine whether planned infrastructure satisfies future demand [3]. Newell (1982) [18] was the first to develop a range of capacity-driven airport terminal models based on the deterministic queue model. These models track key performance indicators such as passenger waiting and service times, and the number of passengers in queue per queuing area within the airport. The simple queue model is extended by Tosic et al. (1983) [4], who investigates a stochastic queue model where arrival profiles are already developed. Monte Carlo methods are used to calculate the total service and wait times and the length of the queue. [19] expands on a single queuing system by adopting a multi-channel queuing approach for a departing passenger at an airport terminal.

By using the process rates from the queue models and combining them with collected data at airports, probabilistic models can be developed that estimate capacity related performance indicators. Kim et al. (2004) [20] defines a mathematical model focused on the probability density functions of dwelling time to estimate passenger volumes for departing passengers over the span of a day. Solak et al. (2009) [21] specifies delay functions that estimate the maximum passenger delay for airport activities and processing infrastructure. These functions are then incorporated in a stochastic programming model based on a multi commodity flow network to simulate an entire airport terminal. However, this approach can not be applied across airports because delay functions are unique to each airport.

2.2 Efficiency Models

Supplementary to capacity models, early research also explored efficiency models, which focused on operational planning and design [2]. These models simulate passenger flow through the airport terminal and require greater level of detail than capacity models. Initial studies focused on discrete-event approaches to simulate an entire airport terminal, by representing airport activities as individual event nodes [5, 6, 22, 23]. Model outputs constituted of queue lengths, process times, passenger delays and congestion of airport facilities. The above-mentioned models designate each passenger to a facilitation area, thereby disregarding the spatial-temporal aspect of passenger flow. Later models would address this issue by assigning pre-determined paths to passengers via a spatial node [24, 25].

As computational resources became more readily available, research on efficiency models gained traction due to their ability to not just aid capacity problems, but also management of daily operations and airport design [2]. Further advancement of these discrete-event models were made by incorporating new airport terminal processes or increasing the level of detail. Gatersleben and Weij (1999) [26] along with Fayez et al. (2008) [27] included arriving/transferring passenger flows, while Zografos and Madas (2006) [9] incorporated airside arrivals/departures. Meanwhile, Joustra and Dijk (2001) [28] along with Appelt et al. (2007) [29] simulated solely the check-in process of an airport terminal. Due to the continuous improvements in computational power, the trend of increasing the complexity of discrete-event models remains to date. This can be seen in the studies of [30–32].

However, the last two decades has also seen the introduction of a new modelling approach for efficiency models of airport terminals, known as agent-based modelling [2]. This approach focuses on the interactions and operations of multiple agents (i.e., passengers or security operators) with each other and the environment (i.e., the airport terminal). The agent interactions lead to the emergence of higher-level system properties. The first to present a simplistic agent-based model for an airport terminal facility was Wilson et al. (2006) [33], focusing on the security checkpoint. Agents follow a fixed activity-oriented schedule and have perception/reasoning capabilities that prevent collisions and allow them to decide what route to take. Schultz and Fricke (2011) [10] describe the first agent-based model that is associated with artificial intelligence. Each agent has an operational, tactical and emergency planning level that dictate its decision making. Their work demonstrates that human behaviour is an important factor when considering the operations of airport facilities. Thereafter, several studies adopted an agent-based approach to simulate airport terminals. Cheng et al. (2014) [34] explored how group behaviour affects passenger flows. Janssen et al. (2019) [35] assessed security, efficiency and the relation between them at airport terminals under different operating conditions. Verma et al. (2020) [36] analyzed passenger service times under various proposed policies aimed at improving throughput of the airport terminal.

Despite the benefits agent-based approaches had to offer, a search of the literature revealed that discrete-event models were more common. A possible explanation for this are the limited number of agent-based airport simulation software that exist [37]. In general, the choice of modelling paradigm depends on the research question [8].

2.3 Security Models

Lastly, another aspect of airport terminals that has gained significant attention are security models. These models simulate airport procedures, technologies and the operators ability to identify threats [2]. The two main approaches to security models are probabilistic methods and fuzzy methods. To date, several studies focusing on probabilistic methods have investigated the likelihood of a prohibited item or malicious person reaching an aircraft for the departure process, commonly known as missed detection [11, 38, 39]. However, probabilities are difficult to assess due to the inherent nature of the problem, namely that missed detection or terrorist threats do not occur often [12, 40]. Instead of determining probabilities to define the safety of security processes, fuzzy methods can be used to evaluate the effectiveness by expressing input values as linguistic variables [41]. Fuzzy approaches have analyzed cabin baggage screening [12, 42], passenger screening [41] and airport security operators [7].

Recently, research on airport security models has started to focus on the efficiency of security processes. Most of these studies focus on strategies for the management of security checkpoint operations. The standard

methodology is to create an efficiency model for the security checkpoint and then develop algorithms for management of the operations [14]. Soukour et al. (2012) [13] presented a staff scheduling problem at the security checkpoint and solved it using a three-step algorithm focusing on scheduling day offs, shift scheduling and assigning staff. Kierzkowski and Kisiel (2017) [14] tackled the same problem but used an algorithm that establishes a capacity lack margin at the security checkpoint. This lowered the operational cost of the checkpoint while maintaining an acceptable quality of service for the passengers.Ruiz and Cheu (2020) [15] evaluated the performance of the security checkpoint under two operational policies: increase number of security operators or reducing the number of screening lanes.

2.4 Research Gap

Overall, a number of limitations and opportunities of airport terminal literature have been identified. To date, research has focused on higher-level system properties (e.g., process and wait times) of airport facilities using discrete-event modelling paradigm. By contrast, few studies investigate lower-level system properties (e.g., asset utilisation) of airport facilities. Previous studies on airport terminal capacity/efficiency models lack standardized key performance indicators. Efficiency models have not treated the security checkpoint in much detail. Most studies on security checkpoints have neglected the efficiency aspect and focused on security. Therefore, very little is currently known about factors influencing the efficiency of a security checkpoint. The few studies that have developed efficiency models for security checkpoints use it as a tool to optimize operational costs for resource allocation problems, such as the staff scheduling problem. However, such approaches have failed to address how to dynamically allocate operators during day-to-day operations of the security checkpoint in order to improve operational efficiency. In general, dynamic-scheduling approaches have not been investigated in operator scheduling.

Therefore, an efficiency model is necessary to simulate new security checkpoint configurations that can evaluate their performance through standardized key performance indicators. Furthermore, this model needs to have the capability of allocating new security operators during operations. In our work, we expand on an existing agent-based airport terminal simulator (AATOM [37]) to simulate various security checkpoint configurations and investigate a reinforcement learning technique that can dynamically allocate security operators optimally during operations. The reason behind choosing an agent-based simulation approach was threefold. First, an agent-based approach captures detailed complex structures and dynamics, which is beneficial when modelling a socio-technical complex system, such as an airport. Second, an agent-based approach allows to construct models by specifying individual behaviour and interactions. Therefore, it does not require knowledge of the system (i.e., security checkpoint) interdependencies. Lastly, an agent-based approach allows the implementation of agents with learning capability. Furthermore, a reinforcement learning approach was chosen because it has emerged as a suitable paradigm to solve dynamic-scheduling problems because they are able to learn and cope in uncertain environment, they are computationally efficient and they are adaptive [17].

3 Modelling and Simulation of an Airport Security Checkpoint

All civil airports in the European Union must adhere to the common rules of civil aviation security in order to prevent acts of unlawful interference [43]. These rules require each airport to establish areas (e.g., landside, airside and restricted), to which access must be controlled such that no unauthorised people and vehicles enter these areas. Therefore, every passenger along with their luggage must be screened to avoid the carrying of banned articles into restricted areas within an airport and on board the aircraft. Airports implement this by having a security checkpoint that controls access between public and restricted areas. The processes within a security checkpoint and factors affecting the efficiency are explained in Section 3.1. Then, an efficiency model for the security checkpoint is developed in Section 3.2.

3.1 Security Checkpoint Operations

Before a passenger can enter the checkpoint, they are required to scan their boarding pass. Once scanned, the passenger enters the banklining, where they will queue until they reach its endpoint. Next, a security operator assigns the passenger to a security lane. When the passenger arrives at the lane, a divest positions is allocated to them for dropping their cabin luggage. At this point, the passenger is separated from their cabin luggage and they follow separate procedures. The cabin luggage gets scanned by a CT machine, which produces images via tomography. These images are analyzed by algorithms and security operators for forbidden objects. If something suspicious was found by either the algorithms or the security operator, then the luggage is rejected and otherwise it is not. Meanwhile, the passenger gets scanned by a body scanner, which detects metallic and non-metallic objects. If the body scanner detects the presence of a suspicious object on the passenger, an extra inspection of the passenger is conducted in the form of a pat-down by a security operator. Once the pat-down

is completed, the passenger continues provided that the object was deemed unsuspicious. The passenger is now available to retrieve their luggage. If the luggage is cleared, then it can be reclaimed by the passenger. However, if the luggage is rejected, then an additional check must be performed by a security operator at the recheck position, before the passenger can pick up their luggage. Once all cabin luggage has been retrieved, the passenger has completed the security checkpoint process. A visualization of the passenger and luggage flows is shown in Figure 1.

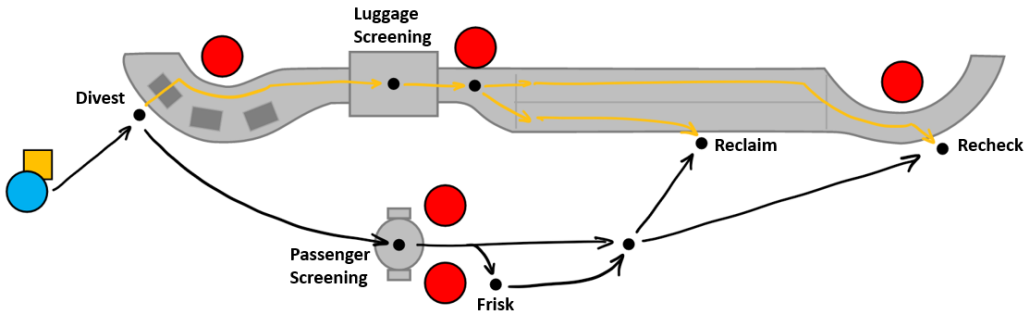


Figure 1: Passenger (blue) and luggage (yellow) flow at a security checkpoint.

Together with security experts from Amsterdam Airport Schiphol, the most important factors affecting the efficiency of a security checkpoint process were determined by observing the day-to-day operations. Efficiency is primarily determined by the throughput. Other aspects affecting the efficiency are the utilisation of operators and frequency of bottlenecks. These factors can be divided into human characteristics (e.g., passenger and security operator) and technical attributes. Each factor and its corresponding value are listed in Table 1. Values were determined by analyzing timestamps of passengers and luggages at Amsterdam Airport Schiphol's security checkpoints. If it was not possible to determine through the airport database, then physical measurements were performed at the security checkpoint and an expert was consulted. Due to confidentiality, logged and measured data could not be published.

Table 1: Collected data on various security checkpoint processes and assets.

Passenger Characteristics

Parameter	**Unit**	**Value**	**Source**
Divest time	$[s/lug]$	$LogN(3.375, 0.669)$	Database
Scan time	$[s/pax]$	$LogN(2.076, 0.885)$	Measurements
Reclaim time	$[s/lug]$	$LogN(3.3, 0.3)$	Measurements + Expert
Image factor	$[lug/pax]$	1.6	Database

Operator Characteristics

Parameter	**Unit**	**Value**	**Source**
Decision time	$[s/lug]$	$LogN(2.450, 0.49)$	Database + Expert
Frisk time	$[s/pax]$	$N(22.5, 18.2)$	Measurements [44]
Recheck time	$[s/lug]$	$LogN(4.325, 0.777)$	Database

Checkpoint Characteristics

Parameter	**Unit**	**Value**	**Source**
Reject rate CT	$[-]$	12.25%	Database
Reject rate SSc	$[-]$	20.96%	Database
Timeout time	$[s]$	30	Database

The normal distribution and lognormal distribution are used to describe the human characteristics in Table 1. Both distributions are common in real-world situations. Furthermore, the lognormal distribution is skewed, which allows it model decision-making processes. Their probability density functions are given in Equation (1) and Equation (2).

$$N(\mu,\sigma): \quad pdf(x,\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{1}$$

$$LogN(\mu,\sigma): \quad pdf(x,\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma x} e^{-\frac{1}{2}\left(\frac{\ln x-\mu}{\sigma}\right)^2} \tag{2}$$

The observations and collected data from daily operations identified which sub-processes and factors need to be considered when developing an agent-based efficiency model of the security checkpoint. The corresponding model is described in Section 3.2.

3.2 Agent-Based Model

We decided to use an agent-based modelling technique for the development of a security checkpoint efficiency model because human behaviour is pivotal to how airport terminal processes function. The proposed agent-based model is an extension of the AATOM simulation model [37] because the original version can not simulate different security checkpoint configurations nor does it contain detailed passenger or operator activities at the checkpoint. Therefore, new environment elements and new checkpoint activities were added into the original model. Consequently, goal and planning modules of agents were also altered in order to execute these new activities. These changes capture the microscopic interactions between humans and include a high level of detail regarding the sub-processes within a security checkpoint. Figure 2 demonstrates the modified architecture of AATOM. Due to the modular architecture, it is possible design and simulate any security checkpoint configuration.

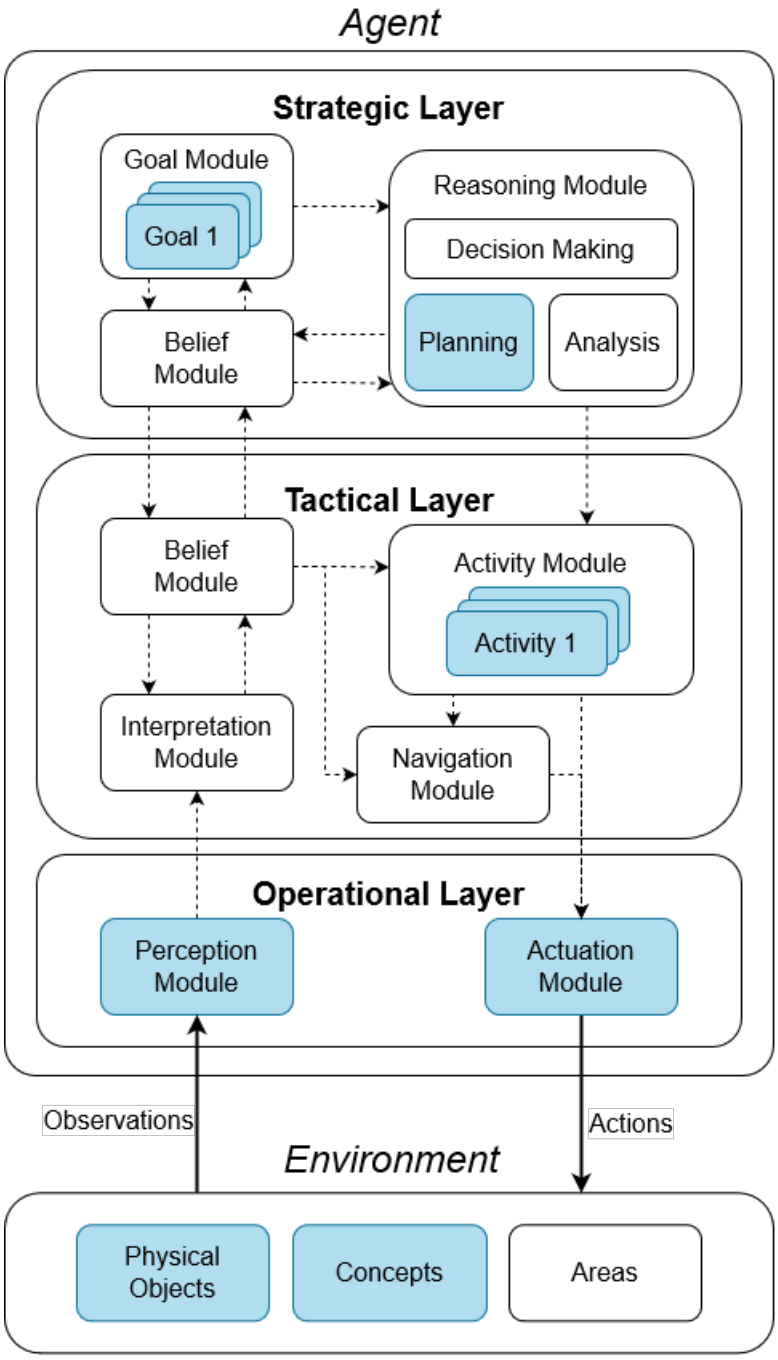


Figure 2: Overview of extended AATOM simulation model, focusing on agent architecture and environment abstraction. The extended modules are highlighted in blue.

The model description is arranged as follows: Section 3.2.1 and Section 3.2.2 describe the environment and agent specifications, respectively. Section 3.2.3 expresses the interaction of an agent with the environment, while Section 3.2.4 outlines the interaction of an agent with other agents. Section 3.2.5 defines the performance indicators used to evaluate the model. Section 3.2.6 determines how many episodes are needed for statistically significant results. Lastly, Section 3.2.7 validates the models performance by comparing it to daily operations of different security configurations at Amsterdam Airport Schiphol.

3.2.1 Environment Specification

The environment is modelled as an abstraction of an airport security checkpoint. It consists of elements that form the static components of the map. Each element is characterized by an environment object type, which can either be a physical object, an area or a concept.

Physical objects represent a list of corner points that agents are not able to enter or pass through. We use four types of physical objects, namely: queues, walls, belts and sensors. Queues and walls separate public from restricted areas. Belts have a start position and end position, along which luggage is transported. Luggage

cannot overtake on the belt and will stop if it reaches the end position or the luggage in front has stopped. Furthermore, belts can have divest or reclaim positions to allow interaction with passengers, such as dropping or picking up luggage. Sensors can observe threat levels of objects when their positions overlap and have a threshold for when to raise an alarm. If the objects threat level is greater than the sensors threshold, then an alarm is raised. Otherwise, no alarm is raised. This threshold represents the reject rate of the security assets (see Table 1).

Areas define a list of corner points and symbolize an airport facility where agents agents can perform certain activities in. The types of areas used in the model are as follows: entrance areas, queuing areas and waiting areas. Entrance areas denote the space where agents are generated on the map. Queuing areas designate a zone where agents line up and their the walking mechanism is turned off. Waiting areas define the space where agents can stop and wait until a condition has been met.

Lastly, physical objects and areas can be combined to create a system. For example, a security lane contains multiple belts, sensors and waiting areas. The environment dynamics establishes how luggage flows through a security lane. Luggage transfers between belts follow a strict order, namely: divest belt → CT belt → decision belt → reclaim belt or reject belt. It is important to note that a luggage can only be transferred once it reaches the end position of the current belt and if the start position of the target belt is free.

3.2.2 Agent Specification

Our agents use the original AATOM agent architecture with slight adjustments to the goal, planning and activity modules (see Figure 2). When an agent is generated at the airport terminal or security checkpoint, it initializes with goals that it wants to accomplish (*Goal Module*). Through reasoning, an agent generates a plan (i.e., a list of activities) to achieve these goals (*Planning Module*). Based on the strategy, an agent will navigate (*Navigation Module*) through the environment and execute activities (*Activity Module*). Agents perform activities by observing (*Perception Module*) and executing actions (*Actuation Module*). It is important to note that agents continuously maintain their beliefs and goals, thereby potentially resulting in new plans. Furthermore, agents are only able to perform one activity at a time. The main sequence of operations followed by an agent is: observation → perception → interpretation → reasoning → activity control → actuation → action. There are two agent types in the simulation: passenger agents and security operator agents.

Passenger Agent

We initialize the passenger agents with the goal to complete the security check. Each passenger will create a plan of activities in order to achieve this goal. Since the security checkpoint has strict procedures, the activities to be completed are the same for all passengers. Activities are performed in chronological order and can start when an initial condition has been satisfied. Activity performance depends on passenger attributes. A list of passenger attributes (P_{Att}) and mandatory airport security terminal activities for passengers (P_{Act}) can be found below.

During the simulation, passengers are spawned continuously in the entrance area such that the banklining of the security checkpoint is always full. This allows us to determine the performance of the security checkpoint under continuous demand. Passengers are destroyed when they exit the security checkpoint.

Passenger Attributes

P_{Att} 01 *CheckedIn*
DESCRIPTION: If checked-in, then true. Otherwise, false.
INITIALIZATION: True.

P_{Att} 02 *Gender*
DESCRIPTION: Male or female.
INITIALIZATION: Either with 50% probability.

P_{Att} 03 *Threat Level*
DESCRIPTION: A value that influences whether or not a passenger will be frisked.
INITIALIZATION: Random sample from uniform distribution ranging between 0 and 1.

P_{Att} 04 *Luggage*
DESCRIPTION: A collection of luggages, each with its own attributes ($L_{Att.}$).
INITIALIZATION: Number of luggages depend on the image factor (see Table 1).

$L_{Att.}$ 01 *Type*
DESCRIPTION: Checked luggage or cabin luggage.
INITIALIZATION: Cabin.

$L_{Att.}$ 02 *Threat Level*
DESCRIPTION: A value that influences whether or not a luggage will be rejected.
INITIALIZATION: Random sample from uniform distribution ranging between 0 and 1.

P_{Att} 05 *Divest Time*
DESCRIPTION: The time it takes to divest.
INITIALIZATION: Random sample from the divest distribution.

P_{Att} 06 *Scan Time*
DESCRIPTION: The time it takes for a passenger to be scanned.
INITIALIZATION: Random sample from the scan distribution.

P_{Att} 07 *Reclaim Time*
DESCRIPTION: The time it takes to reclaim.
INITIALIZATION: Random sample from the reclaim distribution.

Passenger Activities

P_{Act} 01 *Lane Assignment Activity*
Starts when a passenger reaches the end position of the banklining. A passenger observes the occupancy of the security lanes, decides which lane to go to based on the minimum number of passengers, and then goes to the entry position of that security lane.

P_{Act} 02 *Divest Activity*
Starts when a passenger reaches the entry position of the security lane. A passengers waits until a divest position is assigned to them, then goes to the divest position and drops their luggage onto the belt. The total time to drop a luggage is equal to **P_{Att} 05**. Once all luggage has been divested, the passenger continues to the entry position of the security scanner.

P_{Act} 03 *Screen Activity*
Starts when a passenger reaches the entry position of the security scanner. A passenger observes if they can go into the security scanner and then proceeds to go to the scan point if it is unoccupied. When passenger reaches the scan point, an operator performs the passenger screen activity (see **O_{Act} 02**). A passenger completes their activity when they have been scanned and are not suspicious.

P_{Act} 04 *Reclaim Activity*
Starts when a passenger is not suspicious. A passenger goes to the reclaim area and waits for their luggage to be scanned. Next, the passenger observes whether or not their luggage is rejected. If all of their luggage has been rejected, then the activity ends immediately. On the contrary, if a luggage has not been rejected, then the passenger will observe whether a reclaim position is unoccupied. When a reclaim position is unoccupied, they proceed to go to that reclaim position and collect that luggage. If a passenger has another piece of luggage, the process restarts with the passenger observing whether their luggage has been rejected or not. The total time to collect a luggage is equal to **P_{Att} 07**. Once all luggage has been collected, the activity is completed.

P_{Act} 05 *Reject Activity*
Starts when a passenger has rejected luggage. A passenger goes to the recheck area and observes if their luggage has been selected for examination by a security operator. Once selected, the passenger goes to the recheck position and waits for the security operator to perform the luggage recheck activity (see **O_{Act} 03**). When the recheck is finished, the passenger observes if they collected all of their luggage. If all luggage has been collected, the activity is completed. Otherwise, the passenger goes back to the recheck area and the process restarts.

P_{Act} 06 *Exit Activity*
Starts when all luggage has been collected. A passenger goes to the exit position of the security checkpoint. Once the position is reached, the passenger is removed from the simulation.

Operator Agent

We initialize operator agents with a specific security assignment and location. From the security assignment, the operator gets a goal and an activity that they must perform. It is important to note that an operator agent continuously performs the same activity. Activity performance depends on operator attributes. A list of operator attributes (O_{Att}) and potential activities (O_{Act}) can be found below. Operators are destroyed when the simulation ends. A list of operator attributes (O_{Att}) and activities (O_{Act}) can be found below.

Operator Attributes

O_{Att} 01 *Gender*
DESCRIPTION: Male or female.
INITIALIZATION: Depends on the security checkpoint configuration.

O_{Att} 02 *Assignment*
DESCRIPTION: The security task that an operator is assigned to. There are three possible types:

O_{Ass} 01 *CT Check*

O_{Ass} 02 *Passenger Check*

O_{Ass} 03 *Luggage Recheck*
INITIALIZATION: Depends on the security checkpoint configuration.

O_{Att} 03 *Performance Distribution*
DESCRIPTION: The performance distribution of the operator for that assignment.
INITIALIZATION: Depends on the assignment.
O_{Ass} 01 → corresponds to the decision time distribution.
O_{Ass} 02 → corresponds to the frisk time distribution.
O_{Ass} 03 → corresponds to the recheck time distribution.

Operator Activities

O_{Act} 01 *CT Singleplex Activity* ← corresponds to **O_{Ass} 01**
Starts when there is cabin luggage to be checked. An operator selects the first luggage waiting to be checked. Then, they proceed to analyze whether or not the luggage should be rejected. The time to make a decision is determined by taking a random sample from the operator performance distribution. Decisions are based on the information obtained from the CT sensor. If the CT sensor raises an alarm, then the luggage is rejected and otherwise it is cleared. Once a decision has been made, the luggage is able to divert and the activity ends.

O_{Act} 02 *Passenger Screen Activity* ← corresponds to **O_{Ass} 02**
Starts either when there is a passenger to scan or to frisk. If there is a passenger to scan, the operator begins the security scanner. The time of the scan is equal to **P_{Att} 06**. Once scanned, the operator checks whether or not the security scanner raise an alarm. If no alarm was raised, then the passenger is considered as unsuspicious and is cleared. On the contrary, if an alarm was raised, the passenger is considered suspicious and must be frisked. If there is a passenger to be frisked, then the operator commands the passenger to move to the frisk position. It is important to note that an operator can only frisk a passenger if they have the same gender. Once the position has been reached, the passenger is searched for forbidden items. The time of the search is determined by taking a random sample from the operator performance distribution. When the search is done, the passenger is marked as not suspicious and the activity is completed.

O_{Act} 03 *Luggage Recheck Activity* ← corresponds to **O_{Ass} 03**
Starts when there is a luggage on the end position of the reject belt. An operator picks up the luggage and motions for the owner to move to the recheck position. Once the owner has arrived at the position, the operator begins the search. The time of the recheck is determined by taking a random sample from the operator performance distribution. When the search is complete, the luggage is handed over to the passenger and the activity is completed.

3.2.3 Agent Interaction With Environment

Agents can make observations of the environment and execute actions within the environment through the perception and actuation module. There are four types of interactions occurring in the model between agents and the environment: passenger interaction with belts, operator interaction with belts, operator interaction with sensors and operator interaction with concepts.

Passenger Interaction With Belts
Passengers can observe belts that are within their vicinity. Depending on what positions the belt has, a passenger is either able to *drop* or *pick up* their luggage. These interactions occur in **P_{Act} 02** and **P_{Act} 04**.

Operator Interaction With Belts
Operators can observe belts that are part of their activity and add or remove luggages to these belts. Furthermore, an operator can observe which luggage needs a recheck, then perform a recheck and update the record. These interactions occur in **O_{Act} 03**.

Operator Interaction With Sensors
Operators can a observe if a sensor has raised an alarm or not. Subsequently, they process the new information received of the environment and make decisions based on it. In addition, operators can observe and update system records of sensor. For example, a CT sensor keeps track of which luggages have not been checked. An operator can observe which luggage has not been checked, then check the luggage and update the CT sensor record. Similarly, a SSc sensor keeps track of which passengers still need to be scanned. An operator can observe which passenger still needs to be scanned, then scan the passenger and update the SSc sensor record. These interaction occur in **O_{Act} 01** and **O_{Act} 02**.

3.2.4 Agent Interaction with Agents

Agents can communicate with each other through the communication module. In order to communicate, a communication type and message is required. We assume that only operators perform these requests to passengers. Therefore, the only agent to agent interaction in the model is from an operator to a passenger.

Operator Interaction With Passengers
An operator can interact with a passenger by communicating with them. There are two communication types that a passenger can receive from an operator. The first communication type is a *go to* request which requires a destination position as a message. A passenger interprets the message and assigns the destination position as its new navigation goal. The second communication type is a *wait* request which requires a wait time as a message. A passenger interprets the message and proceeds to wait for that duration. There is only one collaboration These interactions occur in $\mathbf{O}_{Act}$ **02** and $\mathbf{O}_{Act}$ **03**.

3.2.5 Performance Indicators

In order to validate the efficiency model of a security checkpoint against real-life operations, the simulated performance of the model must be evaluated. We determine the performance based on the following key performance indicators (KPIs). These indicators are useful for assessing the efficiency of a security checkpoint.

KPI. 01 *Security Checkpoint Throughput* [*pax*]
The total number of passengers that have completed $\mathbf{P}_{Act}$ **06**.

KPI. 02 *Operator Utilisation* [%]
The percentage of total time an operator spent performing their activity versus the total simulation time. This indicator can be separated into each operator assignment.

KPI. 03 *Passenger Time in Area* [*s*]
The total time a passenger has spent in a specific waiting area of the security checkpoint.

KPI. 04 *Percentage of Time in Critical Operations* [%]
The percentage of total time a security sub-process is in critical operations versus the total simulation time. There are four security sub-processes: CT, security scan, reclaim and recheck. With security experts from Amsterdam Airport Schiphol, we define thresholds for each process that represent critical operating conditions. If the number of passengers in an area exceed that threshold, then a process is considered to be in critical operations. For the security scan process, the threshold is when 3 passengers are waiting in the security scan area. For the CT process, the threshold is when 3 passengers are waiting on their luggage in the reclaim area. For the reclaim process, the threshold is when 3 passengers are waiting for a position to free up. For the recheck process, the threshold is when 3 passengers are waiting to get their luggage rechecked.

3.2.6 Calibration

In order to obtain results that are statistically significant, the agent-based model must be run for multiple episodes due to the stochastic nature of the the security checkpoint. The number of episodes required is determined by the coefficient of variation [45].

Coefficient of Variation
The coefficient of variation represents the relative dispersion of a parameter. It is defined as: $CV = \sigma/\mu$ where σ is the standard deviation and μ is the mean of the dataset. After each episode of the simulation, the coefficient of variation for various key performance indicators is determined (see Figure 3). If the coefficient of variation converges, then a sufficient number of episodes have been run. It can be concluded that 50 episodes are sufficient for the simulation.

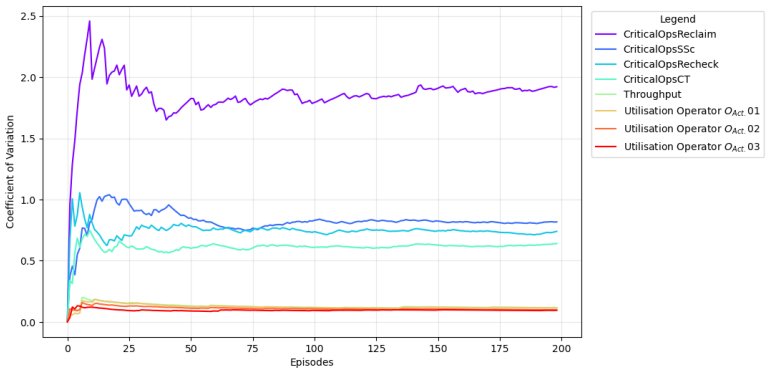
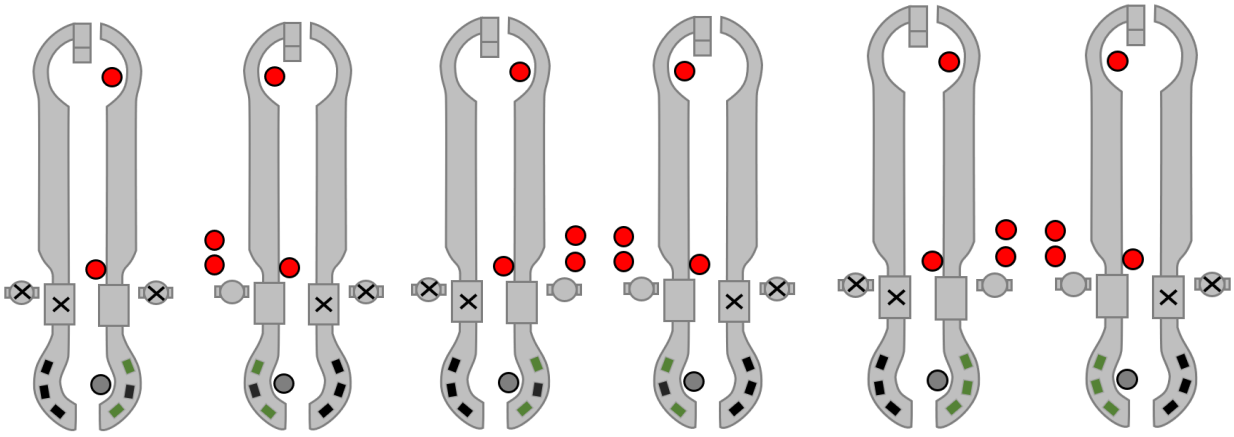


Figure 3: Coefficient of Variation of various KPIs.

3.2.7 Validation

In order to validate the model, several security checkpoint configurations were tested both in the simulator and real-life at Amsterdam Airport Schiphol. Figure 4 shows the different configurations of the security checkpoints that were tested. Figure 4a and Figure 4b represents different configurations tested during the covid pandemic. They satisfy the social distancing measures of 1.5m by using 2 divest positions per lane and no door-scanning [1] at the security scanner. Figure 4c represents the standard security configuration with 3 divest positions per lane and 4 security operators with 2 security scanners per bay using door-scanning.



(a) Covid (1SSc) Configuration (b) Covid Configuration. (c) Standard Configuration

Figure 4: Tested security checkpoint configurations for validation. Configurations vary depending on open divest positions (green), security operators (red) and security scanners.

.

For each configuration, the simulator was run for 100 episodes using the input data of Table 1. Results of the simulations are shown in Table 2. In order to validate these numbers, each configuration was measured for a minimum of 6 hours over the span of multiple days. A measurement team consisted of three people, namely: one person ensuring for continuous passenger demand to the checkpoint and two people for measuring the corresponding throughput. Furthermore, each team was instructed to observe the operations of the checkpoint. The throughput was measured by counting the number of passengers passing through the security scanner. The observations focused on noticing/evaluating which security sub-processes are bottlenecks and how busy security operators were. Results of the measurements are shown in Table 2.

As can be seen from Table 2, there was no significant difference in the throughput of the simulator versus the measurements. Furthermore, the sub-process having the maximum percentage of time in critical operations aligned well to the observed bottleneck in the operations. In the Covid (1SSc) Configuration, it was evident, both through simulation and observation, that the security scan was limiting the throughput of the checkpoint. Similarly, in the Standard Configuration, it was apparent that the CT and recheck were the bottlenecks. By contrast, in the Covid Configuration, there was no observed bottleneck although simulations anticipated the recheck to be in critical operations for 7% of the time. This inconsistency is due to the distribution of critical operations for recheck being positively skewed (i.e. many episodes with low times in critical operations and a

[1] Door-scanning is when one security operator is performing a frisk while another is operating the security scanner.

few with very high times in critical operations). Hence, it was unlikely to see events in real-life where recheck would become a bottleneck. Lastly, security experts from Amsterdam Airport Schiphol were asked to assess the simulators results on the utilisation of operators performing different assignments. They concluded that all utilisations seemed realistic.

Table 2: Results of the simulator (Sim.) and measurements (Msr.) for tested security configurations. All values are averages of the total episodes and total measurements.

KPI	**Unit**	**Configurations**					
		Covid (1SSc)		Covid		Standard	
		Sim.	Msr.	Sim.	Msr.	Sim.	Msr.
Throughput	$[pax/min/lane]$	1.36	1.35	1.77	1.80	2.44	2.4
Throughput	$[pax/min/ops]$	0.272	0.27	0.354	0.36	0.488	0.48
Utilisation $\mathbf{O}_{Ass}$ **01**	[%]	45.1	-	57.6	-	80.7	-
Utilisation $\mathbf{O}_{Ass}$ **02**	[%]	46.6	-	30.6	-	43.3	-
Utilisation $\mathbf{O}_{Ass}$ **03**	[%]	71.5	-	70.2	-	85.1	-
Critical Ops CT	[%]	0.5	-	1.5	-	25.2	Btl.
Critical Ops SSc	[%]	80.4	Btl.	2.8	-	6.9	-
Critical Ops Reclaim	[%]	0	-	0.7	-	10.0	-
Critical Ops Recheck	[%]	2.2	-	7.4	-	27.8	Btl.

It is important to note that in the Covid (1SSc) Configuration, operators assigned to $\mathbf{O}_{Ass}$ **02** were busy 46.6% of the time but the security scan was in critical operations for 80.4% of the time. This is because for each security scan there are two operators, where one operator has high utilisation (85.0%) and the other operator has a low utilisation (8.3%). Differences in utilisation occur because operating the security scanner only required one operator and frisks do not occur often.

4 Learning Agent

In this section, a learning agent is introduced into the agent-based model (see Section 3.2) for allocating reserve operators to improve security checkpoint efficiencies. We used a learning agent because they are able to converge to an optimal policy assuming that all states have sufficiently been explored. Section 4.1 describes the learning algorithm of the agent. Section 4.2 defines the attributes of the learning agent. Section 4.3 outlines the activities that a learning agent can perform. Section 4.4 explains how the learning behaviour between agents are compared. Lastly, Section 4.5 presents the performance indicators used to analyze the policy of an agent and the operational efficiency of the security checkpoint.

4.1 Basic Q-Learning Algorithm

We decided to implement the Q-Learning (QL) algorithm [46] because it is a model-free algorithm (i.e. does not require an internal model) and it converges to the optimal policy provided that state-action pairs are continuously visited and updated [47]. Before a QL agent can be deployed in an environment, its states, actions and rewards must be established. Once deployed, the interaction between the agent and environment occur at discrete time steps, $t = 0, 1, 2, 3, ..., n$. During each time step t, the agent observes a representation of the environment's states $S_t \in S$, and takes an action $A_t \in A(S_t)$. One time step later, the agent is situated in a new state S_{t+1} and receives a reward $R_{t+1} \in R$ to evaluate its transition. Since this process can continue indefinitely, an agents goal is to select actions such that sum of its discounted rewards it will receive in the future, known as expected discounted return, is maximized. The expected discounted return (G_t) is shown in Equation (3), where γ^k is the discount factor and represents the relative importance of the current reward versus rewards in the distant future ($0 \leq \gamma \leq 1$).

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{3}$$

A QL agent manages to achieve their goal by estimating and updating an action-value function $Q(S, A)$, which evaluates the added benefit of performing a given action in a given state. This function is estimated using the action-value of the current state, the observed reward and the best action-value of the successor state. The update of action-value function $Q(S, A)$ occurs every discrete time step according to Equation (4). This learned function directly approximates the optimal action-value function, regardless of the policy an agent follows.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \quad (4)$$

A policy $\pi(s)$ is a probabilistic mapping of performing an action in a given state for all possible state-action pairs. Hence, it governs which action an agent will take for every possible state. We use an *ϵ-greedy* policy for our QL agent such that it does not stuck get stuck in a local optimum. The policy does this by ensuring that there is a balance between exploration and exploitation for when an agent selects an action. Exploration is when an agent performs a random action, thereby encouraging it to experience new states, which might lead to beneficial rewards in the future. However, exploitation is when an agent performs the action that maximizes its value function, thereby allowing it to re-experience previous states considered to be good, which leads to convergence. The *ϵ-greedy* policy has a probability of ϵ of exploration and a probability of 1-ϵ of exploitation.

Algorithm 1 Q-Learning with *ϵ-greedy*

Input $\alpha, \epsilon, d_\epsilon, \gamma$
$Sim \leftarrow$ Simulator() ▷ Contains observation space S, set of actions A and reward function R
$Q(s, a) \leftarrow$ ValueFunction(S, A)

for *episode* in *episodes* **do**
  $Sim \leftarrow Sim$.reset()
  Sim.setSeed(*episode*) ▷ Ensures that each episode has a unique seed.
  while Sim not terminated **do**
    $S_t \leftarrow$ observe(Sim)
    $A_t \leftarrow$ selectAction(S_t) ▷ Selection is performed based on *ϵ-greedy* policy.
    performAction(A_t)
    Sim.update() ▷ The timestep t increased by 1.
    $S_{t+1} \leftarrow$ observe(Sim)
    $R_{t+1} \leftarrow$ observeReward()
    $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$
  end while
end for

4.2 Attributes

Learning agents are initialized with parameters required for Q-Learning (mentioned in Section 4.1) and a specific assignment, which determines the activity that an agent will perform. A list of learning agent attributes ($\mathbf{Q}_{Att}$) are listed below.

Q-Agent Attributes

$\mathbf{Q}_{Att}$ **01** *Learning Rate* (α)
DESCRIPTION: It determines how much an agent learns from new information gained.
INITIALIZATION: Depends on the experiment.

$\mathbf{Q}_{Att}$ **02** *Exploration Probability* (ϵ)
DESCRIPTION: It introduces randomness into the action selection of the QL algorithm through the *ϵ-greedy* policy.
INITIALIZATION: Depends on the experiment.

$\mathbf{Q}_{Att}$ **03** *Epsilon Decay* (d_ϵ)
DESCRIPTION: It determines how much the exploration factor decreases after each episode.
INITIALIZATION: Depends on the experiment.

$\mathbf{Q}_{Att}$ **04** *Discount factor* (γ)
DESCRIPTION: It influences the importance of the future rewards.
INITIALIZATION: Depends on the experiment.

$\mathbf{Q}_{Att}$ **05** *Value Function* ($Q(s, a)$)
DESCRIPTION: It represents the knowledge an agent has learned by indicating the benefit of performing an action in a given state.
INITIALIZATION: We use a Q-Matrix initialized with 0 for all actions possible in each state. It was decided to use a Q-Matrix because the state-space is small and allows us to better analyze the learning behaviour.

$\mathbf{Q}_{Att}$ **06** *Assignment*
DESCRIPTION: The learning task that an agent is assigned to. There are two possible types:
  $\mathbf{Q}_{Ass}$ **01** *Recheck Operator Allocation*

$\mathbf{Q}_{Ass}$ 02 *CT Operator Allocation*
INITIALIZATION: Depends on the experiment.

It is important to note that the states, actions and rewards an agent experiences depend on the activity the agent performs in Section 4.3.

4.3 Activities

From the assignment, the learning agent gets an activity that they must perform. The activity defines the states, the actions possible in each state and the reward function an agent can experience. Learning agents perform the same activity each timestep until the simulation is terminated. It is important to note that only two learning activities (1. recheck operator allocation and 2. CT operator allocation) were explored because those were the main bottlenecks in the standard lane configuration (see Table 2) and have the most potential of increasing the efficiency of a checkpoint. The description of the learning agent activities ($\mathbf{Q}_{Act}$) can be found below.

Q-Agent Activities

$\mathbf{Q}_{Act}$ 01 *Allocating Recheck Operators Activity* ← corresponds to **$\mathbf{Q}_{Ass}$ 01**
A q-agent learns when to allocate security operators to the luggage recheck assignment for a single security checkpoint lane. Starts each timestep of the simulation. An agent observes its old state, selects an action, performs that action, observes the new state, receives a reward and update its value function. Then, the activity is completed.

STATES:
In collaboration with Amsterdam Airport Schiphol, three variables were selected that could describe the operations of the recheck process. These variables could be easily monitored either through their in-house systems or by a security team leader[2]. They are as follows:

S. 01 Number of operators assigned to luggage recheck that are available.
S. 02 Number of operators assigned to luggage recheck that are busy.
S. 03 Number of luggages waiting on the recheck belt.

It is important to note that the total number of operators assigned to a luggage recheck can not exceed 3 operators due to spacial constraints of the security lane. Similarly, the number of luggages on the recheck belt can not exceed 5.

ACTIONS:
An agent can control the performance of the recheck process by adjusting the number of operators working at a security lane. Thus, there are three actions considered:

A. 01 Remain the same.
A. 02 Add an operator assigned to luggage recheck.
A. 03 Remove an operator assigned to luggage recheck.

If there are already 3 operators assigned to luggage recheck, then adding another is illegal and a different action must be selected. Similarly, if there are zero operators assigned to luggage recheck, then removing an operator is not possible and a different action must be selected. Lastly, it is not possible to remove an operator whose activity is in progress.

REWARDS:
The purpose of allocating operators to the luggage recheck assignment was to improve the operational efficiency of the security lane. In order to accomplish this, rewards were given to the learning agent for luggage and operators. The formulation for each reward can be found below.

R. 01 Luggage
If there are more pieces of luggage on the recheck belt (n_L) than there are available recheck operators ($n_{OpsAvailable}$), then a negative reward ($\text{-}r_L$) is received for each additional luggage. Otherwise, the reward is 0.

$$R_L = \begin{cases} \text{-}r_L \cdot (n_L - n_{OpsAvailable}), & \text{if } n_L - n_{OpsAvailable} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

R. 02 Operator
If there are more recheck operators available ($n_{OpsAvailable}$) than there are luggages on the recheck belt (n_L), then a negative reward ($-r_{Ops}$) is received for each additional operator.

[2] A team leader is responsible for the security operator assignments and functioning of a checkpoint.

This represents the operating cost of an operator. Otherwise, a positive reward (r_{Ops}) is received for each recheck operators in progress ($n_{OpsInProgress}$), which represents the throughput.

$$R_{Ops} = \begin{cases} \text{-}r_{Ops} \cdot (n_{OpsAvailable} - n_L), & \text{if } n_{OpsAvailable} - n_L > 0 \\ r_{Ops} \cdot n_{OpsInProgress}, & \text{otherwise} \end{cases} \tag{6}$$

The total reward at each timestep is the sum of the luggage and operator reward $R_{total} = R_L + R_{Ops}$.

Q_{Act} 02 *Allocating CT Operators Activity* ← corresponds to **Q_{Ass} 02**

A q-agent learns when to allocate security operators to the CT check assignment for a single security checkpoint lane. Starts each timestep of the simulation. An agent observes its old state, selects an action, performs that action, observes the new state, receives a reward and update its value function. The activity is completed in the same timestep.

States:

In collaboration with Amsterdam Airport Schiphol, three variables were selected that could describe the operations of the CT process. These variables should be easily monitored either through their in-house systems or by a security team leader. They are as follows:

S. 01 Number of operators assigned to luggage check that are available.
S. 02 Number of operators assigned to luggage check that are busy.
S. 03 Number of luggages that have not been checked.

It is important to note that the total number of operators assigned to a CT check can not exceed 3 operators due to spacial constraints of the security lane. Similarly, the number of luggages that have not been checked can not exceed 5.

Actions:

An agent can control the performance of the CT process by adjusting the number of operators working at a security lane. Thus, there are three actions considered:

A. 01 Remain the same.
A. 02 Add an operator assigned to CT check.
A. 03 Remove an operator assigned to CT check.

If there are already 3 operators assigned to CT check, then adding another is illegal and a different action must be selected. Similarly, if there are zero operators assigned to CT check, then removing an operator is not possible and a different action must be selected. Lastly, it not possible to remove an operator whose activity is in progress.

Rewards:

The purpose of allocating operators to the CT check assignment was to improve the operational efficiency of the security lane. In order to accomplish this, rewards were given to the learning agent for luggage and operators. The formulation for each reward can be found below.

R. 01 Luggage

If there are more luggages waiting to be checked (n_L) than there are available CT operators ($n_{OpsAvailable}$), then a negative reward ($-r_L$) is received for each additional luggage. Otherwise, the reward is 0.

$$R_L = \begin{cases} \text{-}r_L \cdot (n_L - n_{OpsAvailable}), & \text{if } n_L - n_{OpsAvailable} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

R. 02 Operator

If there are more CT operators available ($n_{OpsAvailable}$) than there are luggages waiting to be checked (n_L), then a negative reward ($-r_{Ops}$) is received for each additional operator. This represents the operating cost of an operator. Otherwise, a positive reward (r_{Ops}) is received for each CT operator in progress ($n_{OpsInProgress}$), which symbolizes the throughput.

$$R_{Ops} = \begin{cases} \text{-}r_{Ops} \cdot (n_{OpsAvailable} - n_L), & \text{if } n_{OpsAvailable} - n_L > 0 \\ r_{Ops} \cdot n_{OpsInProgress}, & \text{otherwise} \end{cases} \tag{8}$$

The total reward at each timestep is the sum of the luggage and operator reward $R_{total} = R_L + R_{Ops}$.

4.4 Learning Behaviour Analysis

In order to analyze the learning behaviour of an agent, we constructed a network per episode based on the state transitions ($S_t \rightarrow S_{t+1}$) because it illustrates the actions an agent takes and the system mechanics. Each network is directed where the nodes correspond to all possible states an agent can be in and the weight of each link between node i and j equals $w^*_{ij} = n_{ij}$ the total number of transitions n_{ij} from i to j within an episode.

To compare different learning agents, we computed the similarity of their state transition networks per episode. Since the state transition networks have the same node set, it is possible to use known-node correspondence techniques when comparing networks. Among these techniques, the DeltaCon method has emerged as a powerful tool that satisfies axioms for network comparison [48]. As such, we decided to use the DeltaCon method to compare the learned behaviour per episode between agents.

4.4.1 DeltaCon Method

Assume that two graphs G_1 and G_2 are being compared. The first step was to compute the pairwise node affinities of each graph. Results are stored in an $n \times n$ matrix ($\mathbf{S}$), where each entry (s_{ij}) indicates the influence of node i on node j. We used belief propagation to compute node affinities (see Equation (9)) because it is based on maximum likelihood estimation on marginals and it takes into account k-step away neighbours with decreasing weight on the influence.

$$\mathbf{S} = [s_{ij}] = \left[\mathbf{I} + \omega^2\mathbf{D} - \omega\mathbf{A}\right]^{-1} \tag{9}$$

Next, the distance between each respective node from the pairwise node affinity scores S_1 and S_2 was computed. These distances indicate the difference in influence node i has on node j between the two scores. We used a rooted euclidean distance because it enables detection of small changes in graphs by boosting the node affinities. The distance matrix ($\mathbf{D}$) is calculated according Equation (10).

$$\mathbf{D} = [d_{ij}] = \left[\left(\sqrt{s_{1,ij}} - \sqrt{s_{2,ij}}\right)^2\right] \tag{10}$$

Finally, the similarity between the graphs can be computed from the distance matrix according to Equation (11). This conversion bounds the result to the interval between $[0, 1]$, where 1 means the networks are identical and 0 means the networks are completely different.

$$similarity = \frac{1}{1 + \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}}} \tag{11}$$

4.5 Performance Analysis

Since learning agents were introduced to allocate reserve operators for improving the security checkpoint operational efficiency, it is important to analyze their final policy and its effect on the operational efficiency once learning has finished. Therefore, we run an additional 200 test episodes once learning has completed where the learning agent is able interact with the environment but its value function $Q(s, a)$ does not update anymore.

An agent's final policy is analyzed by determining the action with the maximum q-value for each state, where the value function $Q(s, a)$ corresponds to the one used for the test episodes. Operational efficiency is evaluated using the previously defined performance indicators (see Section 3.2.5) with one new indicator on the test episodes.

KPI. 05 *Number Of Operators* [−]
The average number of operators during the total simulation. This indicator can be separated into each operator assignment.

5 Experiments and Results

This section presents the experiments and corresponding results. An overview of each experiment can be found below. For each experiment, a set of hypothesis is established, a simulation setup is outlined and then the results are evaluated.

Table 3: Overview of the experiments

Experiment	Topic	Discussed in
1	The sensitivity of the learning agent's behaviour when varying the learning rate	Section 5.1
2	The sensitivity of the learning agent's behaviour when varying the exploration probability	Section 5.2
3	The effect of the reward function on the learning agent's performance.	Section 5.3

5.1 Experiment 1: Sensitivity on the Learning Agent's Behaviour when Varying the Learning Rate

The rate at which an agent learns from new information is an important aspect of Q-Learning. Therefore, its sensitivity is analyzed in this section. This experiment focuses on varying the learning rate and analyzing how the learning behaviour of an agent changes. The goal of this experiment is to determine an optimal value for the learning rate.

Hypotheses

H. 01 An increase in learning rate leads to an increase in slope of the total reward received during the exploration phase.

H. 02 A decrease in learning rate leads to a higher reward when an agent's policy has converged.

Simulation Setup

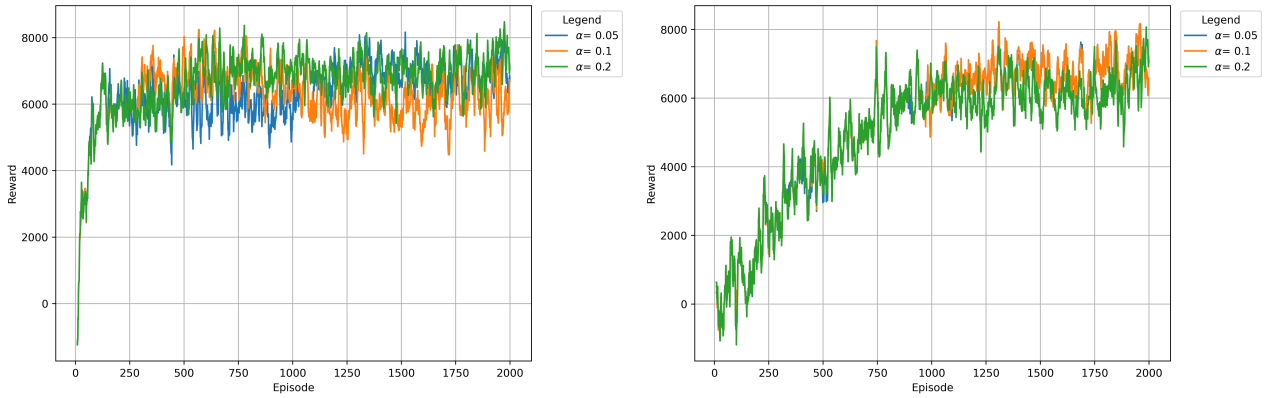
Table 4 presents the parameters chosen for experiment 1. It was decided to test how both learning activities ($Q_{Act.}$) respond to a change in learning rate (α) under two different spans of exploration phase. The span of the exploration phase ($P_{Explore}$) encompasses the episodes where the learning agent is still performing random actions. It was controlled through the exploration decay (d_ϵ) and can be determined by dividing the exploration probability by the decay (ϵ/d_ϵ). By contrast, the exploitation phase ($P_{Exploit}$) encompasses all episodes after the exploration phase. Control variables are the individual rewards (r_L and r_{Ops}), the exploration probability (ϵ) and the discount factor (γ).

Table 4: Input parameters for experiment 1.

Parameter	Value
$Q_{Act.}$	[$\mathbf{Q}_{Act}$ **01**, $\mathbf{Q}_{Act}$ **02**]
r_L	-1
r_{Ops}	10
α	[0.05, 0.1, 0.2]
ϵ	0.9
d_ϵ	[0.01, 0.001]
γ	0.1

Results

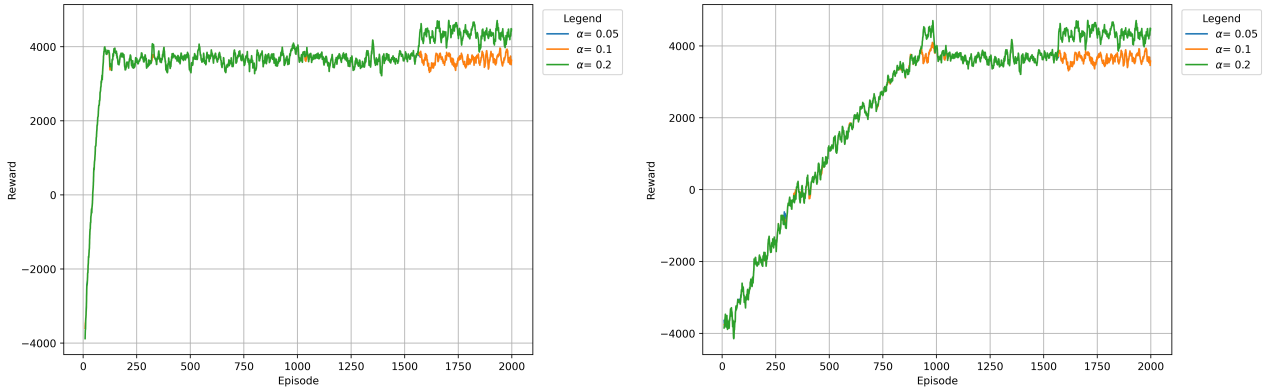
An increase in learning rate has no effect on the slope of the total reward received during the exploration phase for both learning activities. Figure 5 and Figure 6 demonstrate that the total reward received for different simulation is equivalent during the exploration phase. This occurs because a learning agent is experiencing the exact same state transition networks for the majority of the exploration phase, as indicated by Figure 7 and Figure 8, thus receiving the same rewards. The identical state transition networks during the exploration phase is somewhat surprising since varying the learning rate changes how much an agent learns from new information, thereby potentially changing an agent's policy. A possible explanation for the identical networks is the inherent dynamics of the simulation. Although the simulation is stochastic, there is a greater probability of certain states occurring than others. This has led to a subset of states that are visited more frequently for both learning activities. For example, in $\mathbf{Q}_{Act}$ **01** states where the number of luggages waiting on the recheck belt is low (i.e. $\mathbf{Q}_{Act}$ **01 S. 03** < 1) are visited more often. Similarly, in $\mathbf{Q}_{Act}$ **02**, states where the number of luggages waiting to be checked are low (i.e. $\mathbf{Q}_{Act}$ **02 S. 03** < 2) are visited more often. The frequent visits to each of these states drives the agent to learn the same policy. Since few simulations occur where other states are experienced, the majority of the simulations are identical due to agents having the same policy for frequently visited states.



(a) $d_\epsilon = 0.01$
$P_{Explore} = [0, 90)$ and $P_{Exploit} = [90, 2000)$

(b) $d_\epsilon = 0.001$
$P_{Explore} = [0, 90)$ and $P_{Exploit} = [90, 2000)$

Figure 5: Total reward received per episode from the allocating recheck operator activity ($\mathbf{Q}_{Act}$ **01**) with varying α and constant $r_L = -1$, $r_{Ops} = 10$, $\epsilon = 0.9$, $\gamma = 0.1$.



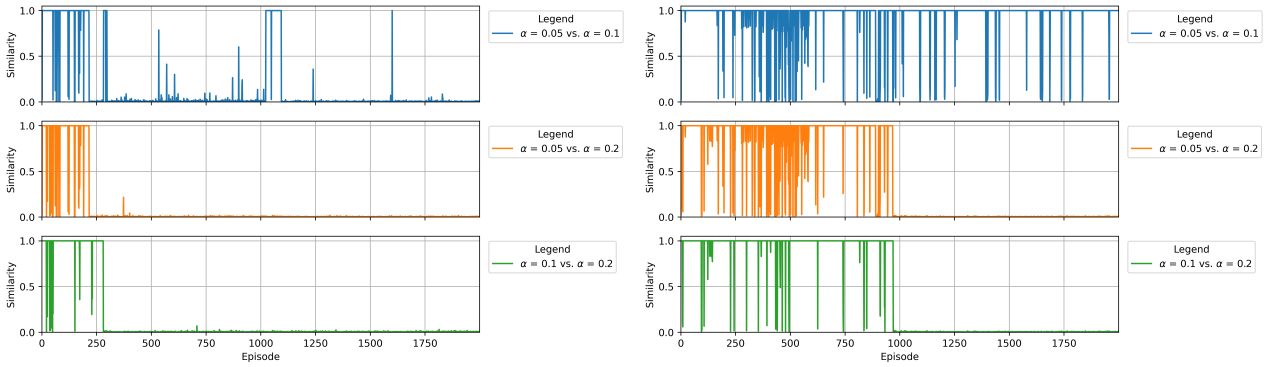
(a) $d_\epsilon = 0.01$
$P_{Explore} = [0, 90)$ and $P_{Exploit} = [90, 2000)$

(b) $d_\epsilon = 0.001$
$P_{Explore} = [0, 900)$ and $P_{Exploit} = [900, 2000)$

Figure 6: Total reward received per episode from the allocating CT operator activity ($\mathbf{Q}_{Act}$ **02**) with varying α and constant $r_L = -1$, $r_{Ops} = 10$, $\epsilon = 0.9$, $\gamma = 0.1$.

A decrease in learning rate does not lead to a higher reward when an agent's policy has converged for both learning activities. Figure 5 illustrates that the total reward received upon convergence changes for various learning rates in learning activity $\mathbf{Q}_{Act}$ **01**. However, the ranking of best to worst changes between the different spans of the exploration phase (d_ϵ). Figure 5a, where $d_\epsilon = 0.01$, shows that increasing learning rate leads to a higher reward once an agent's policy has converged. By contrast, Figure 5b, where $d_\epsilon = 0.001$, shows that decreasing the learning rate leads to a higher reward once an agent's policy has converged. This discrepancy may be explained by the fact that agents with lower spans of exploration are less likely to try different actions for a specific state. Therefore, an agent experiences less state-action pairs. In order to prevent an agent from overfitting on these limited state-action pairs, a higher learning rate can be used which regularizes the training. By contrast, agents with high spans of exploration phase experience a plethora of states-action pairs. Hence, a lower learning rate is better because the smaller step size offers a more exact solution.

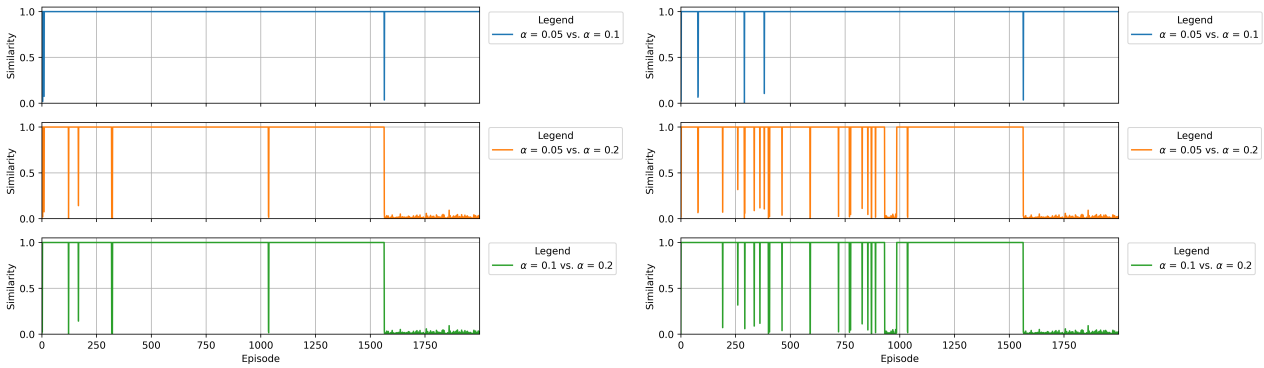
Despite the trend shown in learning activity $\mathbf{Q}_{Act}$ **01**, the total reward received upon convergence increases for increasing learning rates with respect to both spans of the exploration phase in learning activity $\mathbf{Q}_{Act}$ **02**, as shown in Figure 6. This outcome is contrary to earlier findings which have suggested that total reward received upon convergence increases with decreasing the learning rate for higher spans of exploration phases. A possible explanation for this might be that lower learning rate has caused to agent to overfit in $\mathbf{Q}_{Act}$ **02**. This is further supported by Figure 8, which shows that the state-transition networks are identical until episode 1564, where the higher learning rate ($\alpha = 0.2$) manages to escape a local optimum.



(a) $d_\epsilon = 0.01$
Compares the simulations in Figure 5a.

(b) $d_\epsilon = 0.001$
Compares the simulations in Figure 5b.

Figure 7: Similarity in the state transition network per episode between different simulations. Both sub-figures compare simulations from the allocating recheck operator activity ($\mathbf{Q}_{Act}$ **01**) with varying α and constant $r_L = -1$, $r_{Ops} = 10$, $\epsilon = 0.9$, $\gamma = 0.1$.



(a) $d_\epsilon = 0.01$
Compares the simulations in Figure 6a.

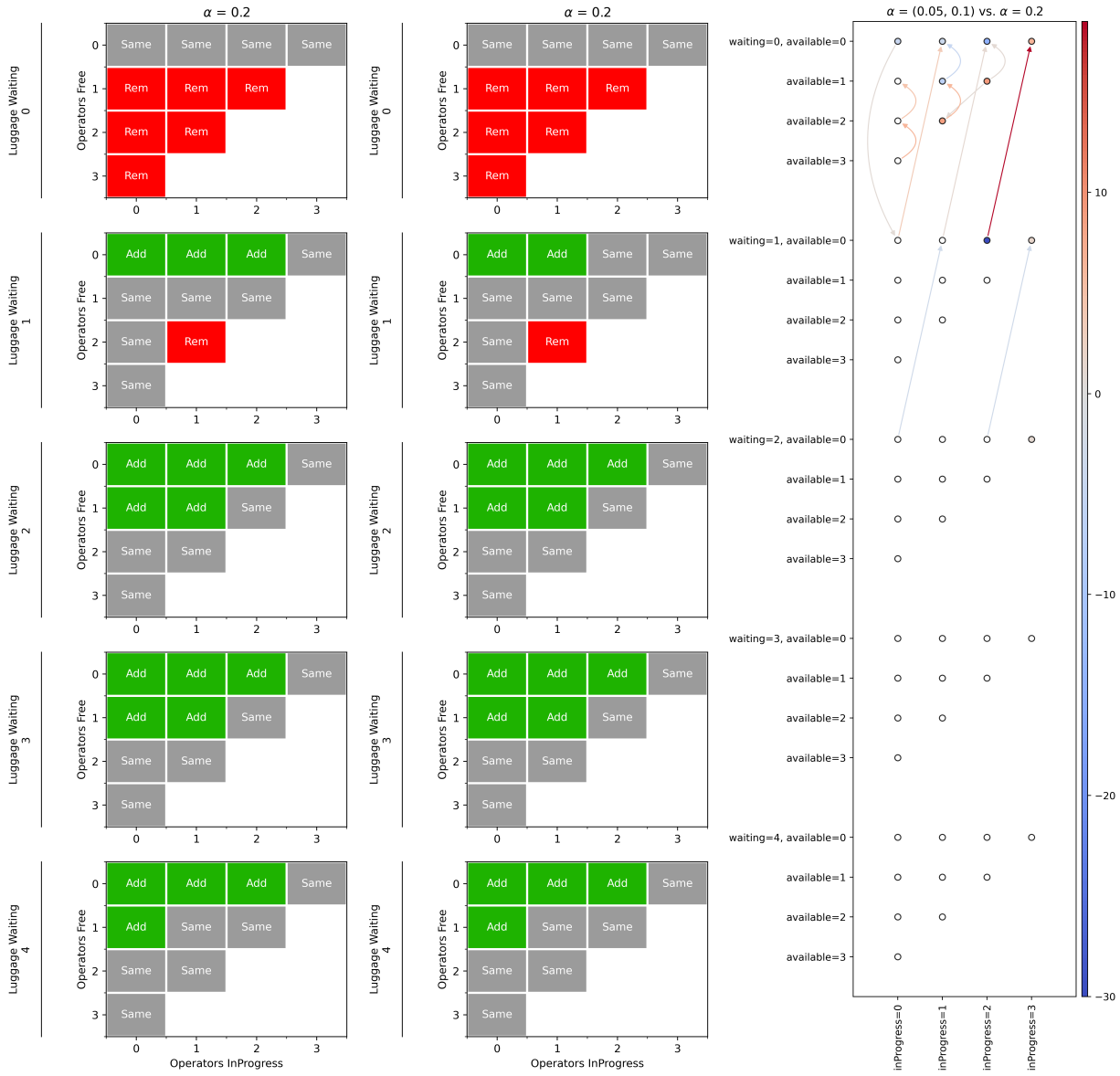
(b) $d_\epsilon = 0.001$
Compares the simulations in Figure 6b.

Figure 8: Similarity in the state transition network per episode between different simulations. Both sub-figures compare the allocating CT operator activity ($\mathbf{Q}_{Act}$ **02**) with varying α and constant $r_L = -1$, $r_{Ops} = 10$, $\epsilon = 0.9$, $\gamma = 0.1$.

Figure 9 shows that the agent assigned to learning activity $\mathbf{Q}_{Act}$ **02** with a learning rate of $\alpha = 0.2$ changes its policy at episode 1564. Specifically, when an agent is in state $(0, 2, 1)$[3], the best action changes from *adding an operator* to *remaining the same*. While the exact reason behind the policy change stems from mathematical context of the update function (see Equation (4)), the difference in state transition networks can help explain the intention behind the change. Figure 9c demonstrates the impact of the policy change on the state transition network, namely: the agent with $\alpha = 0.2$ increases its number of visits to states with a higher luggage waiting value than agents with $\alpha = 0.05, 0.1$. By contrast, agents with $\alpha = 0.05, 0.1$ increase their number of visits to states with a higher number of operators available than agents with $\alpha = 0.2$. In general, states with higher luggage waiting value are punished less than states with higher number of operators available. Therefore, the agent with $\alpha = 0.2$ manages to achieve a higher reward than agents with $\alpha = 0.05, 0.1$, as shown in Figure 6. A possible explanation as to why other agents did not manage to learn this new decision is that it comes with an immediate punishment. By choosing to *remain the same* in state $(0, 2, 1)$, an immediate punishment is received for the luggage waiting, whereas by *adding an operator* no punishments are received since the new operator can check the luggage.

In general, the policy learned by all agents for either activity can be summarized as follows: add an operator if the number of luggages waiting is greater than the number of operators available; remove an operator if the number of operators available is greater than the number of luggages waiting; remain the same if the number of operators available is equal to the number of luggages waiting. Furthermore, states that are rarely visited usually prefer to remain the same since that is the default option. Therefore, the change in policy of learning agent $\mathbf{Q}_{Act}$ **02** $\alpha = 0.2$ at episode 1564 is significant because the number of operators and luggages waiting are valued differently.

[3] A state (x, y, z) corresponds to x operators available, y operators in progress and z luggages waiting.



(a) Best action of value function ($Q(s,a)$) for simulation $\alpha = 0.2$ at episode 1563.

(b) Best action of value function ($Q(s,a)$) for simulation $\alpha = 0.2$ at episode 1564.

(c) Average difference in the state transition networks between simulation $\alpha = 0.05, 0.1$ and simulation $\alpha = 0.2$ for episodes 1564 to 1574.

Figure 9: Explanation of the similarity transition from identical to dissimilar at episode 1564 for Figure 8b, which compares the allocating CT operator activity ($\mathbf{Q}_{Act}$ **02**) with varying α and constant $r_L = -1$, $r_{Ops} = 10$, $\epsilon = 0.9$, d_ϵ=0.001, $\gamma = 0.1$.

5.2 Experiment 2: Sensitivity of Varying Exploration Probability on Learning Agent's Behaviour.

The probability of exploration an agent has when interacting with the environment is crucial to converging to an optimal policy. Therefore, its sensitivity is investigated in this section. This experiment focuses on varying the exploration probability and analyzing how the learning behaviour of an agent changes. The goal of this experiment is to determine an optimal value for the exploration probability.

Hypotheses

H. 01 An increase in exploration probability leads to a higher reward when an agent's policy has converged.

Simulation Setup

Table 5 presents the parameters chosen for experiment 2. It was decided to test how both learning activities ($Q_{Act.}$) respond to a change in exploration probability (ϵ) under two different spans of exploration phase. Again,

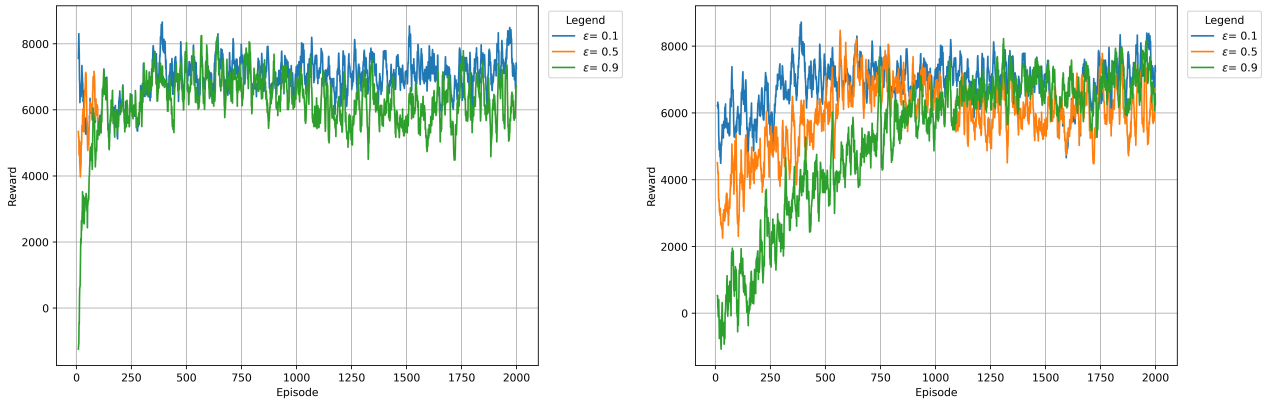
the span of the exploration phase was controlled through the exploration decay (d_ϵ). Control variables are the individual rewards (r_L and r_{Ops}), the learning rate (α) and the discount factor (γ).

Table 5: Input parameters for experiment 2.

Parameter	Value
$Q_{Act.}$	[$\mathbf{Q}_{Act}$ **01**, $\mathbf{Q}_{Act}$ **02**]
r_L	-1
r_{Ops}	10
α	0.1
ϵ	[0.1, 0.5, 0.9]
d_ϵ	[0.01, 0.001]
γ	0.1

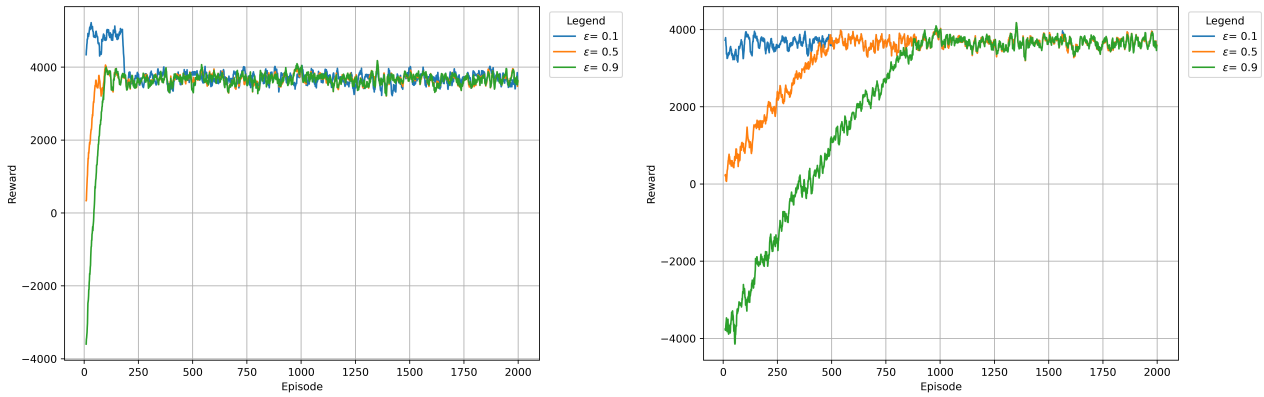
Results

An increase in exploration probability does not result in a higher reward when an agent's policy has converged. Figure 10 illustrates that the total reward received upon convergence increases for decreasing exploration probability with respect to both spans of the exploration phase in learning activity $\mathbf{Q}_{Act}$ **01**. By contrast, Figure 11 demonstrates that the total reward received upon convergence remains the same for varying exploration probability with respect to both spans of the exploration phase in learning activity $\mathbf{Q}_{Act}$ **02**. A possible explanation for these contradictory results is that the agent gets stuck in different local optimums due to the exploration probability. Typically, higher exploration probability helps an agent escape a local optimum. However, in learning activity $\mathbf{Q}_{Act}$ **01**, a higher exploration is detrimental because the agent is not able to put together a good sequence of actions. An agent takes an action every timestep, yet the operator activity $\mathbf{O}_{Act}$ **03** that is being allocated takes on average 102 seconds to complete. Therefore, when a new luggage arrives on the recheck belt, it is inevitable that the agent will add a new operator before the existing operator finishes their activity. This reduces the punishment for waiting luggage and increases the reward for having an operator in progress. Furthermore, the cost of operators also increase, especially when an operator finishes their activity (i.e. not in progress anymore). By contrast, in learning activity $\mathbf{Q}_{Act}$ **02**, the operator activity $\mathbf{O}_{Act}$ **01** that is being allocated takes on average 13 seconds to complete. Therefore, when a new luggage arrives and is waiting to be checked, it is less likely that an agent will add a new operator before the existing operator finishes their activity. Hence, an increase in exploration probability does not result to a decrease in total reward received.



(a) $d_\epsilon = 0.01$ (b) $d_\epsilon = 0.001$

Figure 10: Total reward received per episode from the allocating recheck operator activity ($\mathbf{Q}_{Act}$ **01**) with varying ϵ and constant $r_L = -1$, $r_{Ops} = 10$, $\alpha = 0.1$, $\gamma = 0.1$.



(a) $d_\epsilon = 0.01$ (b) $d_\epsilon = 0.001$

Figure 11: Total reward received per episode from the allocating CT operator activity ($\mathbf{Q}_{Act}$ **02**) with varying ϵ and constant $r_L = -1$, $r_{Ops} = 10$, $\alpha = 0.1$, $\gamma = 0.1$.

5.3 Experiment 3: Effect of the Reward Function on the Learning Agent's Performance

An agent's policy impacts the operational efficiency of the security checkpoint. By changing inputs of the reward function, an agent learns to prioritize other factors which can lead to a different converged policy, thereby affecting the operation of the security checkpoint. Therefore, the effect of the reward function is examined in this section. This experiment focuses on varying the individual rewards (r_L and r_{Ops}), thereby affecting the converged policy, and analyzing how the performance of the security checkpoint changes. The goal of this experiment is to determine which reward structures lead to beneficial policies at the security checkpoint.

Hypotheses

H. 01 An increase in the reward ratio r_L/r_{Ops} leads to an increase in average number of operator over the duration of the simulation.

H. 02 An increase in the reward ratio r_L/r_{Ops} leads to an increase in throughput over the duration of the simulation.

Simulation Setup

Table 6 presents the parameters chosen for experiment 3. It was decided to test how the performance of the security checkpoint changes under both learning activities ($Q_{Act.}$) and varying individual rewards (r_L and r_{Ops}). Control variables are the learning rate (α), the exploration probability along with its decay (ϵ and d_ϵ) and the discount factor (γ).

Table 6: Input parameters for experiment 3.

Parameter	Value
$Q_{Act.}$	[$\mathbf{Q}_{Act}$ **01**, $\mathbf{Q}_{Act}$ **02**]
r_L	[-1,-2,-3,-5,-10]
r_{Ops}	[2,5,10]
α	0.1
ϵ	0.9
d_ϵ	0.001
γ	0.1

Results

The implementation of the learning agent increases the throughput of the security checkpoint and decreases the percentage of time spent in critical operations for the corresponding subprocess, as shown in Table 7 and Table 8. It is important to note that the throughput increase is greater for learning activity $\mathbf{Q}_{Act}$ **01** than learning activity $\mathbf{Q}_{Act}$ **02**. This is because the recheck process often goes into a dieback[4] during normal operations, which significantly reduces the throughput. By contrast, a backlog of luggage does not occur as

[4] A dieback occurs when the recheck belt is full, thereby blocking the diverter from transferring luggages to the reclaim and recheck belt.

often in the CT process during normal operation since the operator performance of $\mathbf{O}_{Act}$ **01** is sufficient to keep up with the demand.

An increase to the reward ratio r_L/r_{Ops} does increase the average number of operators for both learning activities. This result may be explained by the fact that operators are less valued in comparison to the luggage when the reward ratio increases. Therefore, an agent may learn that to maintain the same number of operators in specific states is beneficial because the punishment of having an operator available is less severe than the punishment for having a luggage waiting. Furthermore, it is important to note that increase in average number of operators is only apparent for when the reward ratio r_L/r_{Ops} is greater than a threshold, 10 for $\mathbf{Q}_{Act}$ **01** and 5 for $\mathbf{Q}_{Act}$ **02**. Below this threshold, the average number of operators remains the same for varying reward ratios. This is because the value function $Q(s, a)$ is the same for frequently visited states.

An increase to the reward ratio r_L/r_{Ops} does not increase the throughput. This can be explained by the oversupply of operators. Although the average number of operators increases with increasing the reward ratio, the amount of work that needs to be done remains the same. Therefore, the throughput remains the same once a sufficient number of operator has been reached, since there is no extra work to be done. This can be seen in Table 7 and Table 8, where the operator utilisation decreases as the reward ratio increases.

Table 7: Performance indicators of the security checkpoint with a learning agent performing the allocating recheck operator activity ($\mathbf{Q}_{Act}$ **01**) under varying reward magnitudes r_L, r_{Ops} and constant $\alpha = 0.1$, $\epsilon = 0.9$, d_ϵ=0.001, $\gamma = 0.1$. All values are averages of the 200 test episodes.

Learning Agent			**Security Performance Indicators**				
r_L	r_{Ops}	r_L/r_{Ops}	*Throughput [pax/min]*	*Avg. Number [–]*	*Utilisation [%]*	*Critical Ops [%]*	*Time in Area [s]*
				$O_{Ass.}03$	$O_{Ass.}03$	Recheck	Recheck
1	10	0.1	2.64	1.45	86.32	0.1	12
1	5	0.2	2.65	1.57	81.17	0.13	9
1	2	0.5	2.65	1.57	81.17	0.13	9
2	10	0.2	2.65	1.57	81.17	0.13	9
2	5	0.4	2.65	1.57	81.17	0.13	9
2	2	1.0	2.65	1.57	81.17	0.13	9
3	10	0.3	2.65	1.57	81.17	0.13	9
3	5	0.6	2.65	1.57	81.17	0.13	9
3	2	1.5	2.65	1.57	81.17	0.13	9
5	10	0.5	2.65	1.57	81.17	0.13	9
5	5	1.0	2.65	1.57	81.17	0.13	9
5	2	2.5	2.65	1.57	81.17	0.13	9
10	10	1.0	2.65	1.57	81.17	0.13	9
10	5	2.0	2.65	1.57	81.17	0.13	9
10	2	5.0	2.65	1.57	81.17	0.13	9
10	1	10.0	2.63	1.89	51.74	0.14	9
10	0.5	20.0	2.63	2.37	26.29	0.2	10
10	0.2	50.0	2.63	2.83	26.35	0.22	10
Current Ops			2.4	1.0	86.0	31.85	301

Table 8: Performance indicators of the security checkpoint with a learning agent performing the allocating CT operator activity ($\mathbf{Q}_{Act}$ **02**) under varying reward magnitudes r_L, r_{Ops} and constant $\alpha = 0.1$, $\epsilon = 0.9$, d_ϵ=0.001, $\gamma = 0.1$. All values are averages of the 200 test episodes.

Learning Agent			**Security Performance Indicators**				
r_L	r_{Ops}	r_L/r_{Ops}	*Throughput [pax/min]*	*Avg. Number [–]*	*Utilisation [%]*	*Critical Ops [%]*	*Time in Area [s]*
				$O_{Ass.}03$	$O_{Ass.}03$	Recheck	Recheck
1	10	0.1	2.43	1.15	70.26	11.75	25
1	5	0.2	2.43	1.15	70.26	11.75	25
1	2	0.5	2.43	1.15	70.26	11.75	25
2	10	0.2	2.43	1.15	70.26	11.75	25
2	5	0.4	2.43	1.15	70.26	11.75	25
2	2	1.0	2.43	1.15	70.28	11.84	25
3	10	0.3	2.43	1.15	70.26	11.75	25
3	5	0.6	2.43	1.15	70.26	11.75	25
3	2	1.5	2.43	1.15	70.28	11.84	25
5	10	0.5	2.43	1.15	70.26	11.75	25
5	5	1.0	2.43	1.15	70.28	11.84	25
5	2	2.5	2.43	1.15	70.28	11.84	25
10	10	1.0	2.43	1.15	70.28	11.84	25
10	5	2.0	2.43	1.15	70.28	11.84	25
10	2	5.0	2.45	1.32	61.87	11.39	24
10	1	10.0	2.45	1.61	38.49	10.39	22
10	0.5	20.0	2.42	2.09	31.18	12.82	27
10	0.2	50.0	2.43	2.92	14.48	12.05	24
Current Ops			2.4	1.0	79.57	26.55	43

6 Discussion

This section reflects on our research objective by evaluating strengths and limitations of the study. The aim of the research was to develop and investigate a reinforcement learning technique for allocation of reserve airport security operators to improve operational efficiency of a security checkpoint. In order to achieve the research objective, a security checkpoint model was developed that enables the implementation of learning agents. The discussion begins by considering various model components and their implication on the research objective. Next, the learning technique components are discussed. Finally, the significance of our research in the field of airport security is addressed.

6.1 Model Discussion

Modelling Technique

Agent-based modelling is successfully able to predict the performance of the security checkpoint by simulating agent and asset behaviour. The decentralized behaviour allows for easy implementation of new elements that follow their own set of rules, such as single or multiple learning agents. Although individual mechanisms are known, it is difficult to understand how an agents evolution, interaction and decision making influence the underlying processes of the model. Furthermore, these individual mechanisms are stochastic (i.e. the probabilistic decision making of agents and assets), which generates variability in the results. Hence, multiple simulations must be performed to quantify the variability of the model. Not only does this make this technique computationally demanding, but it also makes it difficult to analyze. Macroscopic behaviour of the model (i.e. performance indicators) can be described well using distributions because the data structures do not contain many dimensions. However, microscopic behaviour (i.e. decision making of each learning agent) is difficult to analyze because of the large data structures arising from the combination of input parameters and stochasticity of the model.

Validation

Validation of the model was twofold. First, expert opinion judged the realism of microscopic reactions and relational details of the model. Then, macroscopic performance indicators of the model were compared to aggregated empirical output data. The main limitation is that it is unknown if alternative microscopic specifications are

able to generate similar or better model dynamics. In addition, several challenges occurred when measuring and aggregating data. Agents complete certain activities by multi-tasking (i.e. simultaneously divesting objects into multiple trays), thereby making it difficult to measure. Passenger or asset performance for specific subprocesses are indirectly linked with the operator's ability to provide instructions, thereby influencing the measurement. Lastly, activity performance distributions are aggregated from all empirical samples, whereas underlying distributions may exist depending on attributes of certain objects (i.e., passenger type or tray complexity). Further research should be undertaken to investigate the relation between agent/tray attributes to input parameters for the model (i.e., divest time or decision time) and their effects on the inner workings of the model. Lastly, it is important to note that the empirical data is subjected to change in the future due to modifying processes.

Environment

The model environment represents the standard security configuration of the Amsterdam Airport. Operators are assigned to only one activity, as opposed to rotating through multiple activities and working as a team. Furthermore, the checkpoint consists of only a single lane and was simulated for only a peak hour. These decisions were deliberate in order to simplify the allocation of operators and solely focusing on improving the maximum efficiency of the checkpoint. This approach has yielded initial policies for how to utilize reserve operators, but fails to take into account the entire checkpoint. A further study with more focus on allocating reserve operators to the entire checkpoint is therefore suggested. In addition, further research should be undertaken to investigate the allocation of operators over a longer period, such as days or weeks, which would also require the implementation of operator teams.

Another important aspect of the environment is its effect on the learning agent. For example, the frequency of states visited is strongly influenced by the dynamics of the model. Therefore, the training samples given to the learning agent are very skewed and prevent them from converging to good policies for states with low visits. It may be beneficial for the learning agent to experience more extreme scenarios in the model, however these are difficult to generate artificially without changing the underlying mechanisms of the model. Furthermore, the stochastic environment makes it difficult for an agent to differentiate between the effects of its action and inherent dynamics of the model. This is an important issue for future research and may be addressed by changing the learning technique.

6.2 Agent Learning Discussion

States

For both learning activities, the states had to be directly observable by a security team leader in order for them to implement the policies. Therefore, questions remain whether temporal aspects (i.e. time of a luggage waiting) or rates (i.e. number of passenger completing previous activity over the last minute) would be beneficial for the learning agent to allocate operators. However, our results have demonstrated that an agent is able to learn realistic policies that improve operational efficiency without these more complex aspects.

Actions

For both learning activities, an agent is able to add an operator, remove an operator or choose to remain the same, provided that the action performed does not violate any constraints. Constraints were introduced to prevent a large state space (i.e. not possible to have more than 3 operators performing a certain activity) and prevent the model from malfunctioning (i.e. not possible to remove an operator that is performing an activity). Overall, the results demonstrate that these actions improve the throughput of the security checkpoint and decrease the average time spent in critical operations for their corresponding subprocess. However, the constraints force the agent to explore a specific part of the state-space, which might be undesired. Therefore, a further study with less constraints on the actions is suggested.

Rewards

For both learning activities, the total reward at each timestep consisted of a luggage reward, where waiting luggage was punished, and an operator reward, where operators available were punished and operators in progress were rewarded. This reward structure enabled the agents to learn to add an operators when there is excess luggage waiting and remove operators when there are excess operators available. Although the agents knowledge resulted in a decrease in average time in critical operations for the corresponding subprocess in both learning activities, it did not lead to a direct improvement in some of the security performance indicators (i.e. throughput). In order to prevent policies being learned that are not beneficial to the performance of the security checkpoint, it would be beneficial to assign rewards directly based on the performance indicators. However, performance indicators must be computed over a duration of time, which would require the reward received by an agent to be delayed by several timesteps after an action has been performed. Furthermore, it would require a utility function that maps all performance indicators into a scalar reward. It was decided not to implement performance rewards because of the credit assignment problem.

Exploration
The exploration strategy used for both learning activities is the ϵ-*greedy* policy, which has a probability of ϵ for taking a random action and a probability of 1-ϵ for taking the current optimal action. Our results demonstrate that increasing the exploration probability leads to a less skewed distribution of the frequency of visits to a state. Nonetheless, the model dynamics prevent the agent from reaching states with high number of luggages waiting because these do not occur often during operations. Furthermore, increasing exploration did not result in an increase in total reward received due to the interference between the dynamics of the model and random exploration. This interference prevented the agent from learning a good sequence of actions for states that were visited less frequently. Future research could use different exploration strategies, such as count-based exploration, to tackle the problem.

Policies
The policies learned by the agents for the various simulations were thematically the same. In general, agents learned to increase the number of operators when there were more luggages waiting than operators available and decreased the number of operators when there were more operators available than luggages waiting. Some agents showed indications of evaluating the ratio between luggages waiting and operators available differently in other states. This often led to an increase in total reward received since the expected return exceeded the immediate reward. However, these tradeoffs rarely occurred across all simulations. A possible explanation is that the reward function used is too restrictive, thereby forcing the agent to learn the same policy. Future studies should focus on developing a reward function that enables agents to evaluate the ratio between luggages waiting and operators available differently in other states since it is expected that these tradeoffs will improve the security checkpoint performance.

6.3 Outlook on individual operator allocation at the security checkpoint using q-learning

Airports managers have several strategic objectives when it comes to airport operations, as identified by the International Civil Aviation Organization [49]. This study has demonstrated the ability of reinforcement learning techniques to suggest policies that improve one of the strategic objectives, namely operational efficiency, for security checkpoint. It is reasonable to expect that reinforcement learning can also improve other strategic objectives. As such, it is an interesting topic for future research.

The current policies are easily understandable and actionable for airport managers, such that a trial could be done in real-life. However, in order to fully implement these policies at a security checkpoint, there are several things that this study needs to improve upon. The scope of the reinforcement learning problem could be more realistic. For example, reserve operators should be allocated for an entire security terminal instead of a single security lane. Furthermore, future studies should focus on achieving better training samples for the learning agent, which would result in better performance of the test set. This could be achieved by including tailored scenarios during training. Next, the agents exploration strategies should be improved to prevent getting stuck in a local optimum. Lastly, financial aspects need to be considered when allocating reserve operators. A tradeoff should be established between cost of maintaining an average number of operators and the percentage increase in the throughput of the security checkpoint. This would allow airport managers to determine the amount of operators required to achieve a specific throughput.

7 Conclusions

This study set out to develop and investigate a q-learning based approach for allocation of individual operators to security teams in order to improve operational efficiency of a security checkpoint. The research approach consists of two parts. First, we modelled the security checkpoint using an agent-based approach, Second, we introduced a learning agent, whose goal was to allocate operators to a security team during the simulation. The model was calibrated and validated in collaboration with data and experts from Amsterdam Airport Schiphol. We demonstrated that model accurately predicts security checkpoint performance for different configuration scenarios. Furthermore, we identified that the CT and recheck process are the bottlenecks in the standard security checkpoint configuration at Amsterdam Airport Schiphol. We proposed two learning activities for the learning agent in order to improve the operational efficiency of a security checkpoint. Activity 1 allocates individual recheck operators to the recheck process. Activity 2 allocates individual CT operators to the CT process.

Our results show that introducing a learning agent with Activity 1 or Activity 2 decreases the average time spent in critical operations for the recheck process or CT process, respectively. Furthermore, both activities lead to an increase in throughput of the security checkpoint. The strategies learned by the agents were to

add an operator when there is excess luggage waiting and remove an operator when there are excess operators available. A limitation of this study is that it failed to establish a relation between average number of operators and throughput of a security checkpoint lane. Consequently, it was not possible to determine the amount of individual operators needed for an entire security terminal.

Based on the findings, there are several directions for further research. First, the modelling of security operators can be more realistic. Currently, operators work as individuals with their only goal to perform their respective assignment. Future studies should focus on incorporating a a hierarchy into the security team, consisting of a team leader and regular operators. This gives the opportunity to create team goals and develop coordination strategies. Second, the learning activity can be more realistic. Individual operators should be allocated for an entire security terminal instead of a single security lane. Furthermore, there should be a fixed number of reserve operators to be used. Third, multiple learning agents should be deployed during the simulation. Learning agents should coordinate during their learning activities when allocating individual operators. Lastly, financial aspects need to be considered because operational costs increase if a fixed number of reserve operators are maintained.

References

[1] *Economic performance of the airline industry*. End-Year Report. International Air Transport Association, Dec. 2019. URL: https://www.iata.org/en/iata-repository/publications/economic-reports/airline-industry-economic-performance---december-2019---report/. [Accessed 08-10-20].

[2] P.P.Y. Wu and K. Mengersen. "A review of models and model usage scenarios for an airport complex system". In: *Transportation Research Part A: Policy and Practice* 47 (2013), pp. 124–140. DOI: 10.1016/j.tra.2012.10.015.

[3] A.R. Odoni and R. de Neufville. "Passenger terminal design". In: *Transportation Research Part A: Policy and Practice* 26.1 (Jan. 1992), pp. 27–35. DOI: 10.1016/0965-8564(92)90042-6.

[4] V. Tosic, O. Babic, and M. Janic. "Airport passenger terminal simulation". In: *Annals of Operations Research in Air Transportation, Faculty of Transport and Traffic Engineering. University of Belgrade* (1983), pp. 83–103.

[5] S. Eilon and S. Mathewson. "A simulation study for the design of an air terminal building". In: *IEEE Transactions on Systems, Man, and Cybernetics* 4 (July 1973), pp. 308–317. DOI: 10.1109/TSMC.1973.4309241.

[6] L. McCabe and M. Gorstein. "Airport Landside". In: 1 (June 1982).

[7] J. Skorupski and P. Uchroski. "A fuzzy model for evaluating airport security screeners work". In: *Journal of Air Transport Management* 48 (2015), pp. 42–51. DOI: 10.1016/j.jairtraman.2015.06.011.

[8] N. Metzner. "A comparison of agent-based and discrete event simulation for assessing airport terminal resilience". In: *Transportation Research Procedia* 43 (2019). INAIR 2019 - Global Trends in Aviation, pp. 209–218. ISSN: 2352-1465. DOI: 10.1016/j.trpro.2019.12.035. URL: https://www.sciencedirect.com/science/article/pii/S2352146519306027.

[9] K.G. Zografos and M.A. Madas. "Development and demonstration of an integrated decision support system for airport performance analysis". In: *Transportation Research Part C: Emerging Technologies* 14.1 (Feb. 2006), pp. 1–17. DOI: 10.1016/j.trc.2006.04.001.

[10] M. Schultz and H. Fricke. "Managing passenger handling at airport terminals". In: *9th Air Traffic Management Research and Development Seminars*. 2011.

[11] P.K. Chawdhry. "Risk modeling and simulation of airport passenger departures process". In: *Proceedings of the 2009 Winter Simulation Conference (WSC)*. Austin, TX, USA: IEEE, Dec. 2009, pp. 2820–2831. DOI: 10.1109/WSC.2009.5429244.

[12] J. Skorupski and P. Uchroski. "A fuzzy system to support the configuration of baggage screening devices at an airport". In: *Expert Systems with Applications* 44 (2016), pp. 114–125. DOI: 10.1016/j.eswa.2015.08.032.

[13] A.A. Soukour et al. "Staff scheduling in airport security service". In: *IFAC Proceedings Volumes* 45.6 (2012). 14th IFAC Symposium on Information Control Problems in Manufacturing, pp. 1413–1418. ISSN: 1474-6670. DOI: 10.3182/20120523-3-RO-2023.00169. URL: https://www.sciencedirect.com/science/article/pii/S1474667016333493.

[14] A. Kierzkowski and T. Kisiel. "Simulation model of security control system functioning: A case study of the Wroclaw Airport terminal". In: *Journal of Air Transport Management* 64 (Sept. 2017), pp. 173–185. DOI: 10.1016/j.jairtraman.2016.09.008.

[15] E. Ruiz and R.L. Cheu. “Simulation model to support security screening checkpoint operations in airport terminals”. In: *Transportation research record* 2674.2 (2020), pp. 45–56. DOI: 10.1177/0361198120903242.

[16] T. Hagras and J. Janeek. “Static vs. dynamic list-scheduling performance comparison”. In: *Acta Polytechnica* 43.6 (2003).

[17] C. Shyalika, T. Silva, and A. Karunananda. “Reinforcement Learning in Dynamic Task Scheduling: A Review”. In: *SN Computer Science* 1.6 (2020), pp. 1–17. DOI: 10.1007/s42979-020-00326-5.

[18] G.F. Newell. *Applications of queueing theory*. Chapman and Hall, 1982.

[19] F.X. McKelvey. “Use of an analytical queuing model for airport terminal design”. In: *Transportation Research Record* (1988), pp. 4–11.

[20] W. Kim, Y. Park, and B.J. Kim. “Estimating hourly variations in passenger volume at airports using dwelling time distributions”. In: *Journal of Air Transport Management* 10.6 (Nov. 2004), pp. 395–400. DOI: 10.1016/j.jairtraman.2004.06.009.

[21] S. Solak, J.P.B Clarke, and E.L. Johnson. “Airport terminal capacity planning”. In: *Transportation Research Part B: Methodological* 43.6 (July 2009), pp. 659–676. DOI: 10.1016/j.trb.2009.01.002.

[22] R. Lui, R. Nanda, and J.J. Browne. “International passenger and baggage processing at john f. kennedy international airport”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 2 (Apr. 1972), pp. 221–225. DOI: 10.1109/TSMC.1972.4309096.

[23] J.P. Braaksma and W.J. Cook. “Human orientation in transportation terminals”. In: *Transportation engineering journal of the American Society of Civil Engineers* 106.2 (1980), pp. 189–203.

[24] K.J. Hee and Y.C. Zeph. “An airport passenger terminal simulator: A planning and design tool”. In: *Simulation Practice and Theory* 6.4 (1998), pp. 387–396. ISSN: 09284869. DOI: 10.1016/S0928-4869(97)00018-9.

[25] H.K. Jim and Z.Y. Chang. “An airport passenger terminal simulator: A planning and design tool”. In: *Simulation Practice and Theory* 6.4 (May 1998), pp. 387–396. DOI: 10.1016/S0928-4869(97)00018-9.

[26] M.R. Gatersleben and S.W. van der Weij. “Analysis and Simulation of Passenger Flows in an Airport Terminal”. In: *Proceedings of the 31st Conference on Winter Simulation: A Bridge to the Future*. Vol. 2. Phoenix, Arizona, USA: Association for Computing Machinery, Dec. 1999, pp. 1226–1231. DOI: 10.1145/324898.325045.

[27] M.S. Fayez et al. “Managing airport operations using simulation”. In: *Journal of Simulation* 2.1 (Dec. 2008), pp. 41–52. DOI: 10.1057/palgrave.jos.4250030.

[28] P.E. Joustra and N.M. Van Dijk. “Simulation of check-in at airports”. In: *Proceeding of the 2001 Winter Simulation Conference*. Vol. 2. Arlington, VA, USA: IEEE, Dec. 2001, pp. 1023–1028. DOI: 10.1109/WSC.2001.977409.

[29] S. Appelt et al. “Simulation of passenger check-in at a medium-sized US airport”. In: *2007 Winter Simulation Conference*. IEEE. Dec. 2007, pp. 1252–1260. DOI: 10.1109/WSC.2007.4419729.

[30] I.E. Manataki and K.G. Zografos. “A generic system dynamics based tool for airport terminal performance analysis”. In: *Transportation Research Part C: Emerging Technologies* 17.4 (Aug. 2009), pp. 428–443. DOI: 10.1016/j.trc.2009.02.001.

[31] I.E. Manataki and K.G. Zografos. “Assessing airport terminal performance using a system dynamics model”. In: *Journal of Air Transport Management* 16.2 (Mar. 2010), pp. 86–93. DOI: 10.1016/j.jairtraman.2009.10.007.

[32] 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*. IEEE. Macao, China, Dec. 2010, pp. 1621–1625. DOI: 10.1109/IEEM.2010.5674286.

[33] D. Wilson, E.K. Roe, and S.A. So. “Security checkpoint optimizer (SCO): An application for simulating the operations of airport security checkpoints”. In: *Proceedings of the 2006 Winter Simulation Conference*. Monterey, CA, USA: IEEE, Dec. 2006, pp. 529–535. DOI: 10.1109/WSC.2006.323126.

[34] L. Cheng et al. “Analysis of passenger group behaviour and its impact on passenger flow using an agent-based model”. In: *2014 4th International Conference On Simulation And Modeling Methodologies, Technologies And Applications*. IEEE. Vienna, Austria, 2014, pp. 733–738. DOI: 10.5220/0005086807330738.

[35] S. Janssen, A. Sharpanskykh, and R. Curran. “Agent-based modelling and analysis of security and efficiency in airport terminals”. In: *Transportation research part C: emerging technologies* 100 (2019), pp. 142–160. DOI: 10.1016/j.trc.2019.01.012.

[36] A. Verma, D. Tahlyan, and S. Bhusari. "Agent based simulation model for improving passenger service time at Bangalore airport". In: *Case Studies on Transport Policy* 8.1 (2020), pp. 85–93. DOI: 10.1016/j.cstp.2018.03.001.

[37] S. Janssen et al. "AATOM: An Agent-Based Airport Terminal Operations Model Simulator". In: SummerSim '19. Berlin, Germany: Society for Computer Simulation International, July 2019, p. 12. DOI: 10.5555/3374138.3374158.

[38] V.L.L. Babu, R. Batta, and L. Lin. "Passenger grouping under constant threat probability in an airport security system". In: *European Journal of Operational Research* 168.2 (Jan. 2006), pp. 633–644. DOI: 10.1016/j.ejor.2004.06.007.

[39] X. Nie et al. "Passenger grouping with risk levels in an airport security system". In: *European Journal of Operational Research* 194.2 (Apr. 2009), pp. 574–584. DOI: 10.1016/j.ejor.2007.12.027.

[40] E. Miller, G. LaFree, and L. Dugan. *Global Terrorism Database*. National Consortium for the Study of Terrorism and Responses to Terrorism. May 2018. URL: https://start.umd.edu/data-tools/global-terrorism-database-gtdg. [Online; Accessed 08-10-20].

[41] J. Skorupski and P. Uchroski. "A fuzzy model for evaluating metal detection equipment at airport security screening checkpoints". In: *International Journal of Critical Infrastructure Protection* 16 (2017), pp. 39–48. DOI: 10.1016/j.ijcip.2016.11.001.

[42] J. Skorupski and P. Uchroski. "A fuzzy reasoning system for evaluating the efficiency of cabin baggage screening at airports". In: *Transportation Research Part C: Emerging Technologies* 54 (2015), pp. 157–175. DOI: 10.1016/j.trc.2015.03.017.

[43] *Regulation (EC) No 300/2008*. Legislation Report L 97/72. European Comission, Mar. 2008. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32008R0300&from=EN. [Accessed 08-10-20].

[44] S. Janssen et al. "Data-Driven Analysis of Airport Security Checkpoint Operations". In: *Aerospace* 7.6 (May 2020), p. 69. DOI: 10.3390/aerospace7060069.

[45] C.E. Brown. "Coefficient of variation". In: *Applied multivariate statistics in geohydrology and related sciences*. Springer, 1998, pp. 155–157. DOI: 10.1007/978-3-642-80328-4_13.

[46] C. Watkins and P. Dayan. "Q-learning". In: *Machine learning* 8.3-4 (1992), pp. 279–292. DOI: 10.1007/BF00992698.

[47] R.S. Sutton and A.G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[48] D. Koutra, J.T. Vogelstein, and C. Faloutsos. "Deltacon: A principled massive-graph similarity function". In: *Proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM. 2013, pp. 162–170.

[49] *Strategic objectives of ICAO for 2005-2010*. Strategic Report. International Civil Aviation Organization, Dec. 2004. URL: https://www.icao.int/Documents/strategic-objectives/strategic_objectives_2005_2010_en.pdf. [Accessed 08-10-20].

Appendices

A Performance Indicators for the Security Checkpoint with $Q_{Act.}$ 01

Table 9: Performance indicators of the security checkpoint with a learning agent performing the allocating recheck operator activity (**Q_{Act} 02**) under varying configurations.

Simulator						**Results**														
r_L	r_{Ops}	α	ϵ	d_ϵ	γ	*Throughput [pax/min]*	*Avg. Number [−]*			*Utilisation [%]*			*Critical Ops [%]*				*Time in Area [s]*			
							$O_{Ass.}01$	$O_{Ass.}02$	$O_{Ass.}03$	$O_{Ass.}01$	$O_{Ass.}02$	$O_{Ass.}03$	CT	SSc	Reclaim	Recheck	Divest	SSc	Reclaim	Recheck
1	10	0.05	0.9	0.001	0.1	2.64	1.0	2.0	1.45	84.96	45.25	86.32	17.06	8.16	0.02	0.1	84	22	29	12
1	10	0.05	0.9	0.01	0.1	2.63	1.0	2.0	1.37	85.0	45.34	88.8	16.4	8.0	0.21	2.29	85	22	28	37
1	10	0.1	0.1	0.001	0.1	2.63	1.0	2.0	1.36	84.93	45.14	88.59	16.77	7.64	0.03	0.53	85	22	28	24
1	10	0.1	0.1	0.01	0.1	2.63	1.0	2.0	1.36	84.92	45.11	88.59	16.68	7.63	0.03	0.53	85	22	28	24
1	10	0.1	0.5	0.001	0.1	2.63	1.0	2.0	1.59	84.48	45.03	82.33	15.65	7.82	0.01	0.16	85	22	27	9
1	10	0.1	0.5	0.01	0.1	2.63	1.0	2.0	1.59	84.48	45.03	82.33	15.65	7.82	0.01	0.16	85	22	27	9
1	10	0.1	0.9	0.001	0.1	2.64	1.0	2.0	1.45	84.96	45.25	86.32	17.06	8.16	0.02	0.1	84	22	29	12
1	10	0.1	0.9	0.01	0.1	2.62	1.0	2.0	1.59	84.26	44.91	82.34	15.79	7.75	0.02	0.41	85	22	27	9
1	10	0.2	0.9	0.001	0.1	2.63	1.0	2.0	1.32	84.8	45.45	90.61	16.79	8.35	0.03	0.12	84	23	28	19
1	10	0.2	0.9	0.01	0.1	2.63	1.0	2.0	1.32	84.76	45.42	90.6	16.72	8.3	0.04	0.18	84	22	28	20
1	2	0.1	0.9	0.001	0.1	2.65	1.0	2.0	1.57	85.07	45.79	81.17	16.44	8.34	0.03	0.13	84	23	28	9
1	5	0.1	0.9	0.001	0.1	2.65	1.0	2.0	1.57	85.07	45.79	81.17	16.44	8.34	0.03	0.13	84	23	28	9
10	0.2	0.1	0.9	0.001	0.1	2.63	1.0	2.0	2.83	84.72	45.12	26.35	15.81	8.09	0.01	0.22	84	22	27	10
10	0.5	0.1	0.9	0.001	0.1	2.63	1.0	2.0	2.37	84.79	44.98	26.29	15.93	8.07	0.02	0.2	85	22	28	10
10	1	0.1	0.9	0.001	0.1	2.63	1.0	2.0	1.89	84.64	45.06	51.74	15.27	7.96	0.01	0.14	85	22	27	9
10	10	0.1	0.9	0.001	0.1	2.65	1.0	2.0	1.57	85.07	45.79	81.17	16.44	8.34	0.03	0.13	84	23	28	9
10	2	0.1	0.9	0.001	0.1	2.65	1.0	2.0	1.57	85.07	45.79	81.17	16.44	8.34	0.03	0.13	84	23	28	9
10	5	0.1	0.9	0.001	0.1	2.65	1.0	2.0	1.57	85.07	45.79	81.17	16.44	8.34	0.03	0.13	84	23	28	9
2	10	0.1	0.9	0.001	0.1	2.65	1.0	2.0	1.57	85.07	45.79	81.17	16.44	8.34	0.03	0.13	84	23	28	9
2	2	0.1	0.9	0.001	0.1	2.65	1.0	2.0	1.57	85.07	45.79	81.17	16.44	8.34	0.03	0.13	84	23	28	9
2	5	0.1	0.9	0.001	0.1	2.65	1.0	2.0	1.57	85.07	45.79	81.17	16.44	8.34	0.03	0.13	84	23	28	9
3	10	0.1	0.9	0.001	0.1	2.65	1.0	2.0	1.57	85.07	45.79	81.17	16.44	8.34	0.03	0.13	84	23	28	9
3	2	0.1	0.9	0.001	0.1	2.65	1.0	2.0	1.57	85.07	45.79	81.17	16.44	8.34	0.03	0.13	84	23	28	9
3	5	0.1	0.9	0.001	0.1	2.65	1.0	2.0	1.57	85.07	45.79	81.17	16.44	8.34	0.03	0.13	84	23	28	9
5	10	0.1	0.9	0.001	0.1	2.65	1.0	2.0	1.57	85.07	45.79	81.17	16.44	8.34	0.03	0.13	84	23	28	9
5	2	0.1	0.9	0.001	0.1	2.65	1.0	2.0	1.57	85.07	45.79	81.17	16.44	8.34	0.03	0.13	84	23	28	9
5	5	0.1	0.9	0.001	0.1	2.65	1.0	2.0	1.57	85.07	45.79	81.17	16.44	8.34	0.03	0.13	84	23	28	9
Current Ops						2.4	1.0	2.0	1.0	79.57	42.44	86.0	26.55	6.45	1.96	31.85	90	21	43	301

Table 10: Performance indicators of the security checkpoint with a learning agent performing the allocating CT operator activity (Q_{Act} **02**) under varying configurations.

Simulator						**Results**													
r_L	r_{Ops}	α	ϵ	d_ϵ	γ	*Throughput [pax/min]*	*Avg. Number [–]*			*Utilisation [%]*			*Critical Ops [%]*				*Time in Area [s]*			
							$O_{Ass.}01$	$O_{Ass.}02$	$O_{Ass.}03$	$O_{Ass.}01$	$O_{Ass.}02$	$O_{Ass.}03$	CT	SSc	Reclaim	Recheck	Divest	SSc	Reclaim	Recheck
1	10	0.05	0.9	0.001	0.1	2.43	1.15	2.0	1.0	70.26	43.46	86.9	11.75	7.59	2.86	30.16	90	22	25	304
1	10	0.05	0.9	0.01	0.1	2.43	1.15	2.0	1.0	70.28	43.47	86.89	11.8	7.56	2.8	30.17	90	22	25	304
1	10	0.1	0.1	0.001	0.1	2.42	1.15	2.0	1.0	70.31	43.34	86.97	12.11	7.53	3.02	31.03	90	22	25	308
1	10	0.1	0.1	0.01	0.1	2.44	1.16	2.0	1.0	70.57	43.62	86.68	11.46	6.89	2.99	30.78	89	22	24	306
1	10	0.1	0.5	0.001	0.1	2.42	1.15	2.0	1.0	70.31	43.34	86.97	12.11	7.53	3.02	31.03	90	22	25	308
1	10	0.1	0.5	0.01	0.1	2.42	1.15	2.0	1.0	70.32	43.36	86.99	12.11	7.54	3.02	31.05	90	22	25	308
1	10	0.1	0.9	0.001	0.1	2.43	1.15	2.0	1.0	70.26	43.46	86.9	11.75	7.59	2.86	30.16	90	22	25	304
1	10	0.1	0.9	0.01	0.1	2.43	1.15	2.0	1.0	70.28	43.47	86.89	11.8	7.56	2.8	30.17	90	22	25	304
1	10	0.2	0.9	0.001	0.1	2.44	1.08	2.0	1.0	75.23	43.25	87.06	11.38	6.49	3.04	31.49	89	21	24	315
1	10	0.2	0.9	0.01	0.1	2.44	1.08	2.0	1.0	75.26	43.12	87.04	11.5	6.43	2.94	31.68	89	21	24	316
1	2	0.1	0.9	0.001	0.1	2.43	1.15	2.0	1.0	70.26	43.46	86.9	11.75	7.59	2.86	30.16	90	22	25	304
1	5	0.1	0.9	0.001	0.1	2.43	1.15	2.0	1.0	70.26	43.46	86.9	11.75	7.59	2.86	30.16	90	22	25	304
10	0.2	0.1	0.9	0.001	0.1	2.43	2.92	2.0	1.0	14.48	43.26	86.67	12.05	7.67	3.37	33.07	89	23	24	322
10	0.5	0.1	0.9	0.001	0.1	2.42	2.09	2.0	1.0	31.18	42.6	87.41	12.82	7.28	3.5	32.64	90	22	27	332
10	1	0.1	0.9	0.001	0.1	2.45	1.61	2.0	1.0	38.49	43.18	87.34	10.39	7.17	2.74	31.81	89	22	22	306
10	10	0.1	0.9	0.001	0.1	2.43	1.15	2.0	1.0	70.28	43.47	86.9	11.84	7.6	2.86	30.18	90	22	25	305
10	2	0.1	0.9	0.001	0.1	2.45	1.32	2.0	1.0	61.87	43.25	87.35	11.39	7.77	2.76	32.9	89	23	24	315
10	5	0.1	0.9	0.001	0.1	2.43	1.15	2.0	1.0	70.28	43.47	86.9	11.84	7.6	2.86	30.18	90	22	25	305
2	10	0.1	0.9	0.001	0.1	2.43	1.15	2.0	1.0	70.26	43.46	86.9	11.75	7.59	2.86	30.16	90	22	25	304
2	2	0.1	0.9	0.001	0.1	2.43	1.15	2.0	1.0	70.28	43.47	86.9	11.84	7.6	2.86	30.18	90	22	25	305
2	5	0.1	0.9	0.001	0.1	2.43	1.15	2.0	1.0	70.26	43.46	86.9	11.75	7.59	2.86	30.16	90	22	25	304
3	10	0.1	0.9	0.001	0.1	2.43	1.15	2.0	1.0	70.26	43.46	86.9	11.75	7.59	2.86	30.16	90	22	25	304
3	2	0.1	0.9	0.001	0.1	2.43	1.15	2.0	1.0	70.28	43.47	86.9	11.84	7.6	2.86	30.18	90	22	25	305
3	5	0.1	0.9	0.001	0.1	2.43	1.15	2.0	1.0	70.26	43.46	86.9	11.75	7.59	2.86	30.16	90	22	25	304
5	10	0.1	0.9	0.001	0.1	2.43	1.15	2.0	1.0	70.26	43.46	86.9	11.75	7.59	2.86	30.16	90	22	25	304
5	2	0.1	0.9	0.001	0.1	2.43	1.15	2.0	1.0	70.28	43.47	86.9	11.84	7.6	2.86	30.18	90	22	25	305
5	5	0.1	0.9	0.001	0.1	2.43	1.15	2.0	1.0	70.28	43.47	86.9	11.84	7.6	2.86	30.18	90	22	25	305
Current Ops						2.4	1.0	2.0	1.0	79.57	42.44	86.0	26.55	6.45	1.96	31.85	90	21	43	301

II

Literature Study

Previously graded under AE4020.

Chapter 1

Introduction

Air transportation is one of the backbones to modern society and a driving force behind globalization. In 2018, the expenditure on air transport alone was 1% of the worlds GDP, equivalent to $845 billion *Economic performance of the airline industry*. In this critical infrastructure, airports make up central nodes, where the circulation of people, resources and capital are managed. Airports are constantly evaluating their terminal procedures to ensure safety at the airport and on-board the aircraft, while maintaining an efficient level of service. Efficient airport security implies minimal waiting times for each activity, thereby creating constant flow of passengers/luggage from terminal entrance to gate. However, minimal waiting times contradict effective security procedures. The juxtaposition between efficiency and security is evident and outlined as two of the six strategic objectives by the International Civil Aviation Organization *Strategic objectives of ICAO for 2005-2010*.

Since 9/11, terrorism has transformed airport security into a global dilemma Schouten (2014). The Transportation Security Administration (TSA) was established by the Aviation and Transportation Security Act to provide security for the transportation system of the nation following the incident. Their budget for 2019 totalled at $7.8billion dollars according to *FY 2020 Budget in Brief*. In contrast, European total expenditure (18 states) totalled $2.7billion in 2002 and was estimated to be $7.6billion dollars in 2013 by *Study on civil aviation security financing*. Despite these investments, terrorists are able to exploit vulnerabilities at airport terminals, leading to continuous attempts of attacks Miller et al. (2018). Aviation has been a continuous target for terrorist attacks following 9/11 with several bombing attempts, such as the Brussels airport attack or the underwear bomber on flight 253 from Amsterdam to Detroit Miller et al. (2018). Therefore, there is a continuous need for improvement in security operations and infrastructure.

Amsterdam Schiphol Airport is developing a new concept for aviation security operations, which should be implemented on both departure and transfer filters by 2024. The new concept should provide a solution to the security and efficiency challenges that the air transport industry is facing today. To create this new concept, modelling and simulation is necessary to track security and efficiency.

This report examines existing literature on the aforementioned challenges. Chapter 2 begins by introducing regulations for airport terminals before examining each specific security checkpoint activity. Furthermore, it introduces *Schiphol Airport* current security concept, which will be used as a baseline model to evaluate other concepts. In order to be able to simulate security concepts, it is necessary to know how these operations can be modelled. Hence, Chapter 3 presents existing simulation models of airport terminals, focusing on the three key themes: capacity, efficiency and security. Often these models are expanded into airport terminal simulation software. Therefore, Chapter 4 describes existing software developed by the government, industry and academia. Simulation software is of paramount importance as it allows airport designers to test their concepts before evaluating them in real life. Since, airport terminal simulation models have widely been explored, techniques for optimizing and determining optimal policies are analyzed in the following chapters. These techniques are interesting to analyze because they allow researchers to gain new insights in their simulation models or develop new operational concepts from their simulation models. Chapter 5 deep dives into the field of simulation optimization, a term for techniques used to optimize stochastic simulations. Chapter 6 begins by laying out the fundamentals of reinforcement learning (learning through interaction with the environment), then describes the challenges experienced with learning systems (sparse rewards and reward shaping) and continues to express other types of learning systems (automated planning and multi-agent learning).

Chapter 2

Airport Terminal

In this chapter, the *Schiphol Airport* terminal is introduced. Section 2.1 presents regulations imposed on the civil aviation security within the EU. They represent guidelines that must be followed by *Schiphol*. Next, the security checkpoint of *Schiphol Airport* is introduced in Section 2.2. Finally, the passenger activities and steps at a security checkpoint are outline in Section 2.3. It is important to note that this focuses on the security Terminal 3 at *Schiphol Airport*.

2.1. EU Regulations

2.1.1. Common Rules of Civil Aviation Security

Regulation (EC) No 300/2008 govern aviation safety by defining general rules and minimum requirements applicable to all civil airports in the EU (i.e. including *Schiphol Airport*) as well as to air carriers and businesses delivering services/goods through or to these airports. It replaces *Regulation (EC) No 2320/2002*, which established common rules of civil aviation security in the aftermath of the September 11, 2001 attacks. Common basic standards for protecting civil aviation are summarized in the list below.

Reg. 01 *Screening of Passenger and Cabin Baggage*
To avoid the carrying of banned articles on board the aircraft, such as guns and explosives.

Reg. 02 *Screen Hold (Check-In) Luggage*
To stop prohibited luggage to be loaded onto the aircraft.

Reg. 03 *Airport Security*
To control access to different areas of aiports (i.e. staff screening, vehicle checks, surveillance and patrols). To prohibit the entrance of unauthorised people into these areas. To ensure that there are no prohibited articles on board.

Reg. 04 *Security Controls of Cargo*
To stop prohibited articles being loaded on to the aircraft.

Reg. 05 *Security Controls for Airport Supplies*
To control supplies intended for shops/restaurant or in-flight. To ensure that no prohibited articles enter the airport.

Reg. 06 *Staff Recruitment and Training*
To ensure that staff is not suspicious.

Reg. 07 *Security Equipment Performance*
To ensure that equipment is capable of performing the security controls concerned.

In November 2015, the European Commission adopted *Implementing Regulation (EU) 2015/1998*, which presents procedures on how to monitor airports implementation of these regulations. This regulation repealed *Regulation (EC) No 185/2010*, as it was amended more than 20times.

The amendments recognize a list of non-EU countries that apply equivalent standards to those of the EU on civil aviation security Furthermore, it introduces new rules regarding civil aviation security. New obligations on EU counties and airports/airlines are as follows.

Each EU country must:

- Designate a single authority responsible for the protection of aviation.
- Establish a national programme for civil aviation security to define responsibilities for the implementation of the common basic standards.
- Establish a national programme for quality management to verify the quality of civil aviation security.

Each airport or air carrier:

- Define and implement a security programme.
- Ensure internal quality control.

The Commission performs inspections on airports, airport carriers and other companies in collaboration with national authorities. Any shortcoming must be corrected by the national authority, which are responsible for quality control and enforcement of aviation security. Hence, they must carry out audits and inspections of relevant businesses.

2.1.2. Common Evaluation Process of Security Equipment

The *ECAC Common Evaluation Process of Security Equipment* is a security equipment testing program against the performance requirements of the European Civil Aviation Conference (ECAC), developed by ECAC member states. These requirements provide the national administration with a common reference for the certification of security equipment installed at airports.

In addition, the ECAC publishes tested configurations that have met the Common Evaluation Process (CEP) standard to their website. These configurations can be used by industry stakeholders (i.e. airports). Non-ECAC nations, such as Australia, Canada, Israel or the United States, have their own security equipment testing programmes. However, they understand the importance of the CEP and engage in ECAC professional meetings to share knowledge and work toward harmonization of performance requirements and testing procedures.

The aims of the CEP are:

- To assess the technical efficiency of the security equipment in an impartial and systematic way through the various Participating Test Centers involved in the process.
- To provide ECAC Member States with accurate reports on equipment performance against the technical standards adopted.

The CEP currently applies to the following categories of security equipment:

- Explosive Detection Systems (EDS)
- Liquid Explosive Detection Systems (LEDS)
- Security Scanners (SSc)
- Explosive Trace Detection (ETD) equipment
- Metal Detection Equipment (MDE)
- Explosive Detection Systems for Cabin Baggage (EDSCB)
- Walk Through Metal Detectors (WTMD)

2.2. Security Configuration

The current security configuration at terminal 3 of *Schiphol Airport* is depicted in Figure 2.1. As can be seen in the figure, there is a divest belt with 3 divest positions, where passengers are able to drop off their luggage. The average divest time is approximately 40 seconds. Each passenger has a unique divest time that depends on how fast they are able to drop off their luggage.

The divest belt flows into the infeed belt which flows into the x-ray belt. Airport operators are able to control the speed of the infeed belt and x-ray belt. A CT scanner is located on the x-ray belt and is operated by a single operator (or sometimes multiple). Operators determine with the help of reject rate algorithms whether or not a luggage is suspicious. The average decision time for rejecting a luggage is 12 to 15 seconds. It is important to note that the decision time varies per tray/luggage.

Once through the x-ray belt, the luggage flows onto the decision belt. In an ideal situation, the operator makes a decision of whether or not to reject the luggage before it arrives at the end of the belt. Once a decision has been made, the luggage moves to reclaim belt or reject belt if marked unsuspicious or suspicious respectively. However, if the operator has not made a decision, then the luggage waits at the end of the decision belt until one has been made (or after 30seconds it automatically gets rejected).

When the luggage is on the reclaim belt, the passenger is free to pack his belongings and pick up his luggage. By contrast, when the luggage is on the reject belt, then the passenger must wait for an airport operator to check their luggage before they can collect their belongings.

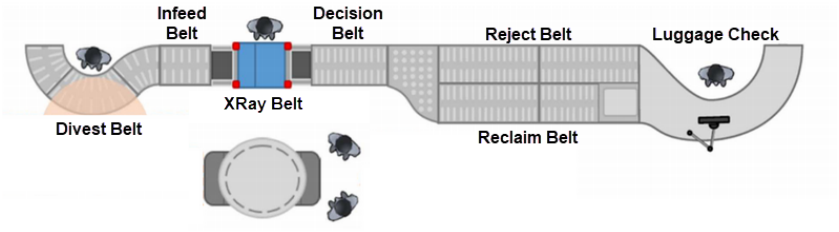


Figure 2.1: Current security checkpoint concept at terminal 3 of *Schiphol Airport*

2.3. Security Checkpoint Activities

In this section, the airport security checkpoint activities are described. A passenger first enters a queue (Section 2.3.1), then he is directed to a security lane station where a baggage drop position is allocated (Section 2.3.2). Once the passenger has a drop position, he proceeds to unload his luggage onto the xray belt (Section 2.3.3). Next, the passenger and the luggage are screened for forbidden items (Section 2.3.4). If the alarm of one of the systems is triggered during screening, then an additional screening is done (Section 2.3.5). Once screening (or additional screening) are finished, the passenger proceeds to reclaim their luggage (Section 2.3.6).

2.3.1. Queue

The first process encountered by a passenger entering the security checkpoint is the queue. The queue is a one-way directional system, therefore once a passenger enters it, they will follow the queue until they reach its endpoint. Furthermore, passengers are not able to overtake other passenger in-line while in the queue. Passengers accumulate in the queue as a result of the arrival rate of passengers to the security checkpoint and the throughput of the security checkpoint system. This phenomenon is known as queue forming. It is important to note that this phenomenon cannot be influenced by passengers or operators.

2.3.2. Lane and Position Allocation

After queueing has finished, the passengers reach the lane and position allocation. At *Schiphol Airport* it was initially believed that the passengers intuitively find their way through the security filter and automatically move from the waiting area to an available drop-off position in an open lane. In practice, it has been found that a number of the passengers have difficulty making the choice themselves. In order to achieve the highest possible traffic flow, Schiphol has decided to hire assigners (this is not qualified security personnel) for these two different positions, see the figure below.

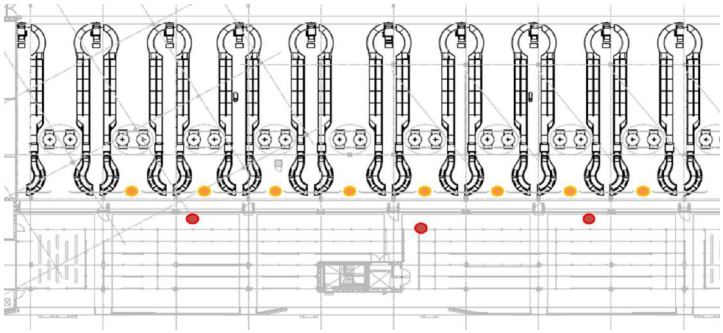


Figure 2.2: Position assigners at security checkpoint
red = channel assignment, *yellow* = position assignment

At the front of a security filter, after the queue barriers, one lane assigner indicates to the passenger which side (left / right) the passenger may take to go through security. If several lanes are open, extra lane assigners will guide the passenger further along the queue area and send them to the exact lane where they have to join a mini queue.

As soon as the passenger is in the bay, they are, again, received by a position assigner and as soon as a position is available, they are transferred to 1 of 6 drop-off positions (each lane has 3 drop-off positions)

where the passenger can prepare before everything is checked.

These assigner roles will soon become automated at *Schiphol Airport*, allowing passengers to move independently first the correct lane and then to the assigned drop-off position.

2.3.3. Baggage Drop

Once a passenger has received a drop-off position, the passenger is able to enter the security lane. They then proceed to put all their relevant baggage and belongings into trays onto the conveyor belt, as directed by one of the airport operators. This includes all carry-on baggage, electronics, liquids and other items. Both passenger and luggage then proceed onto the screening process.

2.3.4. Screening

Passenger and luggage are subjected to screening via a body scanner and x-ray (CT) scanner system respectively. The body scanner searches for any metallic objects on the passenger and triggers an alarm if one is detected. Similarly, the xray scanner produces three types of images that are shown to an operator, who is then able to trigger an alarm should there be something suspicious.

2.3.5. Additional Screening

If the body scanner detects the presence of a suspicious object on the passenger, an extra inspection of the passenger is conducted in the form of a pat-down. A security employee gets indicated where the suspicious object is located on the body via the body scanner. They then proceed to follow the the contours of the passengers in the vicinity of the objects location in order to identify the object. Once the pat-down is completed, the passenger is let through provided that the object was deemed unsuspicious.

When an x-ray scanner detects the presence of items that do not comply with security regulations, a security employee searches through the belongings until the object is identified. Similarly to the body scanner, the x-ray scanner presents the type of objects that the security employee is searching for. Once found, another inspection, usually visual but in some scenarios another x-ray, is undertaken to assess whether or not the item should be allowed on board. If not, the item is disposed of at the airport.

2.3.6. Baggage Reclaim

Once screening has been finished, the passenger is ready to start reclaiming their luggage. They proceed to their respective tray that is located on the reclaim belt and start packing their belongings back together. Once everything is packed up, the passenger is able to move to their gate or perform discretionary activities. Should the luggage require additional screening, then the passenger must go to an additional luggage check. The additional luggage screening is performed there after which the passenger can take his luggage and go to their gate.

Chapter 3

Simulation Models

Complex phenomena are typically modeled on advanced simulators that,depending on their precision, can be very demanding in terms of simulation time and computational resources. In their review of simulating airport terminal literature, Wu and Mengersen (2013) identified four types of models, each simulating different aspects of airport terminals: capacity, efficiency, security and airport performance. For the purpose of this literature review, only the first three model types are reviewed because airport performance is not relevant as it determines passenger satisfaction levels. Section 3.1, Section 3.2 and Section 3.3 review capacity, efficiency and security airport terminal models respectively.

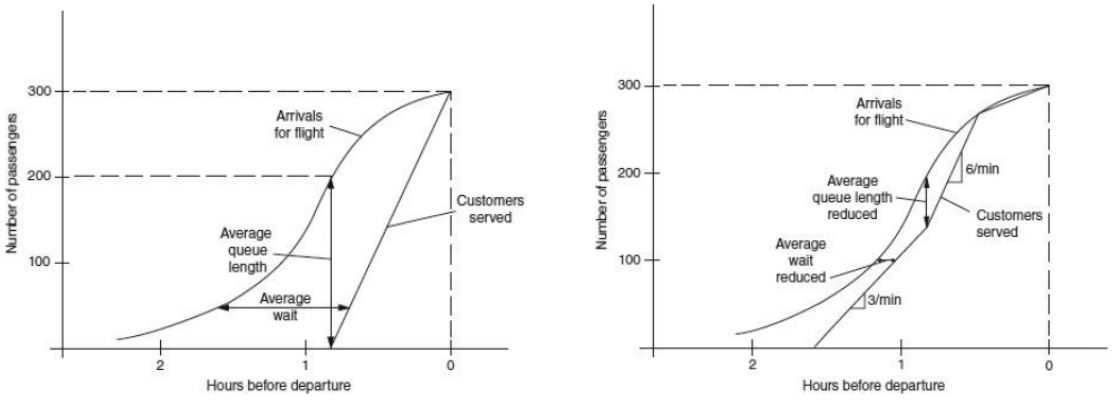
3.1. Capacity Models

Capacity models are used to determine if the planned infrastructure satisfies future demand A.R. Odoni and R. d. Neufville (1992). In these scenarios, the model is a guide to enable architects, consultants and decision makers in making decisions regarding future facilities. It tracks key performance indicators such as *average wait time*, *queue length* and *service time* at a facility within an airport. These indicators help inform airport planners/designers in deciding on prospective infrastructure. Section 3.1.1, Section 3.1.2 and Section 3.1.3 discuss capacity models simulated via queuing theory, stochastic queue models and statistical distributions respectively.

3.1.1. Queuing Models

Capacity planning proved an important literary genre in the early airport terminal modelling community, as mentioned by Tosic (1992). Typically, the model is used to help address the question of whether the proposed infrastructure will satisfy the projected future demand A.R. Odoni and R. d. Neufville (1992).

Newell (1982) was the first to develop a range of capacity-driven airport terminal models based on the deterministic queue model. These models track key performance indicators such as passenger waiting and service times, and the number of passengers in queue per queuing area within the airport. A graphical representation is adopted by using the cumulative arrival profile and departure profiles for each infrastructure facility. By integrating arrival and departure diagram, as shown in Figure 3.1, researchers can measure the average queue length and the waiting time on the basis of on the vertical or horizontal size respectively per facility. However, this method does not consider the ambiguity of the arrival and departure profiles nor is it possible to determine the maximum wait time for each passenger. Furthermore, it assumes that passenger arrival in the queue dictate the order at which they are served.



(a) Not adjusted (b) Adjusted

Figure 3.1: Cumulative arrival and departure profiles from R. D. Neufville and A. Odoni (2003)

These models include the arrival and departure profiles as main inputs, which are usually determined from approximated profiles or from data measurements collected data collected at an airport as shown by Newell (1982). Analyses using cumulative arrival and departure diagrams offer a clear way to successfully define and overcome signifcant difficulties. They are a valuable tool that every skilled designed should be able to use. However, these analyses have limitations: They are deterministic in that they do not explain the differences that exist in reality. Furthermore, they are only applicable for one airport activity at a time. Hence, there is no hint of the relationship between system processes, such as when disruptions in one process influence the patterns of arrival in subsequent processes.

The simple queue model is extended by Tosic et al. (1983), who investigates a stochastic queue model where arrival profiles are already developed. Monte Carlo methods are used to calculate the total service and wait times and the length of the queue. Horonjeff et al. (2010) further extend the approach by determining mathematical formulations which adequately represent the processing system. One such mathematical expression is that of a multi-station queuing system as shown in Equation (3.1), with a Poisson arrival distribution characterized by its demand rate λ. Additionally, the formulation utilizes a service time distribution, which is characterized by the average service time, t, and the variance of the average service time σ^2.

$$W_s = \left(\frac{\sigma^2 + t^2}{2t^2}\right) \frac{\lambda^k t^{k+1}}{(k-1)!(k-\lambda)\, 2\sum_{n=0}^{k-1} [(\lambda t)^n/n!] + (\lambda t)^k/[(k-1)!(k-\lambda t)]} \tag{3.1}$$

Equation (3.1) determines the average delay per person and is valid when the average demand rate λ is less than total the service rate $k\mu$, where k is the number of processors and $\mu = 1/t$.

McKelvey (1988) expands on a single queuing system by adopting a multi-channel queuing approach (as shown in Figure 3.2) for a passenger enplaning network at an airport terminal. The passenger enters the terminal building D. He then has the possibility of going to the check-in activity for tickets only T or for tickets and luggage B. After check-in, the passenger proceeds to being checked at the security checkpoint X. Once through security, he continues to the gate area S where he waits for boarding G.

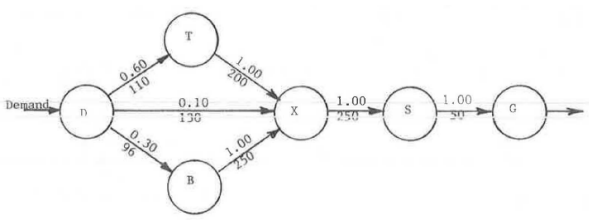


Figure 3.2: Typical Passenger Enplaning Network from McKelvey (1988).

Models such as those suggested by W.J. Dunlay Jr and Park (1978) consider a variety of connected facilities where the arrival profile of a facility is modelled as the departure profile of the previous activity, with some offset. Recent developments in the field of surveillance technologies, such as (i.e. Gongora and Ashfaq (2006) and B.-s. Kim et al. (2008), have led real-time data collection of arrival and departure profile, which can be used to simulate a deterministic queue model.

3.1.2. Stochastic Models

Bevilacqua and Ciarapica (2010) provides a stochastic queue model for assessing passenger processing and utilization of service (i.e. number of counters) for airport check-in. In order to find the steady state system performance, a monte carlo simulation is used unlike in deterministic queue models. The findings are then used to determine a common check-in configuration versus a dedicated carrier check-in configuration. However, Bevilacqua and Ciarapica (2010) and queue-based research of section 3.1.1 have failed to resolve the dynamic aspects of the airport environment Eilon and Mathewson (1973); Joustra and Dijk (2001)

3.1.3. Statistical Models

W. Kim et al. (2004) defines a mathematical model focused on the distribution of dwelling time to estimate passenger volumes for departing passengers over the span of a day. Passenger volume is influenced by the flight schedule, but it is not the only determinant. Longer stay or dwell times are also influenced by departing passengers arriving much earlier at the airport than their boarding time. Similarly, by designing time functions,

Solak et al. (2009) suggests an analytical estimate of maximum passenger delay for airport activities and processing infrastructure. Time functions can be derived from the relationship between flow rates and width of passageway, along with collected data measuring processing time fluctuations and measured walking speeds. Not only is Solak et al. (2009) the first to consider a holistic model of airport capacity by considering linked facilities, however, they go further and proposes an algorithm for the Airport Terminal Capacity Planning (ATCAP) Problem using multi-stage stochastic programming. The ATCAP deals with "determining the optimal design and expansion capacities for different areas of the terminal in the presence of uncertainty with regards to future demand levels and expansion costs" (Solak et al. 2009). During the initial development process of an airport terminal, the solution algorithm can be run to provide optimal capacity requirements for each area along with expansion strategies under stochastic future demand.

When viewing previous capacity studies already discussed in this section, they fail to suggest an optimal solution to the capacity planning. Instead, they provide architects, consultants or decision makers with a simulation model that identifies potential bottlenecks in the operations. Since Solak et al. (2009) present both a solution and model to simulate, it is viewed as an improvement on the existing literature. The study would have been more useful if they had focused on generalizing their approach such that it can be used across airports. For example, Solak et al. (2009) could have generated a delay approximation function that considers factors influencing processing times.

Lastly, Solak et al. (2009) gives an account of existing research that address similar airport capacity optimisation problems. For instance, Saffarzadeh and Braaksma (2000) describe a planning model similar to Solak et al. (2009) without considering a holistic airport system.

The models previously discussed neglect a number of important performance metrics such as space. Brunetta et al. (1999) address this issue by combining space with the deterministic queue model in an integrated time and space Level Of Service (LOS) performance indicator. This indicator is based on the arrival profile, average dwell time and the spatial area. Brunetta et al. (1999) argue that this indicator is more useful for decision making than previous indicators based solely on time.

3.2. Efficiency

This section reviews several models, all described to be suitable for operational planning and design. Efficiency and capacity models of Section 3.1 share a number of key features such as evaluating the same performance indicators. However, these models focus specifically on day-to-day airport operations and require a higher level of detail. Typically, they require passenger arrival/departure schedules, processing rate of each airport activity and the geometric layout of the airport. Their goal is to simulate passenger flow through the airport terminal and produce key performance indicators (KPIs), which can be used to evaluate the utilization of airport facilities along with resource scheduling. The main techniques used to simulate efficiency models are: Discrete event-event based simulation, discussed in Section 3.2.1, which model processes as queuing/activity networks where state changes in the system occur at discrete points of time. System dynamics models, presented in Section 3.2.2, simulate a system as a series of stocks and flows in which the state changes are continuous. Lastly, agent based modelling, discussed in Section 3.2.3, which simulate complex systems as a collection of agents who generate beliefs that allow them to make decisions and plan activities.

3.2.1. Discrete-Event Based Models

Eilon and Mathewson (1973) studied the effects of airport terminal processes, flight schedules, passenger characteristics (e.g. nationality) and service rates on the processing time and congestion of facilities within the terminal. Their model collects statistics for the aggregate population of passengers, by simulating individual passengers going through their airport activities.

One of the distinguished features of using simulation is the ability to represent results in a graphical context. Simulations can output airport specific performance metrics during the course of the run time, allowing to plot the evolution of KPIs over a specific period of time. For instance, the throughput rate of the security checkpoint for each 15 minute interval. From these plots, architects, consultants and managers are easily able to evaluate airport performance and alternative solutions. However, Eilon and Mathewson (1973) states that it is important to note that these graphs also have their limitations, namely: the inability to identify causal factors influencing airport performance.

To address this issue, Eilon and Mathewson (1973) start to analyze causal relations by looking into the correlation between system performance level and multiple explanatory variables. For their system perfor-

mance KPI, a congestion metric of airport facilities is used, which was found to be linearly correlated with passenger delays and staffing levels. As mentioned previously, multiple parameters were tested, but only two correlations are presented in the study.

In discrete-event simulation models, the order of process activities and their spatial location dictate all passenger flow through airport facilities. Instead of knowing the spatial location of individual passengers, these models designate each passenger to a facilitation area (e.g. check-in area). Based on the number of passengers in a facilitation area and the area's spatial footprint, metrics such as congestion can be calculated.

Furthermore, to better understand the simulation of airport terminal, Eilon and Mathewson (1973) compared simulation models and their corresponding queueing network (see Figure 3.3 for an example). They state that computing requirements have decreased significantly, specifically 17.9 minutes for the simulation and 0.78 minutes for the queuing network, while results have stayed very similar.

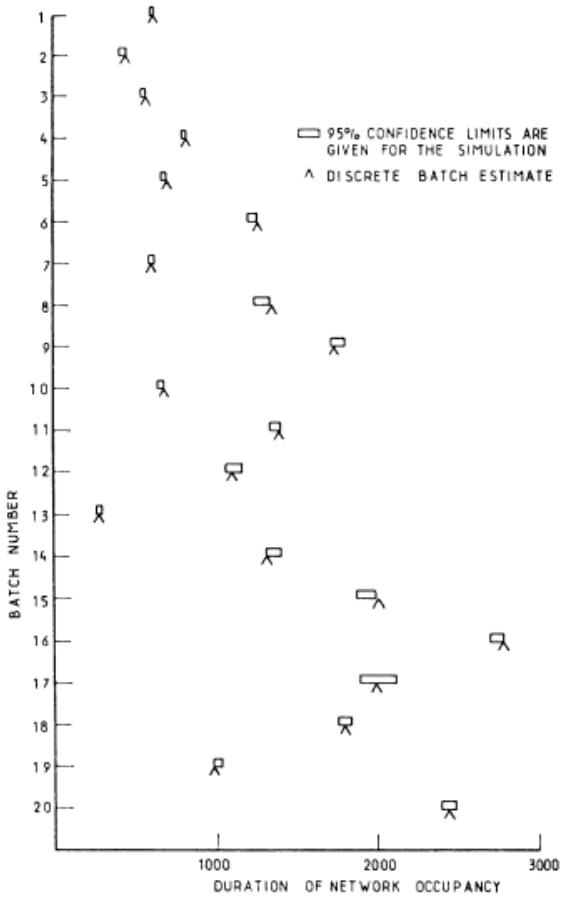


Figure 3.3: Comparison between discrete-event based simulation and queuing networks from Eilon and Mathewson (1973).

There are a large number of published studies that expand upon Eilon and Mathewson (1973) work, as mentioned by Tosic (1992).

For example, Crook (1998) was commissioned by British Airway to visualise passenger flows through the airport system and individual processes, while retaining the ability to evaluate changes to the internal design of the airport terminal. The proposed model helps airport operators improve their knowledge between airport terminal design and passenger service, such as knowing when to open an extra service counter.

The Airport Landside Simulation Model (ALSIM) from McCabe and Gorstein (1982) is a computer simulation program which models the movement of passengers and vehicles across the *landside* of the airport. From literature, the landside of the airport is defined as a region encompassing everything between a passenger entering the airport and boarding onto the plane. Model inputs are airport geometry, facility characteristics (e.g. service distributions), flight schedules (determine arrival/departure profiles) and passenger characteristics. ASLIM outputs a report for each simulated hour and for the total simulation. Hourly reports

include queuing time distribution of each landside facility, from which it is possible to determine the average queue time often used in airport congestion analysis. The total simulation report allows to deep dive into each landside facility, by providing outputs of average queue time, average utilization, occupancy, passenger flow and queue size for each. Overall, ALSIM is a discrete-event, fast-time computer simulation model that is macroscopic, probabilistic and capable of producing passenger flow and airport congestion parameters. These scenarios are similar to those discussed by Braaksma and Cook (1980) and Eilon and Mathewson (1973). These models help airport operators plan landside application such as airport design and master planning, analysis and cost/benefit analysis of landside investments, management of daily operations and landside capacity by answering the following questions:

Airport Modelling Questions

- What are the bottlenecks in the operations?
- Are the resources balanced?
- Is the space plan efficient?
- Is the design capable of handling peak hours?
- What are the wait times?
- Are the facilities well located and sized?

Another practical study of airport terminal is that of Lui et al. (1972), in which they present a discrete-event based model for New York JFK Airport. The inbound-passenger flow was modelled, starting with the arrival of an airplane (with different passenger sizes) at a gate and terminating after passengers collected their luggage and exited the terminal. Their model addresses the questions above and helps planners with short-term expansion plans. According to Lui et al. (1972), the model would be expanded to help with long-term expansion plans and be used at other airports, however no additional information could be found.

For the purposes of planning, it is of paramount importance to determine peak hours/periods, as explained by R. D. Neufville and A. Odoni (2003). Literature has surfaced many definitions over the decades, thus a broad definition is used in this study. Design peak hours and design peak days should not be the hour or day respectively, with the greatest traffic demand because this would oversize the airport. Instead, days should be chosen where their corresponding demand is only exceeded by several cases during the year, thereby ensuring that airport facilities will not be overdesigned, yet still capable of handling capacity requirements. Estimates of total design peak hours are difficult because of the juxtaposition between peak-hour departing passengers and peak-hour arriving passenger. In fact, it is unusual for design peak hours (total, arriving and departure) to occur in the same hour of the day. However, such estimates are necessary for the design of passenger facilities, where many facilities are scaled according to arrivals, departures or sometimes both.

Zografos and Madas (2006) also apply a practical context to the airport modelling questions, by preparing Athens International Airport for the 2004 Olympic Games and its possible role as a southeast Mediterranean hub. Furthermore, senior management of Athens Airport have tasked them with the following modelling questions:

- Will the airport be able accommodate the expected rise in traffic demand?
- Will the airport be able to sustain the expected rise in traffic demand on a permanent basis?
- What are the consequences on airport operations in both cases?

To answer these question, Zografos and Madas (2006) used both a landside and airside model to simulate the temporary traffic increase. Although the models established that the runway system (airside) is the limiting factor for the capacity of Athens Airport, Zografos and Madas (2006) demonstrated this would not affect the operational performance of the airport because it is still able to sustain the demand projections of both scenarios.

Jim and Chang (1998) also examined the flow of arrival and departure passengers using a holistic discrete-event based model. Similarly to Lui et al. (1972) and Eilon and Mathewson (1973), the flow is simulated from facility to facility. Model inputs are the same as for ALSIM (McCabe and Gorstein 1982) with the addition of domestic/international passengers and baggage. Interestingly, there were no differences in model outputs with respect to ALSIM. In a study investigating inbound and outbound capacity bottleneck at

Schiphol Airport, Gatersleben and Weij (1999) present a similar model. Solutions for bottlenecks can be judged by simulating airport performance under new scenarios, which provided insight into present and future situation, allowing for pro-active strategy in preventing future bottlenecks. Likewise, Roanes-Lozano et al. (2004) examined the outbound process Malaga Airport using an analogous model to aid terminal design. Lastly, Joustra and Dijk (2001) and Appelt et al. (2007) both apply a similar methodology for simulating check-in. Furthermore, they consider new check-in scenarious such as online check-in.

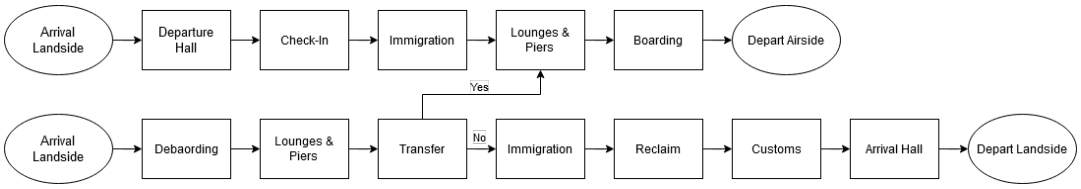


Figure 3.4: Process of passenger handling replicated from Gatersleben and Weij (1999).

Additionally, Jim and Chang (1998) propose that airport organization should adopt simulation cycle for terminal design seen in Figure 3.5. It begins by evaluating the relationship between airport system components, also known as result. These results are then interpreted by airport designers/operators and conclusions can be drawn. From the conclusions, recommendations can be drawn that improve the airport terminal, which in turn can be simulated.

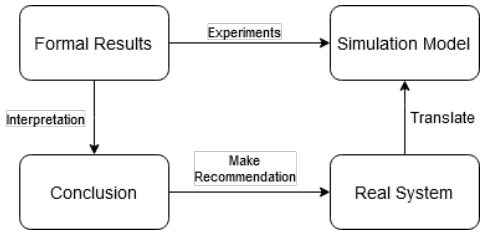


Figure 3.5: Simulation cycle adapted from Jim and Chang (1998).

Takakuwa et al. (2003) present a similar model to Roanes-Lozano et al. (2004) for Kansai International Airport applied to outbound passengers using the Arena modelling package (a discrete-even based simulator). Although most model characteristics are the same, the does include some additional features, namely: number of bags and passenger groups. Furthermore, routes are pre-determined to transfer passengers to the specified airport facility, or the next process, where each facility has a physical location. The most striking result to emerge from the simulation, whose purpose was to analyse facilitation process capacities, was that the key-bottleneck for departing passengers is the check-in lane, which accounted for more than 80percent of the total waiting time of passengers at the airport. Furthermore, it was proven that these waiting times can be reduced by using business class counters to serve economy class passengers as well.

An analogous simulation model to Takakuwa et al. (2003) is demonstrated by Ju et al. (2007), also implementing the Arena modelling package. However, the main focus of the model is the waiting area for large airports. Outputs are similar to those of Takakuwa et al. (2003). Furthermore, this is the first study reported that uses simulation optimization tools on the airport terminals. Ju et al. (2007) uses combinatorial optimization techniques to maximize profit and size the number of facilities for the baggage department and safety inspection department.

Similarly, Fayez et al. (2008) models inbound and outbound processes for passengers using the airport terminal simulation tool called AirSim. Due to the complexity of the airport terminal, an extensive discrete-event simulation model was developed that was both flexible to simulate different scenarios and configurations. The model contains information with respect to enplanement and deplanement of passengers, security screening checkpoints, baggage claim and passenger flow. The following scenarios were considered when determining the airport Level of Service (LOS):

- The current state.
- The new security scenario.
- Opening one more security lane.

- Opening two more security lanes.

Lastly, Fayez et al. (2008) claim that their simulator is able to evaluate trends such as new passenger arrival/departure patterns, airport capacities, new safety requirements, and public transportation components.

3.2.2. System Dynamic Models

To better understand the airport modelling questions and aid with operational concepts, Manataki and Zografos (2009) and Manataki and Zografos (2010) propose a system dynamic model. They argue that complex socio-technical systems such as airports cannot be modelled accurately using macroscopic models because of their stochastic nature. Therefore, their simulation is composed of two hierarchical levels. Level 1 classifies a set of functional areas of the airport terminal system. Level 2 decomposes the functional areas into service facilities (i.e. ticketing) of the airport. It is important to note that each airport service facility is represented by a module. Modules are connected to form airport functional areas (Level 1), which are further combined to create the airport terminal. This can be observed in Figure 3.6.

Accordingly, a mesoscopic approach is defined that operates on an intermediate level of detail while focusing on aggregate characteristics. Early examples of discrete-event based simulation discussed in Section 3.2.1, such as Eilon and Mathewson (1973), are more macroscopic in nature because they do not include detailed passenger interactions.

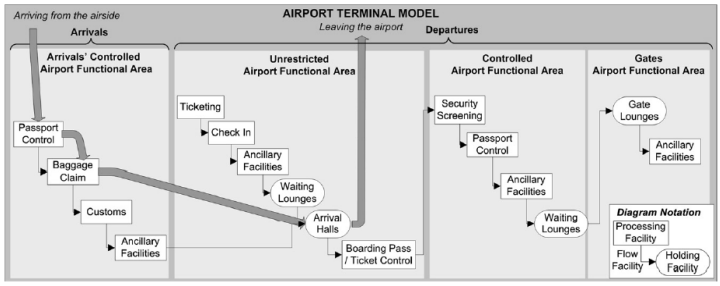


Figure 3.6: Airport terminal structure from Manataki and Zografos (2009).

Recent trends in computational capacity have led to a proliferation of microscopic studies, focusing on passenger interactions. A plethora of these studies use commercially available airport terminal simulators. For example, Kiran et al. (2000) uses the ProModel simulation package to represent the Istanbul Ataturk Airport terminal. The model is able to capture processing time metrics at each facility and resource usage for departing passengers. In addition, it is possible to also use the model for training and demonstration of terminal activities.

Virtually all of the above described models can also be used for visualisation purposes. However, two models specifically address visualisation of airport terminals. Crook (1998) (reviewed above) visualizes cross-flows of passengers along with processes to help in evaluating various terminal design scenarios as requested by British Airways. Similarly, Koch (2004) applies a discrete-event based model for visualising airport security operations, enabling designers to explore new technologies and procedures.

3.2.3. Agent-Based Models

Wilson et al. (2006) are the first to present a simplistic agent based model for an airport terminal, focusing specifically on the security checkpoint. Agents are generated into the simulation but still follow a fixed activity-oriented schedule. Routes for agents are determined by traversable elements in the simulation. They generate instructions for an entity to follow through their respective area. Since it is possible for an agent to have many valid routes through a region (a collection of multiple traversable elements), the agent must decide which route to take. Furthermore, a region ensures that agents do not collide and obey constraints such as spacing and non-accessible areas. This study has led to development of a simulation tool, known as Security Checkpoint Optimizer (SCO), for the Transportation Security Administration (TSA) to support planning and analysis activities.

The previously reviewed models in Section 3.2.1 and Section 3.2.2 address the need for using multiple performance metrics to assess airport facilities. Wilson et al. (2006) goes a step further to evaluate each of the following separately: operational cost, passenger and bag throughput, resource utilisation and security

effectiveness. The additional capability of security effectiveness refers to the probability of detecting a prohibited item, which is determined by multiplying the probabilities of detection through the different stages of the passenger facilitation process, similar to Chawdhry (2009). Like many others, this model is able to evaluate different scenarios and security configurations ($WHAT-IF$) scenarios. For example, facilities can be analyzed on passenger flow and time to process. A study with similar modelling objectives is described by Pendergraft et al. (2004), however no implementation details are provided.

Schultz and Fricke (2011) describe the first agent-based model that is associated with artificial intelligence. The model simulates departing passengers by analyzing passenger flow. Each agent has an operational, tactical and emergency planning level that dictate its decision making. Passenger walking speed is dependent on density and group behaviour. For example, passengers walking in dense crowds move more slowly and passengers at the front of a group waits for the slowest member to catch up. In addition, an agent is able to determine its own routes based on sign visibility. Navigation signs are placed at various map components which indicate direction of facilities. Agents adopt a Sense-Plan-Act approach to route finding, which evaluate the utility level of itself. The navigation model is based on both static navigation networks and free orientation as shown in Figure 3.7. In addition, Schultz and Fricke (2011) present a model that is able to evaluate aircraft boarding policy. Single door boarding versus two door boarding, random boarding versus block boarding (i.e. passengers board in groups based on their corresponding seat) are compared based on their boarding times.

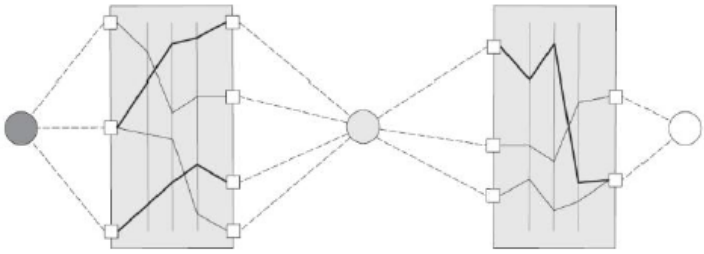


Figure 3.7: Agent (left) uses network (gray box) to navigate heading to an intermediate goal (e.g. sign or information point, center), orientates freely to gather information and use the network to finally reaches the nearest navigation point regarding to the destination (right) taken from Schultz and Fricke (2011).

3.3. Security

Security models simulate airport procedures, technologies and the operators ability to filter out threats from reaching an aircraft (on an airport terminal level). While most capacity or efficiency models follow similar modelling techniques, security models consist of a broad range of techniques, where the main focus is on simulating the probabilistic risk of a terrorism attack.

Since 9/11, airport security has been attracting a lot of interest from researchers and a number of models have been proposed. Existing models simulating airport security focus on the ability or inability of the "combined procedures and technologies in the process to successfully filter out all threats (harmful substances/prohibited items and malicious persons) from reaching an aircraft through the passenger facilitation process" Chawdhry (2009). The majority of the models reviewed in this section are intended to help policy makers evaluate existing and new security regulations. Section 3.3.1 present security definitions that are used in this section. Section 3.3.2 describes the Threat Vulnerability Consequence (TVC) method and how it is applied to security checkpoints. Section 3.3.3 introduce attack trees to calculate the probabilistic risk of an incident. The following Section 3.3.4, Section 3.3.5 and Section 3.3.6 discuss security models from literature using probablistic method, fuzzy logic and game theory, respectively.

3.3.1. Security Definitions

The literature on security has generated various definitions related to security risk and its components. Hence, it is important to formalize the definitions used within this paper. For the purpose of this literature study, Brashear and Jones (2008) definitions are used for risk, threat, vulnerability, consequence and resilience.

Def. 01 *Security Risk*

The potential for loss or harm due to the likelihood of an unwanted event and its adverse consequences.

Def. 02 *Threat*
Any indication, circumstance or event with the potential to cause the loss of, or damage to, an asset or population.

Def. 03 *Vulnerability*
Any weakness in an asset or infrastructures design, implementation or operation that can be exploited by an adversary or contribute to functional failure in a natural disaster.

Def. 04 *Consequence*
The outcome of an event occurrence, including immediate, short and long-term, direct and indirect losses and effects.

Def. 05 *Resilience*
Resilience is broadly defined as the ability to function through an attack or natural event or the speed by which an asset can return to virtually full function (or a substitute function or asset provided).

3.3.2. TVC Method

In TVC methodology, security risk consists of three components: threat, vulnerability and consequence. To evaluate these components, security experts determine the assets in their organization. Once assets have been specified, a set of possible threats can be identified. Finally, risk is determined by estimating the threat, the vulnerability to the threat and the consequence of the threat, see Willis et al. (2006).

$$\begin{aligned} Risk &= P(\text{attack occurs}) \\ &\quad \cdot P(\text{attack results in damage} \mid \text{attack occurs}) \\ &\quad \cdot E(\text{damage} \mid \text{attack occurs and results in damage}) \qquad (3.2) \\ &= Threat \cdot Vulnerability \cdot Consequence \qquad (3.3) \end{aligned}$$

To assess the threat likelihood, estimations are often based on intelligence data, a cost benefit analysis or historical data, such as the *Global Terrorism Database*. However, it is important to note that due to the random nature of terrorist attacks, historic data is not indication of future events.

For example, safety experts use data generated by security sensor vendors, internal assessments and employee surveys to estimate vulnerability. In addition, tools like event trees (see Section 3.3.3) can be used to better estimate vulnerability. Lastly, experts also make use of Red-teaming (real-life simulation of a threat scenario) to assess vulnerability.

The consequence of a threat can be evaluated using consequence assessment techniques, which are frequently expressed in financial costs. As an example, "Value of a Single Life" (VSL) is often used to quantify the loss of a human life. Expert judgement are usually used to estimated the consequences of a threat. Finally, the current situation is compared with the expected reduced security risks for potential controls in order to risk mitigate.

3.3.3. Attack Trees

A methodical way of modelling the security of systems under varying threat scenarios is using attack trees. The system is represented in a tree structure, where the root node (or top-event) reflects a successful attack within the system on a specific asset. Leaf nodes represent independent events that can occur, while internal nodes represent events that depend on their corresponding child nodes. Each node is assigned with a value that reflect their probability of occurring, their execution cost and other parameters. An expert determines the value of leaf nodes, while the value of other nodes is determined on the basis of their respective child nodes. It is possible to model transitions between nodes as either deterministic or non-deterministic. In deterministic transitions, a parent node occurs from a child node or combination of child nodes, while the opposite is true for non-deterministic transitions. Measures to prohibit an attack can be analyzed by evaluating the value of the root node of the tree.

3.3.4. Probabilistic Methods

Chawdhry (2009) A significant analysis and discussion on the subject airport security was presented by Chawdhry (2009), where the likelihood of a prohibited item or malicious person reaching an aircraft is

modelled for the departure process. In literature, this is commonly known as *missed detection*. Using this model, security policies and deployment of new security technologies can be can be quantitatively assessed. Assuming that individual processes in the security are independent, the probability of permeability can be multiplied for each process in order to determine the the probability of a missed detection. The probability of permeability for individual process is determined by security experts. Chawdhry (2009) proposes a scenario where different levels of screening are performed on passengers according to a passenger risk evaluation scheme. Hence, passengers receive different types of security screens (with different probabilities of permeability) based on their attributes in order to reduce the probability of a missed detection. It is important to note that the model proposed is only for security and does not consider the effect of additional screening on passenger flow.

The concept from Chawdhry (2009) is expanded by Babu et al. (2006), who uses a bayesian probabilistic model. The previous assumption of independence is overcome by modelling a missed detection using conditional probabilities. Furthermore, Babu et al. (2006) keeps track of false positives (i.e. detecting a threat when it does not exist) and false negatives (i.e. not detecting an actual threat). Similarly to Chawdhry (2009), this model divides passengers into different risk groups and performs different security screens. However, it goes further in also optimizing the amount of risk groups there should be. Using this model, the impact of security policies can be assessed and it can recommend an optimal implementation of a policy.

Nie et al. (2009) extends the model by Babu et al. (2006) by not just optimizing the risk group but also taking into consideration the number of screen operators at a security checkpoint. The main conclusion is that the screening process can be more effective when considering the passenger attributes. This reduced the the number of false positives while keeping the number of false negatives the same.

3.3.5. Fuzzy Model

Akgun et al. (2010) present a fuzzy model that incorporates experts opinion and multiple criteria to assess the security risk of an airport. The criteria are as follows: deterrence, detection, delay, response and recovery. Previous models analyzed have focused only on the detection criteria (i.e. probabilistic risk). Experts rate each airport function based on the five criteria using fuzzy linguistic variables. Furthermore, dependencies between pairs of functions, pairs of components and dependencies of function to airport mission are rated. It is possible to discover hidden vulnerabilities that arise due to interdependencies by simulating the airport with a fuzzy cognitive map.

The model created by Akgun et al. (2010) emphasizes the importance of detecting dependencies due to their influence on vulnerability and risk. Not only is their framework useful for airport operators to aid in allocation of resources and policy making, but it also targets regulatory authorities and their policy making. Furthermore, it can be used to identify airport facilities that must be improved regarding security. It is important to note that this model does not consider spatial layout of the airport nor passenger flow.

3.3.6. Game Theory

Frameworks based on game theory, such as Brown et al. (2016), construct a security game to represent a threat scenario. A table is created where the rows and columns symbolize the defender and attacker of the game respectively. Each column is a potential target for the attacker, whereas each row is a possible action to deter the target. The outcome of the game equals a combination of vulnerabilities and consequences. By finding the equilibrium of such a game, a defender can obtain their optimal strategy (i.e. preventing threats). Security games are able to simulate a wide range of airport applications such as airport security patrols by Pita et al. (2009) to detection of prohibited items.

3.4. Limitations and Conclusions of Simulation Models

Overall, these studies highlight the need for simulation to improve operational planning of airports. The standard methodology for modelling airport efficiency is as follows: Identify the sequence which activities are completed or facilities or visited in the passenger facilitation process; Obtain arrival/departure profiles through prediction or historical data for peak flight schedules; Populate model parameters using collected data; Verify the model by comparing simulated performance versus actual performance for predetermined scenarios; Simulate new scenarios and output key performance indicators of interest.

While airport terminal simulation has been done for several airports individually, there are no unified approaches nor standardized test systems available. The approaches are fragmented and based on individual airport needs and preferences. This has led to development of several airside and landside models, making

comparison between them very difficult. Recently, a considerable literature has emerged around the theme of causal factors that influence system performance. However, some argue that system performance is dependent on the stochastic nature of an agent leading to emergent behaviour.

While much research has been carried out on the full airport terminal system, few studies exist on detailed simulation of airport activities. The check-in has been the most widely studied airport activity focusing on policies to increase passenger throughput. Surprisingly, so far very little attention has been paid to the role of security checkpoint when considering the efficiency of the airport. This can also be seen in Figure 3.8 where there are no overlapping studies between security and efficiency. Furthermore, it is apparent from the figure that there are no overlapping studies at all.

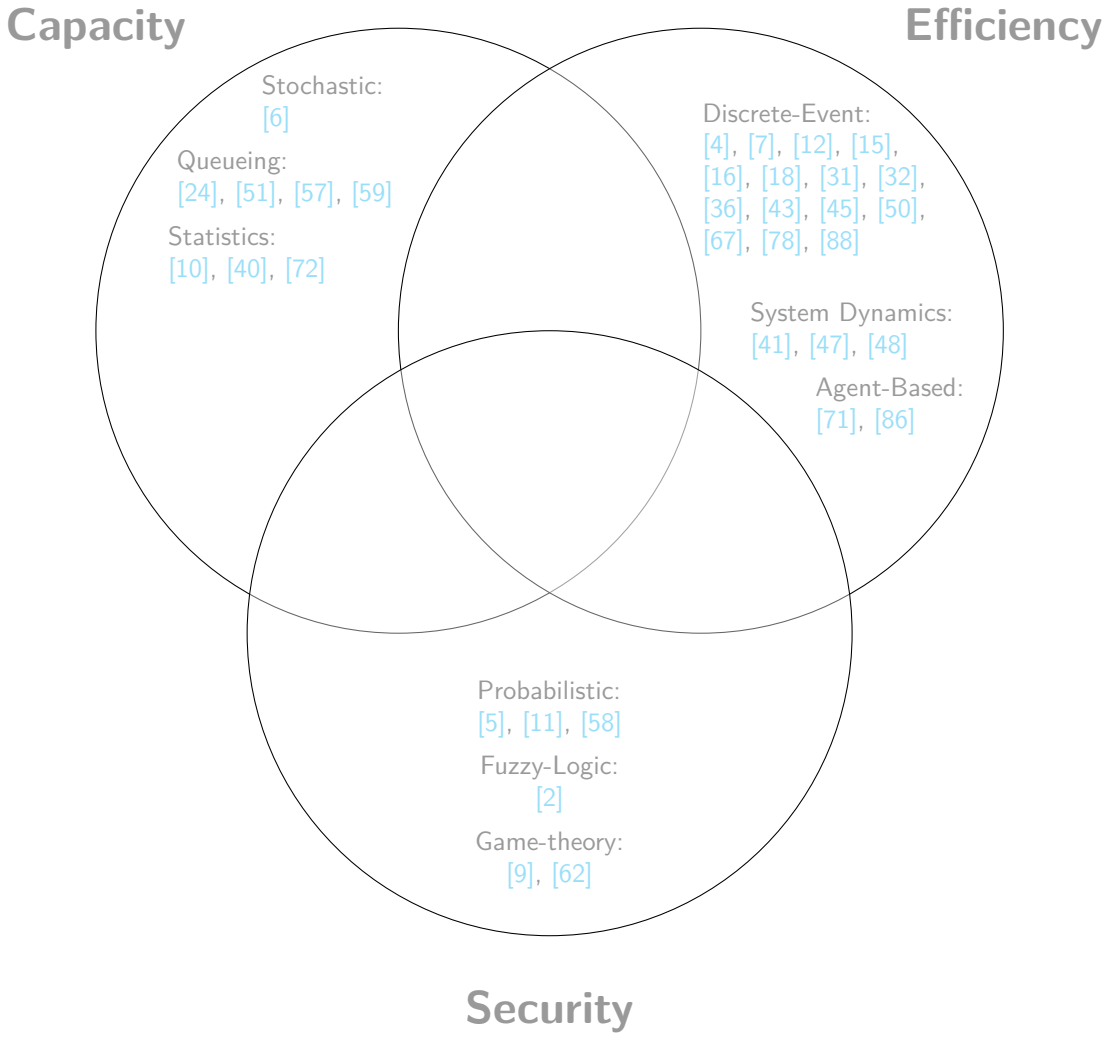


Figure 3.8: Summary of simulation model literature

In light of recent advancements in computational power, it is becoming extremely difficult to ignore the existence of small-scale interaction. Therefore, the past decade has seen the rapid development of more detailed models using discrete-event based simulation. By contrast, a search of the literature revealed few studies which use an agent-based modelling or multi-agent system approach to model airport terminals. A possible explanation for this are the limited number of agent-based simulation software that exist.

Furthermore, computational power has also led to a rise simulation optimization of airport terminal processes. Although few studies exist that use optimization techniques on airport terminal simulations, no studies have been found that use reinforcement learning to learn optimal policies for airport activities.

Two important themes emerge from the studies discussed so far in Section 3.3: airport security is of paramount importance to homeland security and modelling airport security typically requires the consul-

tation of security experts due to the limited number of threat incidents. Although there are various national and international security regulations, their enforcement varies greatly across airports. Organizations exist that keep track of which security equipment meet the security requirements by assessing their ability to meet certain functions. In order to determine the overall security effectiveness, quantitative measures are required, which typically focus on the probability of a missed detection in literature. This requires experts to elicit estimates of probabilities and conditional probabilities of various equipment and activities in the process.

Due to the limited number of terrorist incidents at airports, data about the efficiency and safety of airport security is limited. Thus, making it very challenging to model. As mentioned previously, probabilistic risk is typically evaluated by an security expert. Performance of security equipment has been measured by probabilities and up to now, far too little attention has been paid to their function. Furthermore, in contrast to probabilistic risk at security checkpoints, there is much less information about effects of airport operator fatigue modelling.

Overall, a number of limitations and opportunities for further improvements and future modelling research have been identified: the lack of well standardized key performance metrics of airport effectiveness addressing the performance of the airport airside and landside; the lack of compatibility between input requirements such that various scenarios can be tested interchangeably; the lack of modelling airport activities in detail; and the absence of new modelling techniques (i.e. simulation optimization and reinforcement learning) to help design new airport terminals.

Chapter 4

Simulation Software

One of the main obstacles in airport terminal planning is being able to test new layouts/concepts and their effects on performance. Simulation has emerged as a powerful platform to help plan, design and manage airport terminals. As mentioned previously, initial research focused on creating simulation models that were able capture operational bottlenecks. The airport industry realized the usefulness of such tools and simulation software were developed from these initial models. As time progressed, more of academia presented literature could be found in simulation software because of the convenice, realibility and efficiency in analysis. Section 4.1 and section 4.2 will present simulation platforms for non-Airport terminals and airport terminals respectively.

4.1. Non-Airport Terminal Simulators

Agent-Based Simulators

A plethora of simulations exist in literature, but there are few that allow for an intuitive implementation of airport terminal elements and human behaviour. By definition, agent-based modelling is a natural approach to modelling human behaviour. While there are several agent-based simulation software, such as Netlogo (Tisue and Wilensky 2004) and Gama (Taillandier et al. 2010), that allow specification of human models, they do not contain airport terminal specific elements, such as check-in desks, or passenger interaction with these elements. Implementing these airport specific elements in an existing simulation will be difficult as there are many interdependencies that must be considered. Furthermore, elements should resemble plug-and-play functions which may not be possible in existing literature. One important aspect of agent-based simulators is that they contain various tools to analyze agent behavior and simulation outcomes. Therefore any attempt to replicate agent-based simulators should have similar analysing capabilities.

Pedestrian-Oriented Simulators

Another possibility to simulate passengers, as opposed to agent-based simulators, is to use pedestrian-oriented simulators. These simulators specialize in modelling walking behaviour and contain several algorithms to specify passenger behaviour. Similarly to agent-based simulators, these simulators do not contain any airport specific elements. Furthermore, implementing these type of elements will face similar problems as already discussed in the previous section.

4.2. Airport Terminal Simulators

Simulators need to tradeoff simplicity, speed and accuracy for the consumer. For example, accuracy with smale-scale interaction comes at the price of complexity, speed, and computational cost. For the purpose of this section, different levels of functionatility are introduced such that it is possible to tradeoff simulators. These levels are dependent on what the simulation considers as an input, sophistication of the simulation, detail of the real-wrld system and precision of the output. The levels are as follows:

Level. 01 A system is simulated by one or more events, where changes can occur at event times according to pre-specified logic. Furthermore, these events are able to alter the environment of the system. These types are usually event-oriented simulators and are used for peak-hour analysis.
Considers: arrival/departure profiles to events.

Level. 02 These types are usually process-oriented models which model the flow of passengers through the system based on the order of defined events. It is important to note that delays not not only incurred when demand rate exceed service rate as in queueing theory.
Considers: **Level. 01** and time variation of the system

Level. 03 These types are similar to those of **Level. 02** except that they also include the stochastic nature and probabilistic aspects of the system services.
Considers: **Level. 02** and stochastic and probabilistic nature of the system

Level. 04 These simulators are usually object-oriented models where individual agents are modelled moving through the system. Agents are represetned by objects with distinct attributes that take shape in the form of values and properties. It is important to note that these passengers can have beliefs (see section 3.2.3) but must not have them. Attribute specification of agents is limited. **Considers**: **Level. 03** and passenger interaction and limited passenger properties

Level. 05 This level is similar to that of **Level. 04** except that attributes of passengers are extensive. **Considers**: **Level. 04** and extensive passenger properties.

4.2.1. Academia Developed Simulators

In the 1960's, researchers at universities pioneered the scenes of airport terminal congestion by analyzing the problem using simulation approaches. The first workshops discussing the problems facing air transport operations was held at the Massachusetss Institute of Technology (MIT), which led to the first analytical queuing models from A.R. Odoni and R. d. Neufville (1992) and Horonjeff et al. (2010). Although their work did not culminate into a simulator for the research community, their work would be considered a **Level. 01** model.

W.J Dunlay Jr et al. (1975), whose researched was federally sponsored, simulated terminal operations and performed capacity analysis of the airport. Main research focus was on modelling airport access (W.J. Dunlay Jr and Park 1978) and the flow of passengers through the airport (W.J. Dunlay Jr and Park 1978). Their work resulted in the development of a simulation model *ACAP*, which is considered a **Level. 02** model. ACAP is a FORTRAN-based model and is comprised of a single main program. Furthermore, it simulates individual facilities through the use of component modules. These modules are regression models that are developed from collected data taken at specific airports. It is important to note that these models are only able to replicate operational conditions when the survey data holds because their predictions are entirely based on the data collected. In order to overcome this shortcoming, the modules were modified to use monte carlo methods to predict capacity of individual facilities. These methods sample process times from their corresponding distribution allowing much better representation of operational conditions (Mumayiz 1985). The modifications transformed ACAP from a **Level. 02** into a **Level. 03** model. Despite these improvements, airports did not use ACAP in practical applications.

Not only were airport simulation studies done in the United States, but also in the United Kingdom, Netherlands, Japan, Turkey, Greece and more. However, most of these studies did not result into general airport simulators.

In scotland, a terminal simulation model (AIR-Q) was developed at the University of Strathclyde (Laing 1975). The main purpose of the model is to analyze the efficiciency of different airport terminal configurations, therby aiding in the comparison between designs. It is comprised of all possible activities in the terminal building, where each activity is represented by a queuing model. Furthermore, each activity node may contains its own queue where its corresponding size is determined by the service time and number of services available. The simulation outputs the queue sizes of all activites over the simulated time for a unique throughput of passengers. It is important to node that the model does not consider stochastic variations in service time and is thus determinstic. This modelis consiered to be a **Level. 02**.

At the Delft Univeristy of Technology, a terminal simulation model AATOM was developed Janssen et al. (2019). The model is an agent-based terminal simulator is open-source and programmed in java. Terminal services are represented as physical objects/areas that an agent can is able to interact within. Human agents are central in complex socio-technical systems such as aiport terminals. Hence, a specific architecture is used to mimic human decision making and behaviour. Each human agents has an architecture as presented in Figure 4.1, which consists of the strategic layer, the tactical layer and the operational layer.
The strategic layer, or top layer, of the agent represents the reasoning a human performs about their own beliefs and goals. It is therefore composed by a belief module, goal module and reasoning module. Additionally, the reasoning module consists of analysing and planning of goals/beliefs and decision making. The tactical layer, or middle layer, resembles the actuation of a human to perform activities based on its strategy. Additional responsibilities of the tactical layer are interpretation of observations and navigation in the environment. Lastly, the operational layer, or bottom layer, mimics the senses and stimulus for human to perform actions. The agent is able to interact with the environment by receiving observations as an input and can alsoe execute actions. The stimulus for which action to perform comes from the tactial layer.

According to Janssen et al. (2019), the main sequence of oeprations if followed by the AATOM architecture: "observation → perception → interpretation → reasoning → activity control → actuation → action."

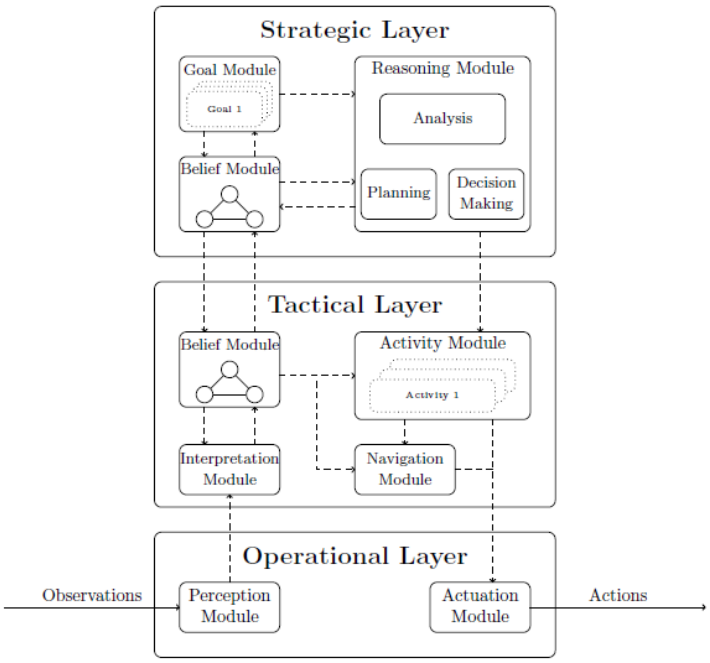


Figure 4.1: AATOM Architecture taken from Janssen et al. (2019)

This simulator is a **Level. 04** type simulator but through user adjustments can be turned into a **Level. 05** type simulator.

4.2.2. Industry-Based Simulators

Another source of airport simulators is industry. Often times, consulting firms are contracted to create airport simulation models that aid airports in designing and managing their terminals. However, little has been published on these simulators due to confidentiality.

Metais (1974) created a time-oriented queuing model of type **Level. 02**, which has the capability of simulating airport terminals containing 50 gates. The Federal Aviation Administration (FAA) bought the software and expanded the model. This new software is known as ALSIM (McCabe and Gorstein 1982) and is described later in section 4.2.3.

Finally, many airport terminal simulators software exist which are in use by airports today. For instance, the Pedestrian Dynamics simulation software from INCONTROL Airport Simulation Solutions. It is possible to simulate up to 100,000 individuals. Quick and easy modeling allow for realistic crowd movement and behavior with unique passenger properties. The output is adjustable for passenger processes and utilization of infrastructure. Lastly, the simulation can be visualized in a 3D environment. It is not possible to simulate the baggage/luggage moving through the airport. This is a **Level. 04** type simulator.

The Pax2Sim simulator from HUB Performance is a **Level. 04** type simulator. It has the ability to optimize cargo systems, baggage handling systems and passenger terminal operations. Furthermore, the simulator has previous experience with a lot of airports such as Paris CDG, London Heathrow, Aeroporti di Roma, etc. The Pax2Sim can be used to size airports or to produce trends on passenger information, which is why it is used to airport planning and infrastructure projects. Additional services that they offer are flow capacity planning and peak flow analysis based on a given flight plan. This analysis produces several forecast such as the waiting time in queue.

The CAST simulator from Airport Research Center is another example of a **Level. 04** commercial simulator. They have proven very useful in capacity planning for airports, and provide valuable insights to airport managers.

4.2.3. Government-Based Simulators

Previous mentioned simulation models such as ACAP (W.J. Dunlay Jr 1978) and the model leading up to ALSIM were sponsored the FAA. The latter was acquired by the FAA and then upgraded by the U.S.

Department of Transportation Systems Center (McCabe and Gorstein 1982). ALSIM is a discrete-event simulation based on probabilities focusing on macroscopic interactions. It is capable of producing key performance indicators on simulated facilities and can capture congestion/flow parameters. It is a **Level. 03** type simulator.

Another model developed by government-based institution is the Calgary Model from the Canadian Air Transportation Administration. It is a general purpose simulation system that uses time oriented queuing models. Hamzawi (1986) of Transport Canada further develops the models by incorporating three modules for passenger and baggage flow, gate assignment and ground transportation. These modules utilize queuing models with Monte Carlo sampling techniques and are thus a **Level. 03** type simulator.

In the passenger flow module, flow is simulated along pre-defined routes through the airport terminal using an event-oriented stochastic model. In the gate assignment module, the user is able to specify a strategy according to which gate are assigned. Usually, the flight schedule order determines how the flights are assigned. Lastly, in the ground transportation module, the flow of vehicles is simulated along the road infrastructure of the airport.

Furthermore, it is important to note that the British Airport Authority developed an airport terminal simulation software that is capable analyzing individual facilities. However, there are no publications on the model and hence little is known about its general properties and features.

4.3. Limitations and Conclusion for Simulation Software

As discussed previously, there are three main sectors that develop simulators: academia, industry and government. All of these simulators have different levels of detail and an overview can be found in table 4.1. An surprising revelation from literature is that there are no **Level. 05** simulators at the moment. This is because simulators do not capture extensive passenger details/attributes. It is important to note that some of the **Level. 04** simulators (such as Janssen et al. (2019)) can be transformed into **Level. 05** simulators by making adjustments. Another interesting fact is that Janssen et al. (2019) is the only academia developed simulator of **Level. 04**, while all others are based on industry.

Table 4.1: Overview of the level of detail for each simulator.

Level 1	A.R. Odoni and R. d. Neufville (1992) Horonjeff et al. (2010)
Level 2	W.J Dunlay Jr et al. (1975) Laing (1975) Metais (1974)
Level 3	Mumayiz (1985) McCabe and Gorstein (1982) Hamzawi (1986)
Level 4	Janssen et al. (2019) McCabe and Gorstein (1982) INCONTROL Airport Simulation Solutions HUB Performance Airport Research Center
Level 5	

The main disadvantage of most simulators is that they are commercial, and thus not open source. This makes it very difficult to try out new modelling techniques because the source code cannot be adjusted. Hence, the user is always constrained to the framework and input parameters that are given by the simulation software.

Another disadvantage is that there is only one simulators that is agent-based (Janssen et al. 2019). Therefore, characterizing human behavior and complex interactions between cognitive agents cannot be modeled well. This has implications on any process where passenger behaviour differs based on passenger types, such as at security where businessman/businesswoman would divest their luggage faster compared to elderly passengers.

In conclusion, it has been decided to use Janssen et al. (2019) simulation software for the following reasons:

- AATOM has an agent architecture that allows to specify agent behaviour at airports. It is possible to specify the beliefs and goals of a passenger along with reasoning behind goals which get translated to activities/actions that an agent performs.
- AATOM is an open source simulator. Therefore, the source code can be adjusted in order to update modules of the simulator if necessary.
- AATOM has been used in several projects at Delft University of Technology. It is favorable to continue using it such that previous work can leveraged upon and perhaps incorporated into new research/experiments.

Chapter 5

Simulation Optimization

Simulation optimization (SO) refers to the optimization of an objective function subject to constraints as shown in Equation (5.1), which can be evaluated through a stochastic simulation. In literature, several competing algorithms exist to address specific features in a particular simulation such as single or multiple outputs, discrete or continuous decision variables, short or long computation time.

$$\begin{aligned} min \quad & E_\omega[f(x,y,\omega)] \\ s.t. \quad & E_\omega[g(x,y,\omega)] \leq 0 \\ & h(x,y) \leq 0 \\ & x_l \leq x \leq x_u \\ where \quad & x \in R_n, y \in D_m \end{aligned} \tag{5.1}$$

Section 5.1 presents single state methods, which focus optimizing a single candidate solution to reach an optimum. On the other hand, Section 5.2 keeps a set of candidate solutions also known as population methods. Lastly, Section 5.3 describe algorithms that optimize multiple objectives at the same time.

5.1. Single State Methods

In mathematics, gradient based optimization is a powerful tool where a candidate solution is optimized by computing the first derivative from a well-understood mathematical function and tweaking the solution in the direction of that derivative. However, in most cases, especially simulations, the function that is being optimized is unknown and therefore the gradient cannot be computed. Simulations are typically a black box where a candidate solution is given as input and the output is the assessed quality/fitness of that candidate solution. It is important to note that it is unknown how the assessment function behaves, which is why the simulation is used to compute an output. A candidate solution can be a range of numbers or structures as it is dictated by the simulation.

To optimize a candidate solution in this scenario, the first step to accomplish is the initialization procedure, which is to provide one or more candidate solutions. Next, the assessment procedure is performed, by running the simulation and assessing the fitness of the candidate solution. Then, the modification procedure is done which consists of two steps: make a copy of the candidate solution; and tweak the candidate solution, to produce a slightly modified candidate solution. Finally, a selection procedure is performed which decides which solution is kept and discarded. By selecting a new candidate solution, the algorithm explores the state space to the problem to find an optimum.

This section introduced three algorithms: hill climbing in Section 5.1.1, simulated annealing in Section 5.1.2 and tabu search in Section 5.1.3.

5.1.1. Hill Climbing

A simple technique of a single-state method is Hill-Climbing, an algorithm related to gradient ascent. However, it does not require the strength of the gradient, or even its direction to be known. This is a result of new candidate solutions being iteratively tested in the region of the current candidate, and new ones adopted if they are better. This enables the hill to be climbed until a local optimum is reached.

5.1.2. Simulated Annealing

Simulated Annealing was developed by Kirkpatrick et al. (1983) in the mid 1980s. In contrast to Hill-Climbing, this algorithm varies in its decision of when to replace the original candidate solution, S, with its newly $tweaked$ child, R. Specifically, if R is better than S, S will be replaced with R; but if R is worse than S, it may still replace S with a certain probability $P(t, R, S)$:

Algorithm 1 Hill Climbing from Luke (2013)

init S ▷ Initial Candidate Solution
repeat
  $R \leftarrow Tweak(Copy(S))$
  if $Quality(R) > Quality(S)$ **then**
    $S \leftarrow R$
  end if
until S is the ideal solution OR we have run out of time
return S

$$P(t, R, S) = e^{\frac{Quality(R) - Quality(S)}{t}} \tag{5.2}$$

where $t \geq 0$. Hence, the algorithm is able to descend hills when exploring the state space. Equation (5.2) is interesting in two ways.

First, if the quality of R is very close to the quality of S, then the probability is close to 1, while if the quality of R is much worse than the quality of S, then the probability is close to 0 because the fraction is larger. Thus, if the fitness of R is slightly worse than that of S, there is a reasonable probability that it will be selected. However, if the fitness of R is much worse, then it is very unlikely for it to replace the candidate solution. Second, a parameter t exists that can be used to tune the probability over time. If t is close to 0, the fraction is again a large number, and so the probability is close to 0, whilst the probability is close to 1 if t is high. The idea in Simulated Annealing is to initially set a high value for t, causing the algorithm explore the candidate solutions regardless of their fitness. Then, as time progresses, t should be decreased slowly, until it reaches 0, at which point a Hill-Climb is being performed.

Algorithm 2 Simulated Annealing from Luke (2013)

init t, γ, S ▷ Temperature, Cooling Rate, Initial Candidate Solution
$S^* \leftarrow S$
repeat
  $R \leftarrow Tweak(Copy(S))$
  if $Quality(R) > Quality(S)$ or if a random number chosen from 0 to $1 < P(t, R, S)$ **then**
    $S \leftarrow R$
  end if
  $t \leftarrow t - \gamma$
  if $Quality(S) > Quality(S^*)$ **then**
    $S^* \leftarrow S$
  end if
until $S*$ is not optimal, run out of time, or $t \leq 0$
return S^*

5.1.3. Tabu Search

Tabu Search, by Glover and Laguna (1998), keeps a history of recently considered candidate solutions known as the tabu list. The algorithm is not able to return to one of the candidate solutions in the tabu list until they are far enough in the past to have dropped out of the list. Hence, when a hill is being climbed, the solution is not permitted to stay or return to the local maximum, thereby resulting in the solution moving down the hill again.

An intuitive approach to Tabu Search is to maintain a Tabu List L, with some maximum size l, of candidate solutions explored so far. A candidate solution is added to the tabu list each time a new solution is explored. Once the list reaches the maximum number of solutions, the oldest candidate solution is removed from L and it becomes possible to consider the solution again. In algorithm 3, n tweaked children are generated but only the ones that are not presently taboo are considered.

Algorithm 3 Tabu Search from Luke (2013)

init l, n, S ▷ Maximum Tabu List, Number of Tweaks, Initial Candidate Solution
$S^* \leftarrow S$
$L \leftarrow$ ▷ A tabu list of maximum length l
Enqueue S into L
repeat
  if $Length(L) > l$ **then**
    Remove oldest element from L
  end if
  $R \leftarrow Tweak(Copy(S))$
  for n-1 times **do**
    $W \leftarrow Tweak(Copy(S))$
    if $W \notin L$ AND $(Quality(W) > Quality(R)$ OR $R \in L)$ **then**
      $R \leftarrow W$
    end if
    if $R \notin L$ **then**
      $S \leftarrow R$
      Enqueue R into L
    end if
    if $Quality(S) > Quality(Best)$ **then**
      $S^* \leftarrow S$
    end if
  end for
until S^* is the ideal solution or we have run out of time
return S^*

5.2. Population Methods

The term *Population Methods* will be used to describe algorithms that keep a set of candidate solutions as opposed to a single candidate solution. While a variety of algorithms exist, it may not be surprising that they take concepts from biology. The phrase *Evolutionary Computation* (EC) encompasses a popular set of techniques that are based on biology, genetics and evolution. An algorithm chosen from this collection is known as an Evolutionary Algorithm (EA). The basic terminology of EA is described in section 5.2.1. Common EAs include Evolution Strategies (ES) discussed in section 5.2.2 and the Genetic Algorithm (GA) discussed in section 5.2.3. Another evolutionary algorithm is Particle Swarm Optimization (PSO) discussed in section 5.2.4. However, this type of alogirthm is modelled after the swarming and flocking behaviours of animals instead of being modelled after evolution.

5.2.1. Evolutionary Algorithm Terminology

Since EAs are inspired by biology, they have adopted terms from genetics and evolution to describe their algorithms. Due to their relevance, this section discusses the meaning of each term as they will be used in upcoming subsection to explain algorithms. Definitions are taken from Luke (2013).

Def. 06 Individual
A candidate solution.

Def. 07 Child and Parent
A child is the tweaked copy of a candidate solution (its parent).

Def. 08 Population
Set of candidate solutions.

Def. 09 Fitness
Quality.

Def. 10 Fitness Landscape
Quality function.

Def. 11 Fitness Assessment
Computing the fitness of an individual.

Def. 12 Selection
Picking individuals based on their fitness.

Def. 13 Mutation
Plain Tweaking. This is often thought as asexual breeding.

Def. 14 Recombination or crossover
A special Tweak which takes two parents, swaps sections of them, and (usually) produces two children. This is often thought as sexual breeding.

Def. 15 Breeding
Producing one or more children from a population of parents through an iterated process of selection and Tweaking (typically mutation or recombination).

Def. 16 Genotype or genome
An individuals data structure, as used during breeding.

Def. 17 Chromosome
A genotype in the form of a fixed-length vector.

Def. 18 Gene
A particular slot position in a chromosome.

Def. 19 Allele
A particular setting of a gene.

Def. 20 Phenotype
How the individual operates during fitness assessment.

Def. 21 Generation
One cycle of fitness assessment, breeding, and population reassembly; or the population produced each such cycle

Overall, an EA follows the standard generational EC algorithm approach. First, an initial population is constructed as a set of candidate solutions. the next steps are done in iterative procedure: assess the fitness of all individual in the population; breed new population of children based on the fitness; select parents and children to form a next-generation population. This cycle continues until the population converges to an optimum or a certain number of iterations has been reached.

5.2.2. Evolution Strategies

Rechenberg (1978) developed the family of algorithms known as Evolution Strategies. ES typically mutate there individuals and use truncation selection as a procedure to select individuals.

One of the simplest ES algorithms is the (μ, λ) algorithm. It begins by generating a random population of λ individuals. Then, the subsequent iterations is as follows: assess the fitness of all individuals; delete $\lambda - \mu$ of the unfittest individuals, thereby the fittest μ individuals remain; tweak the remaining individuals using an ordinary mutation to produce λ/μ children; the children replace the discarded parents and the iterations starts again.

To summarize, μ is the number of parents which survive, and λ is total the number of kids that the μ parents produce. It is important to note that μ should be a factor of λ. The algorithm pseudocode is as follows:

In order to adjust the exploration versus exploitation of the (μ, λ) algorithm, there are three parameters that can be adjusted.

- The size of λ. By controlling λ, the sample size of the population can be controlled.
- The size of μ. By changing μ, the selection procedure can be influenced with low values pushing the algorithm toward a greedy search as only the fittest individuals survive each round.
- The degree to which mutation is performed. A mutation can be altered with respect to the amount of noise that gets incorporated into the offspring. A lot of noise produces children that are very different from their parents. Hence, the algorithm would resemble a random search regardless of the selectivity of μ.

5.2.3. Genetic Algorithm

The Genetic Algorithm was invented by Holland (1992). Although a lot of similarities exist to the (μ, λ) Evolution Strategy such as initialization, quality evaluation and population reassembly, there are primary differences, namely: selection and breeding. In ES, all parents are selected and then used to create children. By contrast, the GA selects a few parents gradually over time, generating children until the population is back to its original sample size.

Algorithm 4 The (μ, λ) Evolution Strategy from Luke (2013)

init μ, λ ▷ Number of parents selected, number of children generated by the parents
$P \leftarrow \{\}$
for λ times **do** ▷ Build Initial Population
    $P \leftarrow P \cup$ new random individual
end for
$S^* \leftarrow [\,]$
repeat
    for each individual $P_i \in P$ **do**
        AssessFitness(P_i)
        if $S^* = [\,]$ or $Fitness(P_i) > Fitness(S^*)$ **then**
            $S^* \leftarrow P_i$
        end if
    end for
    $Q \leftarrow$ the μ individuals in P whose $Fitness()$ are greatest ▷ Truncation Selection
    $P \leftarrow \{\}$ ▷ Join is done by just replacing P with the children
    for each individual $Q_j \in Q$ **do**
        for λ/μ times **do**
            $P \leftarrow P \cup Mutate(Copy(Q_j))$
        end for
    end for
until S^* is the is the ideal solution OR we have run out of time.
return S^*

An empty population of children is used to begin breeding. Then, two parents are selected from the original population and used to generate two offspring, which are then added the population. Children are created by copying the genotype of each parent, crossing it over (see below) with one another and then mutating the result. This process is repeated until the child population is entirely filled. The algorithm in pseudocode is given below.

Crossover

Crossover involves mixing and matching parts of the genotype of two parents to form children and is a paramount feature in the Genetic Algorithm. The genotype structure influences the way mixing and matching process can occur. If the structgure is in the form of a vector, then there are three typical ways of achieving crossover: One-Point, Two-Point, and Uniform crossover. For this literature study, only one of the crossover techniques will be explored, the One-Point crossover. Lets say the vector is of length l. One-Point crossover selects a number c between 1 and l, inclusive, and swaps all the indexes that are less than c, as shown in Figure 5.1. The algorithm is given in Algorithm 6:

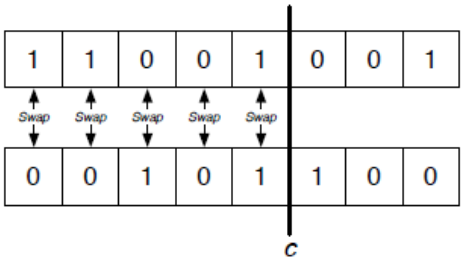


Figure 5.1: One-point crossover.

5.2.4. Particle Swarm Optimization

As mentioned previously, Particle Swarm Optimization is a stochastic optimization technique that is modelled after the swarming and flocking behaviour in animals. Whereas evolutionary algorithms resample populations to produce new ones, PSO is a directed mutation method. Hence, it does not resample but maintains a

Algorithm 5 Genetic Algorithm from Luke (2013)

init $popsize$ ▷ Desired population size (make it even).
$P \leftarrow \{\}$
for $popsize$ times **do**
    $P \leftarrow P \cup newrandomindividual$
end for
$S^* \leftarrow [\,]$
repeat
    for each individual $P_i \in P$ **do**
        AssessFitness(P_i)
        if $S^* = [\,]$ or $Fitness(P_i) > Fitness(S^*)$ **then**
            $S^* \leftarrow P_i$
        end if
        $Q \leftarrow \{\}$
        for $popsize/2times$ **do**
            Parent $P_a \leftarrow SelectWithReplacement(P)$
            Parent $P_b \leftarrow SelectWithReplacement(P)$
            Children $Ca, Cb \leftarrow Crossover(Copy(P_a), Copy(P_b))$
            $Q \leftarrow Q \cup \{Mutate(C_a), Mutate(C_b)\}$
        end for
        $P \leftarrow Q$
    end for
until S^* is the ideal solution or we have run out of time
return S^*

Algorithm 6 One-Point Crossover

$\overrightarrow{v} \leftarrow$ first vector $< v_1, v_2, ..., v_l >$ to be cross over
$\overrightarrow{w} \leftarrow$ first vector $< v_1, v_2, ..., v_l >$ to be cross over
$c \leftarrow$ random integer chosen uniformly from 1 to l
if $cneq1$ **then**
    for i from c to l **do**
        swap the values of v_i and w_i
    end for
end if

single static populations, whose members are tweaked depending on discoveries of the state space made from the population. The technique was developed by Kennedy and Eberhart (1995).

PSO typically operates in real-valued state spaces that are exclusively multidimensional metrics. The reason behind this is that a PSO candidate solution is tweaked towards various best discovered solutions so far. In literature, researchers use the term *swarm of particles* to refer to a population of individuals because it is inspired by flocks and swarms. Since there is no selection, these particle never die but they move through the state space. A particle consists of two parts:

- The particles location in space, $\overrightarrow{x} =< x_1, x_2, ... >$.
- The particles velocity, $\overrightarrow{v} =< v_1, v_2, ... >$, or its speed and direction each timestep. The velocity can be calculated as follows: $\overrightarrow{v} = \overrightarrow{x}^t - \overrightarrow{x}^{t-1}$.

Upon initialization, each particle is randomly generated with a location and velocity vector. The velocity vector is typically computed by choosing half the vector between two random points (other options are a small random vector or a zero vector). Furthermore, in order to adjust a particle positions in the state space, it is important to pay attention to the following:

- The fittest known location $\overrightarrow{x}^*$ that $\overrightarrow{x}$ has discovered so far.
- The fittest known location $\overrightarrow{x}^+$ that any of the *informants* of $\overrightarrow{x}$ have discovered so far. The informants of $\overrightarrow{x}$ are commonly a small set of particles chosen randomly each iteration. It is important to note that $\overrightarrow{x}$ is always one of its own informants.
- The fittest known location $\overrightarrow{x}^!$ that has been discovered by anyone so far.

The following iterations are performed each timestep. First, the fitness of all particles are assessed and the best-discovered locations are updated. Secondly, the particles velocity vector is mutated by incorporating a vector pointing to each of the fittest known locations $\overrightarrow{x}^*$, $\overrightarrow{x}^+$ and $\overrightarrow{x}^!$ from the particle's position $\overrightarrow{x}$. This vector is slightly augmented further with some random noise. Finally, move the particle through space by using the new adjusted velocity vector.

Algorithm 7 Particle Swarm Optimization from Luke (2013)

init $swarmsize$ ▷ Desired swarm size
init α ▷ Proportion of velocity to be retained
init β ▷ Proportion of personal best to be retained
init γ ▷ Proportion of the informants best to be retained
init δ ▷ Proportion of global best to be retained
init ϵ ▷ Jump size of a particle
$P \leftarrow \{\}$
for $swarmsize$ times **do**
    $P \leftarrow P \cup \{$ new random particle $\overrightarrow{x}$ with a random initial velocity $\overrightarrow{v}\}$
end for
$S^* = [\,]$
repeat
    for each particle $\overrightarrow{x} \in P$ with velocity $\overrightarrow{v}$ **do**
        $AssessFitness(\overrightarrow{x})$
        if $S^* = [\,]$ OR $Fitness(\overrightarrow{x}) > Fitness(S^*)$ **then**
            $S^* \leftarrow \overrightarrow{x}$
        end if
    end for
    for each particle $\overrightarrow{x} \in P$ with velocity $\overrightarrow{v}$ **do** ▷ Determine how to Mutate
        $\overrightarrow{x}^* \leftarrow$ previous fittest location of $\overrightarrow{x}$
        $\overrightarrow{x}^+ \leftarrow$ previous fittest location of informants of $\overrightarrow{x}$ ▷ Including $\overrightarrow{x}$ itself
        $\overrightarrow{x}^! \leftarrow$ previous fittest location of any particle
        for each dimension i **do**
            $b \leftarrow$ Random number from 0 to β inclusive
            $c \leftarrow$ Random number from 0 to γ inclusive
            $d \leftarrow$ Random number from 0 to δ inclusive
            $v_i = \alpha v_i + b(x_i^* x_i) + c(x_i^+ x_i) + d(x_i^! x_i)$
        end for
    end for
    for Each particle $\overrightarrow{x} \in P$ with velocity $\overrightarrow{v}$ **do** ▷ Mutate
        $\overrightarrow{x} \leftarrow \overrightarrow{x} + \epsilon \overrightarrow{x}$
    end for
until S^* is the ideal solution or we have run out of time
return S^*

5.3. Multi-Objective Methods

Researches are often interested in optimizing multiple fitness functions instead of a single function. Each fitness function is referred to as an objective that needs to be optimized. Rarely is there a case where an optimal solution can be found for every objective. Typically, objectives are juxtaposed with one another, which results in solutions that are a tradeoff of various objective. Section 5.3.1 discusses the terminology of multi-objective methods and how tradeoffs can be made. Section 5.3.2 and Section 5.3.3 introduce naive and pareto methods for multi objective optimization respectively.

5.3.1. Multi-Objective Methods Terminology

As mentioned previously, solutions to multi-objective methods are often tradeoffs between various objectives. Therefore, defining a set of best options is a difficult task. The predominant way of determining the best solutions is to analyze the *pareto front* of the space of candidate solutions, which is the the set of solutions that are not dominated by another other solution.

Consider two candidate solutions M and N, if M is at least as good as N in all objectives and better than N in at least one objective, then M *pareto dominates* N. Hence, M is a better solution because it as least as good as N in all aspects and better in one. Figure 5.2 depicts the space dominated by a given solution A if there are only two objectives. If M is only better in some objectives than N, then both solutions are of interest.

Thus, the set of solutions that are non-dominated by other solutions are of great interest when analyzing

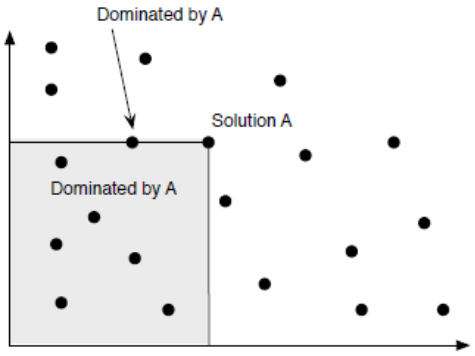


Figure 5.2: Region of solutions Pareto dominated by solution A.

multiple objectives. This set of solutions is known as the *pareto front* or *pareto non-dominated front*. Figure 5.3 shows the Pareto front of the possible solutions in our two-objective space. As can be seen in the figure, the pareto front defines the outer border of the solution space. In an optimization with two objectives, the outer border is curve. In an optimization with three objectives, the outer border becomes a surface. If one solution is better than all the others, the front collapses to that single individual.

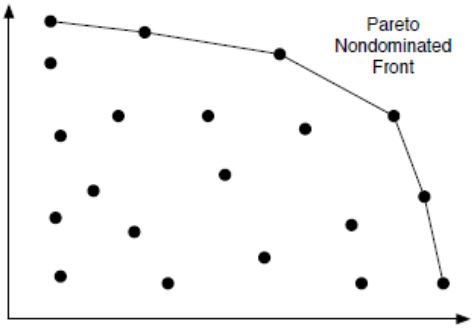


Figure 5.3: The pareto front of non-dominated solutions.

Spread

It is not enough to offer a plethora of solutions that lie on the Pareto front if all are located close to each other, as that does not give much insight into other options that are available. Instead, a set of solutions that are evenly spread across the front are preferred. Hence, most algorithms force diversity measures while optimizing for multiple objectives. Interestingly, the diversity is usually enforced with respect to euclidean distance in fitness and not genotypical distance.

The Problem of Too Many Objectives

If the number of objectives increases, then the population size required to accurately determine the pareto front grows exponentially. Therefore, all multi objective methods face computation challenges when a large number of of objectives are being used (i.e. greater than 3). As stated by Luke (2013), researches have been investigating hypervolumes covered by the pareto front. However, these techniques are complex and will not be discussed further in this section. Instead, this literature study focuses on simple multi-objective methods.

5.3.2. Naive Methods

Before describing Pareto methods, more naive methods used to transform multi-objective problems into single-objective problems are discussed. This will allow traditional metaheuristic algorithms to be used for the optimization. The simplest way of combining objectives into a single fitness function is to sum them up in a linear manner. Thus, the quality of a solution may be defined as a weighted sum of how various objectives are met:

$$Fitness(i) = Throughput(i) + \frac{1}{2}Cost(i) \tag{5.3}$$

This theme has been encountered several times in the past so far. However, there are fundamental problems with this concept. Firstly, the worth of each objective needs to be quantified and compared to another. This is already a difficult task for linear objectives due to their juxtaposition but impossible for non-linear objectives due to scaling issues. Secondly, a weighted sum will not lead the solution to the pareto front. An individual A could be located at the front but still have a lower fitness than individual B that is not located at the front. Therefore, a less desirable individual B would be selected using this fitness strategy. Lastly, the pareto front already indicates tradeoffs between objectives. Therefore, an individual can pick different combinations along the front and still evaluate their combined fitness.

Instead of giving weights to each objective, linear functions could be abandoned and the objectives are simply treated as uncomparable functions. When comparing two individuals, objectives are simulated until one is superior to the other in that objective. A tournament selection can be performed if there is an $ObjectiveValue(objective, individual)$ function in the simulation that establishes the quality of an individual with regard to the given objective. The torunament selection process is given in Algorithm 8.

Algorithm 8 Multi-Objective Majority Tournament Selection from Luke (2013)

$S^* \leftarrow$ individual picked at random from population with replacement
$O \leftarrow \{O_1, ..., O_n\}$ ▷ Objectives to assess with
t ▷ Tournament size ($t \geq 1$)
$j \leftarrow$ random number picked uniformly from 1 to n
for i from 2 to t **do**
    $Next \leftarrow$ individual picked at random from population with replacement
    if $ObjectiveValue(O_j, Next) > ObjectiveValue(O_j, Best)$ **then**
        $S^* \leftarrow Next$
    end if
end for
return S^*

5.3.3. Pareto Methods

The previous algorithm merges multiple objectives into a single fitness value by using tradeoffs. However, it also possible to use the notion of pareto domination to get more desirable solutions in a multi-objective sense. This can be achieved by constructing a tournament selection that is based on pareto domination. As mentioned previously, pareto domination is when an the fitness of individual A is at least as good as B in every objective and better than B in at least one objective (Luke 2013).

Algorithm 9 Pareto Domination Binary Tournament Selection from Luke (2013)

init P ▷ Population
$P_a \leftarrow$ individual picked at random from P with replacement
$P_b \leftarrow$ individual picked at random from P with replacement
if P_a Pareto Dominates P_b **then**
    return P_a
else if P_b Pareto Dominates P_a **then**
    return P_b
else
    return either P_a or P_b, chosen at random
end if

When an individual pareto dominates another, then selecting an individual to keep is trivial. However, how do we select an individual where neither of them pareto dominate one another? Suppose there are two individuals A and B, where individual A has many individuals that pareto dominate it and B has none. It would be preferred to select individual B because the likelihood of selecting an individual better than B in the next round is lower than for A since it closer to the pareto front.

In order to establish how close individuals are to the pareto front, a new concept called *pareto front rank* is introduced. An individual is in Rank 1 if it located in the pareto front. If these individual are removed from the population and a new pareto front is computed, then those individuals in the new front are considered to be in Rank 2. If these individuals are then removed and another new front is computed, Rank 3 would be established, and so on. Figure 5.4 shows the notion of ranks.

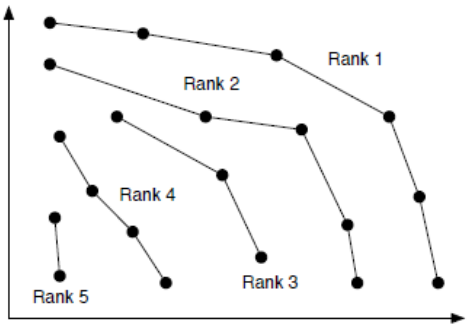


Figure 5.4: Pareto ranks.

For tournament selection, it is of paramount importance to define how to compute a pareto front. The best way to do this is keep a list of individuals that are in the pareto front. Each round the algorithm goes through the population, adds an individual if its not dominated by any of the current individuals in the front and removes corresponding individual in the front that have been dominated by the new individual. The pseudocode can be found in Algorithm 10.

Algorithm 10 Computing a Pareto Non-Dominated Front

init P ▷ Population
$G \leftarrow \{G_1, ..., G_m\}$ ▷ Group of indivduals to compute the front among
$O \leftarrow \{O_1, ..., O_n\}$ ▷ Objectives to assess
$F \leftarrow \{\}$
for $G_i \in G$ **do**
    $F \leftarrow F \cup \{G_i\}$ ▷ Assume G_i is going to be in the front
    for $F_j \in F$ other than G_i **do**
        if F_j pareto dominates G_i given O **then**
            $F \leftarrow F - \{G_i\}$ ▷ Remove G_i from front
            break
        else if G_i pareto dominates F_j given O **then**
            $F \leftarrow F - \{F_j\}$ ▷ Remove F_j from front
        end if
    end for
end for

Chapter 6

Reinforcement Learning

The idea of learning by interacting with the environment is fundamental in the nature of learning. In this chapter, computational approach to learning from interaction is explored. Section 6.1 covers Markov Decision Processes (MDPs), which are a classical formalization of sequential decision making. Section 6.2 and Section 6.3 discsuses monte carlo methods and temporal difference (TD) methods for learning respectively. Section 6.4 develops reinforcement methods that require a model of the environment, known as planning. Section 6.5 proposes methods used for when more than one agent is learning in an environment, also known as multi-agent systems. Lastly, Section 6.6 talk abouts the challenges faced in reinforcement learning (RL).

6.1. Markov Decision Processes

Finite MDPs, which are a classical formalization of sequential decision making, are introduced in this section. An agents actions affect not only immediate rewards, but also their subsequent states, thereby affecting those potential rewards as well. Hence, MDPs have immediate and delayed rewards, which require tradeoffs between them. In order to accurately assign rewards due to long-term consequences from an action selected by an individual, it is essential to know state-dependent quantities.

MDPs can be viewed as a mathematically idealized version of reinforcement learning, where precise theoretical statements can be made. This section introduces key elements to learning, such as agent-environment interface in Section 6.1.1, goals and rewards in Section 6.1.1, returns and episodes in Section 6.1.2 and value functions in Section 6.1.3.

6.1.1. Agent-Environment Interface

MDP are a mathematical realization of learning from interaction to achieve a goal. An agent is defined as an entity that is able to learn and make decisions. The environment encompasses everything that an agent can interact with, excluding itself or other agents. Both of them interact continually, where an agents takes a action and the environment responds to these actions, thereby presenting new situations to the agent. Furthermore, the environment encourages the agent to complete actions by giving it rewards, which the agent seeks to maximize over time.

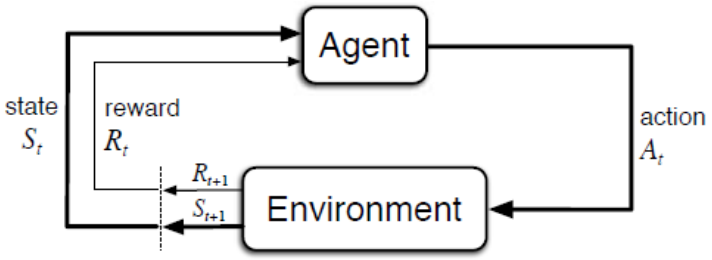


Figure 6.1: The agentenvironment interaction in a Markov decision process replicated from Sutton and Barto (2018).

Note that the interaction between the agent and environment occur at discrete time steps, $t = 0, 1, 2, 3, ..., n$. During each time step t, the agent receives a representation of the environment's states $S_t \in S$, and takes an action $A_t \in A(s)$ based on state S_t. One time step later, the agent receives a reward $R_{t+1} \in R$ and is situated in a new state S_{t+1} as a result of its previous action A_t. The MDP, the environment and the agent produce a sequence shown in Equation (6.1).

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, ... \tag{6.1}$$

A finite MDP has by definition a finite number of states, actions, and rewards. Therefore, the random variables R_t and S_t depend only on the preceding state S_{t-1} and action A_{t-1} and thereby have discrete

probability distributions. For example, given particular values of $s \in S$ and $a \in A(s)$, there is a probability that the next state and reward will be $s' \in S$ and $r \in R$ respectively. This concept is conceptualized in Equation (6.2)

$$P(s', r|s, a) = P(S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a), \tag{6.2}$$

In a MDP, the environment dynamics are completely characterized by the probabilities given by p, because the preceding state and action, S_{t-1} and A_{t-1} define the probability of each possible value for S_t and R_t. This is also known as the *Markov property*. Therefore, all aspects of the past agent-environment interaction that affect the future must be included in the state of the agent.

Goals and Rewards

In RL, the environment passes a special signal to the agent, known as reward, at each time step. The reward is typically an integer number $R_t \in R$ but can also be more complex. The agent's purpose is to maximize the total amount of reward it receives, hence the cumulative reward in the long run.

RL is distinguished by the use of a reward signal to formalize a goal. Sutton and Barto (2018) explains that although "formulating goals in terms of reward signals might at first appear limiting, in practice it has proved to be flexible and widely applicable."

6.1.2. Returns and Episodes

Until this point, the objective of learning has been conceptualized informally, stating that that an agents purpose is to maximize the cumulative reward it receives in the long run.

In a mathematical context, "an agent seek to maximize the expected return, where the return, denoted G_t, is defined as some specific function of the reward sequence" denoted by Sutton and Barto (2018). Typically, the return is described by the sum of the rewards as shown in Equation (6.3), where T is the final step. This approach is compatible in simulations in which there is a final timestep, such as computer games. The agent-environment interaction can be broken into subsequences, known as *episodes*, and tasks with episodes are known *episodic tasks*. In an episodic task, the time of termination, T, varies from episode to episode.

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + ... + R_T \tag{6.3}$$

By contrast, there are many agent-environment interactions that do not break into identifiable episodes, but go on without a limit, known as *continuing tasks*. For continuing tasks, the return of Equation (6.3) is problematic because the return itself could be infinite since the timestep is $T = \infty$.

In order to maximize the return for continuing tasks, an additional concept known as *discounting* is introduced. According to this approach, Sutton and Barto (2018) states that "the agent tries to select actions so that the sum of the *discounted rewards* it receives over the future is maximized". Specifically, an agent selects A_t to maximize the expected discounted return described in Equation (6.4), where γ is $0 \leq \gamma \leq 1$ and called the *discount rate*.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 + R_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{6.4}$$

The discount rate determines the value of future reward experienced in current state S_t. A future reward is worth only γ^{k1} times what it would be worth if it were received immediately, where k represents the time steps in the future. If $\gamma = 0$, the agent is *myopic* and concerned only with maximizing immediate rewards. If $0 < \gamma < 1$ and the reward sequence R_k is bounded, then Equation (6.4) has a finite value.

6.1.3. Policies and Value Functions

Typical RL algorithms involve estimating *value functions*, which are functions of states that estimate "how good it is for the agent to be in a given state" or "how good it is to perform a given action in a given state" as described by Sutton and Barto (2018). The goodness of a state is defined in terms of the expected return (i.e. the cumulative future rewards that can be expected). Note that future reward depend on the actions an agent selects. Hence, value functions are defined by how an agent acts, also known as *policies*.

Mathematically, a *policy* π is the probability of an agent selecting action $A_t = a$ given that the it is in state $S_t = s$. Hence, it is probabilistic mapping $\pi(a|s)$ from states to each possible action in that state.

RL methods specify how the agent's policy changes due to its experiences in the environment. A value function for state s under a policy π, denoted $v_\pi(s)$, is "the expected return when starting in s and following π thereafter" as given by Sutton and Barto (2018). For MDPs, $v_\pi(s)$ is defined according to Equation (6.5), where E_π denotes the expected return value given that the agent follows policy π. The function v_π is called *the state-value function* for policy π.

$$v_\pi(s) = E_\pi\left[G_t|S_t = s\right] = E_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1} \mid S_t = s\right] \quad \forall s \in S \tag{6.5}$$

It is important to note that the value of the terminal state, if any, is zero.

Similarly, $q_\pi(s,a)$ denotes the value of being in state s and taking action a under a policy π and is known as the *action-value function* for policy π. It is defined as the expected return taking the action a from state s, and thereafter following policy π (shown in Equation (6.6)).

$$q_\pi(s,a) = E_\pi\left[G_t|S_t = s, A_t = a\right] = E_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right] \quad \forall s \in S \text{ and } a \in A \tag{6.6}$$

In RL, a basic feature of value functions is that recursive relationships are fulfilled. The following consistency condition in Equation (6.7) holds between the value of s and the value of its possible successor states s' for any policy π and any state s,

$$\begin{aligned} v_\pi(s) &= E_\pi\left[G_t|S_t = s\right] \\ &= E_\pi\left[R_{t+1} + \gamma G_{t+1}|S_t = s\right] \\ &= E_\pi\left[R_{t+1} + \gamma v_\pi(s')|S_t = s\right] \\ &= \sum_a \pi(a|s)\sum_{s'} p(s'|s,a)\left[E[r|s,a,s'] + \gamma v_\pi(s')\right] \\ &= \sum_a \pi(a|s)\sum_{s',r} p(s',r|s,a)\left[r + \gamma v_\pi(s')\right] \end{aligned} \tag{6.7}$$

Equation (6.7) is the *Bellman equation* for v_π, which expresses a mathematical relationship between the value of a state and the values of its successor states.

In Figure 6.2, every open circle represents a state, and a state-action pair is represented by a solid circle. The agent will take a set of potential actions from the root state s (three in Figure 6.2) depending on its π policy. Based on each of these actions, the environment responds with one of several next states s' (two in Figure 6.2) and a reward r that depends on the system dynamics given by p. The Bellman equation states that the starting state value must be equal to the predicted state's discounted value plus the reward averaged over all possibilities along the way. Likewise, there is a bellman equation for $q_\pi(s,a)$ given in eq. (6.8).

$$q_\pi(s,a) = \sum_{s'} p(s'|s,a)\left[E[r|s,a,s'] + \gamma\sum_{a'}\pi(a'|s')q_\pi(s',a')\right]$$

It is important to note that the state-value function can also be written in terms of the action-value function as demonstrated in eq. (6.8).

$$v_\pi(s) = \sum_a \pi(s,a)q_\pi(s,a) \tag{6.8}$$

Optimality Conditions

Solving a reinforcement learning problem involves finding an optimal policy that maximizes the long-term expected return. Policy performance can be described in the following way for finite MDPs: a π policy performs better than or equal to a π' policy if its projected return for all states is greater than or equal to that of π'. In a mathematical context, $\pi \geq \pi'$ if and only if $v_\pi(s) \geq v_{\pi'}(s)$ for all $s \in S$. Notice that at least one policy exists, often referred to as *optimal policy*, that is greater than or equivalent to all

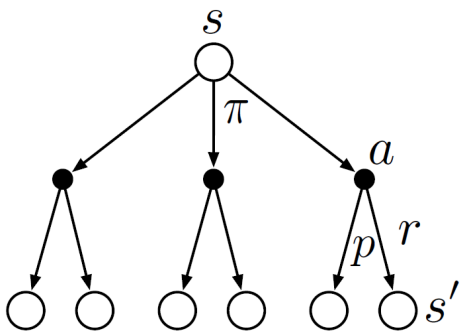


Figure 6.2: Backup diagram bellman taken from Sutton and Barto (2018).

other policies. Although there could be more than one, $\pi*$ denotes all of the optimal policies and their corresponding optimal state-value function is denoted by v_*, as defined in eq. (6.9).

$$v_*(s) = \max_\pi v_\pi(s) \quad \forall s \in S \tag{6.9}$$

Optimal policies also share the same optimal action-value function, denoted Q_*, and defined according to eq. (6.10).

$$q_*(s,a) = \max_\pi q_\pi(s,a) \quad \forall s \in S \text{ and } a \in A \tag{6.10}$$

Since v_* is the value function for a policy, the recursive condition given by eq. (6.7) for state values must be satisfied. However, as it is an optimal value function, a simplified condition can be derived (one without reference to a specific policy). This is the *Bellman Optimality Equation* as shown in Equation (6.11). Intuitively, the Bellman equation for v_* expresses that the expected return for selecting the best action from a state is equal to value of a state under an optimal policy.

$$\begin{aligned}
v_*(s) &= \max_{a \in A(s)} q_{\pi_*}(s,a) \\
&= \max_a E_{\pi_*}[G_t | S_t = s, A_t = a] \\
&= \max_a E_{\pi_*}[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a] \\
&= \max_a E_{\pi_*}[R_{t+1} + \gamma V_*(s') | S_t = s, A_t = a] \\
&= \max_a \sum_{s'} p(s'|s,a)[E[r|s,a,s'] + \gamma v_*(s')] \\
&= \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma v_*(s')]
\end{aligned} \tag{6.11}$$

6.2. Monte Carlo Method

To learn the state-value function for a given policy, Monte Carlo (MC) methods are introduced.

Recall that the definition of the value of a state is the expected return starting from that state. One plausible way to quantify it is to average the returns following a visit to that state. A principle idea of MC methods is the average will converge to the expected value as the number of returns observed increases.

Given a set of episodes earned by adopting policy π and passing through s, it is possible to estimate $V_\pi(s)$. A repetition of state s during an episode is called a *visit* to s. It is important to note that s can be visited multiples times within the same episode.

The every-visit MC method obtains an estimate by averaging the returns following all visits to state s. On the other hand, The first-visit MC method estimates $v_\pi(s)$ by averaging the returns following a first visit to state s. Both of these methods are similar, however their theoretical properties are different. This section focuses on the latter method as it is more intuitive. The algorithm is presented in Algorithm 11.

As mentioned previously, when the number of first visits to s goes to infinity, then the first-visit MC converges to $v_\pi(s)$. Since each return is an independent, yet equal distributed approximation of $v_\pi(s)$ with

Algorithm 11 First-Visit MC Prediction from Sutton and Barto (2018)

init $\pi, V(S) \in R$ ▷ Policy, Value of state s (for all $s \in S$)
$Return(S) \leftarrow \{\}$ ▷ For all $s \in S$
repeat
  Generate an $episode$ following : $S_0, A_0, R_1, S_1, A_1, R_2, ..., S_{t-1}, A_{t-1}, R_t$
  $G \leftarrow 0$
  for each $step$ of $episode$ **do**
    $G \leftarrow \gamma G + R_{t+1}$
    if S_t appears in $S_0, S_1, ..., S_{t-1}$ **then**
      $Returns(S_t) \leftarrow G$
      $V(S_t) \leftarrow average(Returns(S_t)$
    end if
  end for
until Run out of time

finite variance, then the average of these estimates converges to their expected value due to the law of large numbers. It is important to note that each average is an unbiased estimate of $V_\pi(s)$. Therefore, the standard deviation σ decreases by $1/\sqrt{n}$, where n is the number of returns averaged.

6.3. Temporal Difference Learning

Temporal difference (TD) learning is a fundamental concept to learn by reinforcement. It is a combination of dynamic programming (DP) and MC ideas. Similar to MC methods, TD methods do not need a model of environment-dynamic and are able to learn directly from raw experiences. Furthermore, they *bootstrap* like TD methods. Bootstrapping is a technique where a model updates estimates based on other estimates, instead of waiting for the final outcome. It was decided to not talk about dynamic programming in this literature study because it is more theoretical than applied.

6.3.1. Temporal Difference Prediction

Both MC and TD methods use experience to solve the issue of estimation. Using the experience obtained by adopting a policy π, their estimate V of v_π for the nonterminal states is updated. Typically, techniques based on MC wait until a visit reoccurs and use that return to update $V(S_t)$. In eq. (6.12), an every-visit Monte Carlo method suitable for non-stationary environments is demonstrated, where G_t is the actual return following time t, and α is a constant step-size parameter.

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)] \tag{6.12}$$

As mentioned previously, MC methods have to wait until an episode finishes in order to determine the target for $V(S_t)$. By contrast, only the next time step is needed for TD methods. At time $t+1$, TD methods obtain a target that is equal to the sum of the observed reward R_{t+1} and the estimate $V(S_{t+1})$ multiplied by a discount factor γ. Equation (6.13) indicates a simple TD method update that occurs on transition to S_{t+1} and receiving R_{t+1}. The TD method in Algorithm 12 is called one-step TD method.

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \tag{6.13}$$

TD methods offer unique advantages over DP methods and MC methods. First, they do not require a model of the environment, nor next-state probability distributions as in DP methods. The second most apparent benefit of TD methods is that they are naturally applied in an incremental, online manner. This is because they only have to wait one time step to know the expected return whereas MC methods must wait until the end of an episode. Interestingly, this is a crucial consideration when deciding between MC and TD methods. While MC methods are ideal mathematical version of learning, they are slow to learn for various application. Episodes can be very long, which delays learning, or even continuous tasks, making it impossible to learn as there are no episodes. Lastly, TD methods learn from each transition regardless of the action taken. Thus, they do not need to discount episodes as in MC methods.

Algorithm 12 Tabular TD(0) from Sutton and Barto (2018)

init $V(s) \forall s \in S$ ▷ Except that $V(terminal) = 0$
for episode in episodes **do**
  init S
  repeat for each $step$ of $episode$
    $A \leftarrow$ given by π for S
    Obsverve R, S' by taking action A
    $V(S) \leftarrow V(S) + \alpha[R + \gamma V(S') - V(S)$
    $S \leftarrow S'$
  until S is terminal
end for

6.3.2. SARSA

By following the usual pattern of generalized policy iteration, a new algorithm, also known as SARSA, can be constructed by using TD methods for estimation. Similarly to MC methods, algorithms can be classified into *on-policy* and *off-policy* approaches. An on-policy learner discovers policy π while the agent carries out the same policy. An off-policy learner discovers policy π_1 while following the return of another policy π_2. In essence, on-policy learns how good it is to do something by doing it and off-policies learns how good it is to do something while doing another thing.

In this subsection, an on-policy TD control method is presented. The reason SARSA is on-policy is because it approximates $q_\pi(s, a)$ by following policy π for all actions a and states s. In other words, it assumes that the current policy π continues to be followed and then estimates the return for state-action pairs. This can be accomplished by applying the same described in Algorithm 12 for learning v_π but for q_π. In order to derive the algorithm, it is important to know that an episode consists of an alternating sequence of states and stateaction pairs as shown in fig. 6.3.

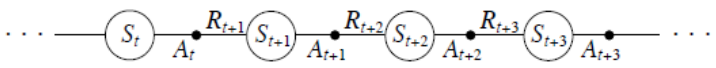


Figure 6.3: State-action pairs taken from Sutton and Barto (2018).

While transition between states were considered for TD methods, this new on-policy method considers transition between state-action pairs and learns the values of state-action pairs. Although these sequences are different, they are both markov chains and obey the same theorems. For example, the law of large number assures the convergence applies to the corresponding update shown in eq. (6.14), similarly to that of $TD(0)$.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)] \tag{6.14}$$

The update is done for each non-terminal state S_t after taking action A_t, where the estimated return is dependent on the reward R_t, the next state S_{t+1} and the next action A_{t+1}. Putting the terms together $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$ give the algorithms its name: SARSA. If S_{t+1} is a terminal state, then the expected return is 0.

Using the SARSA prediction method, it becomes possible to design an on-policy control algorithm. As mentioned previously, on-policy methods estimate q_π by adopting behavior policy π, while changing their policy π to increase the expected return with respect to q_π. Algorithm 13 presents the general SARSA algorithm. The convergence properties of the SARSA algorithm are guaranteed if all state-action pair are visited an infinite number of times.

6.3.3. Q-Learning

The development of an off-policy TD control algorithm known as Q-learning (Watkins and Dayan 1992) was one of the early breakthroughs in RL. Its update policy is defined by eq. (6.15).

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \tag{6.15}$$

Algorithm 13 SARSA

init $Q(s,a) \forall s \in S, a \in A$ ▷ Except that $Q(terminal, \cdot) = 0$
for episode in episodes **do**
  init S
  Choose A from S using policy derived from Q
  repeat for each $step$ of $episode$
    take action A, observe R and S'
    Choose A' from S' using policy derived from Q
    $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma Q(S', A') - Q(S,A)]$
    $S \leftarrow S'$ and $A \leftarrow A'$
  until S is terminal
end for

The reason that Q-learning is off-policy is that it approximates $q_\pi(s,a)$ independently of the adopted policy. In essence, it approximates the expected return for state-action pairs assuming a greedy policy is adopted even though it can be visiting any state-action pair according to some type of policy. Watkins and Dayan (1992) demonstrate if state-action pairs are continually visited and updated, then the algorithm converges to an optimal policy.

From a RL standpoint, this isnt a very interesting algorithm if P and R are already known. However, it becomes interesting if the agent can learn q_π simply by exploring the environment and experiencing P and R without knowing what they actually are. These type of algorithms (i.e. ones that do not know P and R) are known as *model-free* algorithms. Algorithm 14 present the Q-learning algorithm.

Algorithm 14 Q-Learning from Sutton and Barto (2018)

init $Q(s,a) \forall s \in S, a \in A$ ▷ Except that $Q(terminal, \cdot) = 0$
for episode in episodes **do**
  init S
  Choose A from S using policy derived from Q
  repeat for each $step$ of $episode$
    Take action A, observe R and S'
    Choose A' from S' using policy derived from Q
    $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S,A)]$
    $S \leftarrow S'$
  until S is terminal
end for

6.3.4. Double Q-Learning

Q-learning is one of the breakthroughs in TD learning but one of its main disadvantages is the that is uses a maximization function to construct its target policies. As a result, q-learning continuously overestimates its true value $q(s,a)$. This occurs because estimated values $Q(s,a)$ are noisy compared to the true values $q(s,a)$, hence they will be distributed around $q(s,a)$, with some being greater and some less than the true values. If the maximum of the estimated values $Q(s,a)$ is taken, it will be greater than the maximum of the true value $q(s,a)$. Hence, a positive bias exists, which is also known as *maximization bias*. Due to this maximization bias, q-learning performs poorly in stochastic environments.

In a study conducted by Hasselt (2010), it was shown the maximization bias can be avoided by using separate samples to determine the maximizing action and to estimate its values. For example, consider two estimates $Q_1(a)$ and $Q_2 a$, that are two independent estimates of two separate sets of actions, where each estimate is updated by the other in the next state. The first update consists of finding the maximizing action of Q_1 in the next state (i.e. $a^* = \arg\max_a Q_1(a)$). Then, that action a^* is used to determine $Q_2(a^*) = Q2(\arg\max_a Q_1(a))$ in order to update the estimate $Q_1(a)$. Since $E[Q_2(a^*)] = q(a^*)$, then the estimate is unbiased. This process can be repeated with $Q_1(a)$ and $Q_2(a)$ reversed in order to produce a second unbiased estimate $Q_1(\arg\max_a Q_2(a))$. The proposed solution is the fundamental idea behind double learning. It is important to note that double learning does not increase the computation per step because

only one estimate is updated each play. However, it does double the memory requirements because two estimates are stored.

The concept of double learning can easily be integrated into algorithms for MDPs. For example, the time steps in q-learning can be divided in two to create a new algorithm known as double q-learning (Hasselt 2010). In the first time step, the update is as follows:

$$Q_1(S_t, A_t) \leftarrow Q_1(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q_2(S_{t+1}, \arg\max_a Q_1(S_{t+a}, a)) - Q_1(S_t, A_t)] \tag{6.16}$$

In the the second time step, Q_1 and Q_2 are reversed, such that Q_2 is updated. Both of these estimators are treated symmetrically in all of the updates. A complete algorithm for Double Q-learning is given in algorithm 15, where an $\epsilon - greedy$ policy is used as the behvaiour policy for based on the sum of the two estimates Q_1 and Q_2.

Algorithm 15 Q-Learning

init $Q_1(s,a) and Q_2(s,a) \forall s \in S, a \in A$ ▷ Except that $Q(terminal, \cdot) = 0$
for episode in episodes **do**
  init S
  repeat for each $step$ of $episode$
    Choose A from S using policy $\epsilon - greedy$ from $Q_1 + Q_2$
    Take action A, observe R and S'
    if $random(0,1) \geq 0.5$ **then**
      $Q_1(S_t, A_t) \leftarrow Q_1(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q_2(S_{t+1}, \arg\max_a Q_1(S_{t+1}, a)) - Q_1(S_t, A_t)]$
    else
      $Q_2(S_t, A_t) \leftarrow Q_2(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q_1(S_{t+1}, \arg\max_a Q_2(S_{t+1}, a)) - Q_2(S_t, A_t)]$
    end if
    $S \leftarrow S'$
  until S is terminal
end for

6.4. Planning

In this section, a unified view of model-based and model-free RL methods is developed. Model-based methods required a model of the environment whereas model-free methods do not. As such, the primary component of model-based methods is planning, while for model-free methods its learning. Although distinct differences exist between these two methods, they share many similarities as well. For example, both methods compute value functions by updating it with estimated value from looking ahead into the future. Section 6.4.1 introduces model-based methods. Section 6.4.2 and Section 6.4.3 each present a unique algorithm used for planning, dyna-q and prioritized sweeping respectively.

6.4.1. Models and Planning

Agents that model the environment predict how the environment will respond to its actions. A model provides a prediction of the next state and next reward, given a state and an action. If there are several potential next states and next rewards, then the model is stochastic. *Distributed models* produce a description of all possible state-reward pairs and their probabilities of occurring, whereas *sample models* indicate only one of the possibilities.

These types of models can be used to simulate experiences, instead of requiring the agent to explore the environment. A sample model and a distributed model can generate one episode and all possible episodes respectively with their likelihood of occurring. Therefore, they can reduce an agent's need to explore the environment because the models can be used to simulate the environment and provide *simulated experience*.

There are many different definitions of the word *planning*. This literature study adopts the definition from Sutton and Barto (2018) as "any computational process that takes a model as input and produces or improves a policy for interacting with the modeled environment". According to this definition, there are two unique approaches to planning possible in the field of AI. *State-space planning*, which is defined by Sutton and Barto (2018) as "a search through the state space for an optimal policy or an optimal path to a goal".

Value functions are computed over states, where state to state transitions occur due to actions. This is also the approach that this literature study focuses on. On the other hand, *plan-space planning* is instead "a search through the space of plans" defined by Sutton and Barto (2018).

There are two fundamental ideas behind all state-space planning methods: the computation of value functions is done as an intermediate step to updating a policy; the computation of value function updates based on simulated experiences. Figure 6.4 presents a diagram of this common structure.

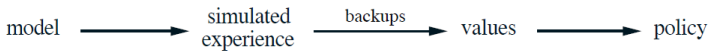


Figure 6.4: State-space planning taken from Sutton and Barto (2018).

6.4.2. Dyna-Q

A number of interesting issues arise when planning is done online, while interacting with the environment. Interaction with the environment leads to new information gained, thereby having the potential to change model, which will influence planning. In order to prevent this, the planning process can be customized such that it only considers states or decisions in the near future. An online planning agent with simple architecture, known as a Dyna-Q agent, is introduced for this purpose.

Real experiences can be used in two ways by planning agents: experiences can update the value function of the agent (similar to RL method discussed in in section 6.2 and section 6.3) and experiences can improve the model (to resemble the real environment better). These are known as *direct reinforcement learning* and *model learning* respectively. Figure 6.5 illustrates how experience affects a policy either directly or indirectly through a model. The latter also known as *indirect reinforcement learning*.

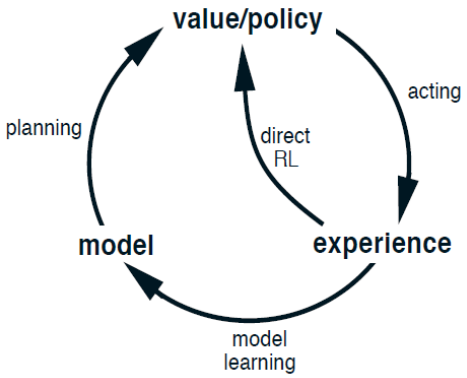


Figure 6.5: Model-learning and direct reinforcement learning diagram taken from Sutton and Barto (2018).

There are advantages and disadvantages to both direct and indirect methods. Indirect methods achieve better policies with fewer environmental interactions due utilizing the experiences more. By contrast, direct methods are not affected by biases that appear when designing a model and are much simpler. Researchers have argued both ways to which type of learning is superior but no common consensus is found. It is interesting to note that most human and animal learning follow direct methods.

A Dyna-Q agent incorporates all of the processes shown in Figure 6.5, which include acting, direct RL, model learning and planning, in a continuous manner. It is a combination of multiple algorithms each used for an individual process. This literature study puts a specific focus on the model-learning aspect as this has not been seen yet. It assumes that the environment is deterministic, such that the output is only determined by the input and initial states. The method is table-based in that it keeps a table that logs all of the transitions of state-action pairs experienced (i.e. $S_t, A_t \rightarrow R_{t+1}, S_{t+1}$). If the model is queried for a specific state-action pair that has been experienced before, it outputs the state and reward that were observed last from that state-action pair. During planning, the model is queried only from state-action pairs that have previously been experienced. In essence, real experience learned by the model give rise to simulated experience. When starting states and actions are selected to be queried by the model to generate simulated experience, it is known as *search control*.

Finally, Sutton and Barto (2018) states that "planning is achieved by applying reinforcement learning methods to the simulated experiences just as if they had really happened". The same RL methods for direct RL and and planning are used to update the policy. Learning and planning share many similarities, with their main difference being the source of their experience.

In Dyna-Q agent, each process occurs simultaneously when considering a conceptualized version. However, due to ease of implementations dicrete timesteps are used. The planning process is the most computationally intensive part in the Dyna-Q agent.

Algorithm 16 Dyna-Q-Learning from Sutton and Barto (2018)

init $Q(s,a), Model(s,a) \quad \forall s \in S, a \in A(s)$
repeat
  $S \leftarrow$ current (nonterminal) state
  $A \leftarrow \epsilon - greedy(S,Q)$
  Take action A, observe resultant reward R, and state S'
  $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_a Q(S0,a)Q(S,A)]$
  $Model(S,A) \leftarrow R, S'$ (assuming deterministic environment)
  for n times **do**
    $S \leftarrow$ random previously observed state
    $A \leftarrow$ random action previously taken in S
    $R, S' \leftarrow Model(S,A)$
    $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_a Q(S0,a)Q(S,A)]$
  end for
until Run out of time

6.4.3. Prioritized Sweeping

When planning occurs for Dyna-Q agents, simulated experience occurs by randomly selecting state-action pairs from a uniform distribution. The main consequence of a uniform selection is that many updates are wasted on state-action pairs that are not useful towards achieving the objective. In order to improve planning, simulated experiences should focus on specific state-action pairs. This is especially useful when large state-action spaces are considered because an unfocused search would require more computation time before converging to an optimal policy.

By going backward from goal states, it is possible to focus the search. However, methods should be tailored for general reward functions instead of an idealistic goal state. Therefore, a search can be focused by working back from states whose values have changed. A change of an estimated value indicates that a plethora of other states should also change values. Hence, consider an agent that is exploring the environment and one of its estimated values for a state changes. Then, a one-step update should focus on the actions that caused a a change of value in that state. By updating the values of these actions, it is possible for predecessor states to change as well. Again, the actions that affect the predecessor states should be updated, which in turn might update states prior to the predecessor states. This backward propagation can continue performing useful updates or can be terminated after a certain amount of steps.

When back propagation is applied over multiple steps, the number of state-action pairs that need to be updated usually increases exponentially. Thus, it is useful to prioritize which pairs to update based on their usefulness. One possible way of accomplishing this is to evaluate the magnitude of the change of an estimated value. Updates done on states whose estimated values changed a lot, have a high likelihood of producing bigger changes in their predecessor states. The prioritization of updates according to urgency is a fundamental concept to prioritized sweeping. Prioritization is tracked by maintaining a queue of every state-action pair whose estimated value changed, order by the magnitude of the change. When the pair with the highest magnitude is updated, then the resulting change on its predecessor is determined. If the change is bigger than a certain threshold it is added to the queue. It is important to note that if the state-action pair already exists in the queue, then the maximum magnitude change is kept. The pairs are re-prioritized and a next updated is performed. Algorithm 17 presents the prioritized sweeping method.

Algorithm 17 Prioritized Sweeping from Sutton and Barto (2018)

init $Q(s,a), Model(s,a) \quad \forall s \in S, a \in A(s)$
repeat
  $S \leftarrow$ current (nonterminal) state
  $A \leftarrow policy(S,Q)$
  Take action A; observe resultant reward R, and state S'
  $Model(S,A) \leftarrow R, S'$
  $P \leftarrow |R + \gamma \max_a Q(S',a)Q(S,A)|$
  if $P > \theta$ **then**
    insert S, A into $PQueue$ with priority P
  end if
  while $PQueue$ is not empty **do**
    $S, A \leftarrow first(PQueue)$
    $R, S' \leftarrow Model(S,A)$
    $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_a Q(S',a)Q(S,A)$
    for S, A predicted to lead S **do**
      $R \leftarrow$ predicted reward for S, A, S
      $P \leftarrow |R + \gamma \max_a Q(S',a)Q(S,A)|$
      if $P > \theta$ **then**
        insert S, A into $PQueue$ with priority P
      end if
    end for
  end while
until Run out of time

6.5. Multi-Agent Learning

Impressive results have been achieve by RL algorithms, especially when considering the convergence results to an optimal policy. Due to their ability to learn from scratch, they are able to discover patterns that escape even humans. However, much of the literature study up to now has been focused on a single learning agent. Therefore, this section will address *multi-agent system*, which are systems where multiple agent are learning at the same time.

Similar to reinforcement learning techniques, the goal for multi-agent system is often to monitor or explore a given space. In RL, an agent learns a map of the area individually, whereas in multi-agent systems multiple agents explore the environment, where each agent has its own perspective of the area that it is exploring. Through *cooperative learning*, agents are able cooperate in order to achieve the best possible outcome for the system. In an exploring setting, this entails that each agent shares their maps in order to achieve a global perspective of the environment. The agents are then able to decide which areas are interesting for further exploration. Another possibility is that agents behave *competitively*, where each agent wants to maximize its own objective. In these environment, agents will exploit other agents weaknesses and thus it becomes interesting to analyze if the agents reach a stable equilibrium.

The reward function for a multi-agent system is a function of the state and all of actions of the agents. Hence, the reward of an agent not only depend on its own actions but also on the actions of other agents. Transitions between state-action pairs are captured by multi-agent MDPs, from which it is very difficult to converge to an optimal policy. When considering only a single agent, the optimal policy maximizes the agents discounted rewards. However, in a multi-agent setting, the optimal policy can be various equilibrium points depending on the user. For example, it is possible to maximize the sum of all agents discounted rewards, also known as *social welfare*. Another possibility is to choose the *Nash equilibrium point*, which is a set of policies where no agent has anything to gain by changing their strategy. It is important to note that the Nash equilibrium point always exists. Using the NashQ-learning approach in Algorithm 18, it is possible to find the Nash equilibrium point.

6.6. Challenges in Reinforcement Learning

Enclosed simulation environments, such as video games, are preferable to use for RL because they have quantifiable rewards and the ability to give many training examples. Computational simulations are able to

Algorithm 18 NashQ-Learning

$t \leftarrow 0$
$s \leftarrow currentstate$
$\forall_{s \in S} \forall_{j \leftarrow 1,...,n} \forall_{a_j \in A_j} Q_j^t(s, a_1, ..., a_n) \leftarrow 0$
repeat
    Choose action a_i^t
    ss
    Observe $r_1^t, ..., r_n^t; a_1^t, ..., a_n^t; s'$
    for j in range n **do**
        $Q_j^{t+1}(s, a_1, ..., a_n) \leftarrow (1\lambda t) Q_j^t(s, a_1, ..., a_n) + \lambda^t (r_j^t + \gamma Nash Q_j^t(s))$
        $NashQtj(s) = Q_j^t(s, \pi_1(s), ..., \pi_n(s))$ and 1(s) ů ů ů n(s) are Nash EP calculated from Q values
    end for
    $t \leftarrow t + 1$
until Run out of time

run in parallel and can be sped up to run faster than real-time. However, many RL techniques (i.e. deep Q learning (Mnih et al. 2013)) require millions of training points due to fundamental problems of learning. The first reason is that RL agents often learn from scratch and have no knowledge regarding the environment. Another reason is that the environment often provides only a sparse reward signal. A sparse reward signal is defined by a sequence of rewards, where most of them are non-positive. Hence, an agent will have difficulties learning which actions contribute to a distant future reward. For very sparse reward systems, an agent might never be able to learn how to perform a task because it never found a reward signal. Sparse reward signal do not hinder humans from achieving tasks, due to their intrinsic motivation (Pathak et al. 2017).

The most intuitive solution to sparse reward problems is *reward shaping*. Mataric (1994) formulated the idea back in 1994, and it has been used widely ever since. Reward Shaping means that we enhance the primary reward of the environment with some additional reward features. By using additional reward features, we shape the primary reward to appropriately reward or punish interactions, filling the gap of the original sparse reward feature (Jaderberg et al. 2016). While the approach has a few upsides, it also comes with various problems. The additional reward functions are usually hand-crafted and require human experts to be successful. In addition to the expertise needed, human-crafted reward functions also have the problem of introducing a human bias to the possible policies the agent will find to solve the problem. Thinking of a complex game like chess, it is not easy to find experts that can design reward functions for the agent that may vary with the state of the game. Moreover, by using hand-crafted reward functions, the agent might fail to discover new policies that humans have not found yet.

A second general idea is to give incentives to an agent to learn about new things that it discovers in its environment. In most default RL algorithms, agents use a greedy exploration, which in many scenarios causes an agent to learn a simple type of behaviour that earns recurring low amount of reward. Since the agent does not continue to explore, it become stuck in a local minimum with small, recurring rewards. One way to combat this is to create an additional reward signal that gives incentives to the agent to explore unseen regions in the state space. One particular paper by Pathak et al. (2017) uses an intrinsic curiosity model to get a more robust exploration strategy.

Lastly, Andrychowicz et al. (2017) use a new method called hindsight experience replay. The general idea behind hindsight experience replay is that the agent should learn from all episodes even if the episode was not successful. It applies a clever trick to get an agent to learn from unsuccessful episodes: instead of telling the agent the episode was unsuccessful and it receives a reward of 0, it will pretend that what the agent accomplished was what it was supposed to do and reinforce it with a *virtual goal*. This essentially create a dense reward signal out of a sparse reward signal.

Chapter 7

Research Proposal

Over the past decades, there has been a dramatic increase in airport terminal models. Such research has demonstrated the ability to model airport processes, events and activities of passengers. These models fall under three distinct categories: capacity planning, airport efficiency and airport security. Many models have been built upon and turned into airport terminal simulators of varying capabilities. Recent breakthroughs in computational power have allowed airport simulators to become ever more detailed, modelling passenger flows to passenger activities.

Despite these incredible results, much is still unknown about the operational impact of the airport security checkpoint. Hence, section 7.1 summarizes limitations of existing literature section 7.2 proposing research questions to address these limitations.

7.1. Summary of limitations in literature

As mentioned previously, many airport terminal simulation software have been analyzed in this literature study (see Chapter 4). In the past, simulators often had to tradeoff more detailed representation with computational cost and greater need for data. However, recent developments in computation have seen a proliferation of more microscopic simulators and models. One of the main pitfalls for academia research is that most simulators are not open-source, therefore it is difficult to obtain and share information as to how they model airport terminals.

From an industry perspective, simulators are of paramount importance because new terminal concepts can be simulated without needing to reconstruct the airport. They allow airport architects to design, test and evaluate new concepts which aid decision making of airport terminal design.

Therefore, *valid simulators are needed that are open-source and able to model new security concepts*. Hence, it has been decided to use the AATOM simulator for future research. Since these new concepts have not been tried before in literature, *it is unclear how different security concepts influence the total performance of the system and where bottlenecks in the operations are*.

Indeed, existing literature has modelled the outbound passenger facilitation process. However, the main focus has been on the overall performance or check-in. Previous studies in the field have not treated the security checkpoint in much detail. Furthermore, it is now well established that small-scale interactions have to be modelled in order to capture emergent behaviour that leads to bottlenecks. *Hence, passenger flow must be modelled in order to capture bottlenecks in the design of a security checkpoint configuration*. Furthermore, the use of advance computational techniques, such as simulation optimization or reinforcement learning, have rarely been seen before in airport terminal literature. The latter has not been used at all in airport terminal literature. Therefore, it would be of interest *to use reinforcement learning to help design airport security checkpoint configurations*.

Lastly, *Schiphol Airport* sees a need to improve their allocation of resources during their operation of the security checkpoint. There is an increasing concern that resources, such as airport operators, body scanners, drop-off positions and security lanes, are not used to their full potential when constraints such as throughput, cost and area are considered. Thus, there is a *need to determine and evaluate optimal policies that improve operational efficiency*. An optimal policy is defined as the best strategy to take, thus the best mapping from perceived state of the environment to action taken. From existing literature, it has been shown that optimal policies can be found through the use of reinforcement learning. In reinforcement learning, an agent converges to its optimal policy by learning to perform actions in an environment that maximize its cumulative reward. Usually, rewards are given to the agent one time step after it performs an action. However, in an security checkpoint setting, this is not possible because the efficiency can only be known after a certain amount of time since that action has been performed. Therefore, *delayed rewards are required to be given to an agent*. For the purpose of this thesis, q-learning will be used, which is a model-free algorithm, as it is one of the most common reinforcement learning techniques in literature.

Therefore, an open-source airport terminal simulator is necessary to accommodate new security checkpoint configurations that can evaluate their performance and *to discover optimal policies that improve*

operational efficiency. The term *operational concept* is used to combine the design of an airport security checkpoint configuration and policies that might be developed.

7.2. Research Objectives

The main objective of this research is:

To design and evaluate new airport security checkpoint operational concepts for outbound passengers by using q-learning with delayed rewards.

In order to address the research objectives, the following research questions need to be answered.

1. How are airport security checkpoint operational concepts going to be designed?
   (a) How are concepts going to differ from one another?
   (b) How much design freedom will a reinforcement learning agent have in a concept?
   (c) How are concepts going to be comparable?
   (d) Which active policies are going to be analyzed?
   (e) How are concepts going to be simulated?
      i. What input parameters can be adjusted in each concept?
      ii. What modifications are required in the Simulator?

   The first research question addresses the operational concepts and their design. Sub-questions are used to help scope a case study by choosing important aspects in the design of an operational concept. These questions will be answered in collaboration with *Schiphol Airport*, who has performed a market consultation in new airport security configurations and industry experience to suggest specific policies of interest. Additionally, the researchers is forced to think about the integration of the operational concepts into the simulator.

2. How will the performance of each airport security checkpoint concept be evaluated?
   (a) What key performance indicators are going to be used?
   (b) How will the results be verified/validated?
      i. What input data will be needed to verify concepts?
   (c) How will the results be compared?

   The second research question focuses on the evaluation of a security checkpoint concept. Sub-questions are used to aid in determining how operational concepts can be analyzed and statistically relevant. These questions will partially be answered by the key performance indicators that *Schiphol Airport* is using, but also allow the researcher to think about new types of analysis.

3. How are agents going to learn to design/change an airport security checkpoint operational concept?
   (a) What reward functions/objectives are going to be used?
      i. How can rewards be shaped to not have a sparse reward system?
      ii. How to prevent an agent from being stuck in a local minimum?
      iii. Will the agent have a single or multiple objectives? If multiple objectives are used, how can they be evaluated?
   (b) Will the agent use on-policy or off-policy methods?
   (c) Will the agent use model-based or model-free algorithms?
   (d) Will the agent have partial or full observability of the environment?
   (e) Will there be one agent or multiple agents learning how to design a checkpoint?
   (f) How can this be implemented in the simulator?

The third research question targets the architecture of the reinforcement learning agent. Sub-questions are asked to guide the researcher in the proper direction when making decisions on how to structure an agent. Furthermore, the researcher is asked on the implementation of learning agents into the simulator.

4. How will the performance of an agent be quantified?
   (a) What means of quantification are there?
   (b) Will the performance be based on the final checkpoint design or the agents achieved reward?

The fourth research questions addresses the evaluation of agents performance. Sub-questions aim at defining what quantification methods there are and if agent performance will be based on its final design or its ability to learn.

7.3. Work Packages

7.3.1. Initial Phase (12 Weeks)

WP. 01 **Implement airport security configurations** (4weeks)
This work package discusses the changes that have to be accomplished in the simulator in order to implement novel airport security configurations. The approach should be modular such that many different configurations can be explored. Weekly meetings are scheduled with *Schiphol Airport* to keep track of progress.

WP. 01A **Creating a simple modular security checkpoint** (1 week)
This sub package assigns time to develop a simple security checkpoint within the AATOM simulator. Although a security checkpoint is already implemented, it does not have the ability to model different configurations. Thus, new classes/objects must be introduced into the AATOM code to achieve modularity. In order to know scope which structural changes in the code are necessary, a meeting will be scheduled with both the creator of the AATOM simulator, Stef Janssen, and a researcher at the TU Delft using the simulator, Adin Mekic. The simple security model will compromise of a divest belt, xray belt and a reclaim belt. It is important to note that the movement of luggage between these system is of primary importance. The belts and the flow of luggage must all be incorporated into the simulator.

WP. 01B **Extend the system to include more security assets** (1 week)
This sub package allocates time to extend the basic security system developed in the previous sub package. By incorporating more security assets into the xray system of the AATOM simulator, such as a decision belt, it possible to increase the complexity of the security configuration.

WP. 01C **Change simulation environment of security checkpoint while simulating**(2 week)
In the AATOM simulator, the environment is currently generated at the start of the simulation and then agents proceed to interact with the environment. However, it would be beneficial to change the environment during the simulation such that assets can be added when needed. For example, the number of divest positions should be adjustable such that during the simulation an extra position can be added if required. In order to know which assets should be made changeable during the simulation, they have to be defined first.

WP. 02 **Define a research scope** (1 week)
This work package designates time to define the case study that will be used to analyze the operational concepts. To do this the following must be accomplished: define operational concepts; define testing conditions (i.e. how to compare different operational concepts); define objectives for agents to learn; define possible actions for agents to take.

WP. 03 **Implement learning agents** (6 weeks)
This work package assigns time to implement agents that have the ability to learn into the AATOM simulator. The approach should be modular such that agents can be reused for different learning criteria.

WP. 03A **Implement learning frameworks**(4 weeks)
This sub package allocates time for the implementation of learning frameworks. By using the basic agent-environment interface (see Section 6.1.1), it becomes apparent which criteria must be defined in order to establish a framework for learning. The observability of an agent must be defined which allow it to determine what state it is currently in. The possible actions an agent can take in a specific state must be defined. The resulting state and reward from a certain action must implemented in the code. Reward shaping is of paramount importance such that an agent converges to a good solution and does not get stuck in a local minimum.

WP. 03B **Analyze learning behaviour**(2 weeks)
As with all learning, it is uncertain what results might yield for the case study. Therefore, this sub package designates time to analyze the preliminary results of learning agents. For example, agents might learn something completely unintended in the simulation. While this results might be interesting, it is very probable that they are not beneficial to the final result. Thus, by taking the time to analyze the result this will help in improving the learning experience of the agent.

WP. 04 **Prepare for midterm meeting**(1 Week)
This work package presents the workload and assignments needed to prepare for the midterm meeting. At the meeting, the case study for this thesis will be presented along with progress made during the initial phase. Presentation will be in the form of a PowerPoint, and multiple tries will be performed during the week. Furthermore, answers should be prepared for specific questions.

7.3.2. Final Phase (12 weeks)

WP. 05 **Implement midterm meeting comments**(1 week)
This work package allocates time to adjust implementation done in the initial phase and the case study depending on the feedback of the midterm meeting.

WP. 06 **Execute case studies**(2 weeks)
This work package dedicates time to executing the case studies defined in previous work packages. Simulations will be run using the new implementations in the AATOM model. Since many simulation runs are needed to train an agent, the simulations will be conducted on the Air Transport Operation cluster. Therefore, learning how to use and book the cluster is of primary importance in this work package.

WP. 07 **Analyze simulation results**(1 week)
This work package assigns time to analyze the simulation results. Analysis will be twofold: the learning behaviour of the agent will be analyzed and the performance of the security concept will be evaluated. Additional code must be written in order to analyze the results. The code shall be general such that in can read a generic log file from the AATOM simulator.

WP. 08 **Execute case studies**(2 weeks)
This work package sets aside additional time to execute further case studies. It was decided to execute case studies iteratively because unforeseen problems could potentially arise in the first batch. For example, an agent may not learn as intended or get stuck in a local minimum. Hence, it is beneficial to have another session where case studies can be repeated.

WP. 09 **Analyze simulation results**(1 week)
This work package allocates time to analyze the new simulation results. It will primarily rely on the code done in the first analysis of the simulation results.

WP. 10 **Execute case studies**(2 weeks)
This work package assigns additional time to execute further case studies. This is the third and final iteration to execute case studies.

WP. 11 **Analyze simulation results**(1 week)
This work package designates time to analyze the new simulation results. It will primarily rely on the code done in the first analysis of the simulation results.

WP. 12 **Write draft version of thesis**(1 week)
This work package sets aside time to write the thesis. It is important to note that the thesis

paper will be worked on during the other work packages as well. Special focus will be put on the methodology and discussion of results.

WP. 13 **Prepare for green light meeting**(1 week)
In order to attend the green light meeting, the draft version of the thesis must be submitted. This work package presents the workload and assignments needed to prepare for the green light meeting. At the meeting, the final results of this thesis will be presented along with progress made during the final phase. Presentation will be in the form of a PowerPoint, and multiple tries will be performed during the week. Furthermore, answers should be prepared for specific questions.

7.3.3. Final Thesis and Defence Phase (4 weeks)

WP. 13 **Implement green light meeting comments** (1 week)
This work package allocates time to adjust analysis done in the final phase depending on the feedback of the green light meeting.

WP. 14 **Finish up thesis** (1 week)
This work package assigns time to finalize the thesis. Special focus will be put on formatting, spelling and layout of the thesis.

WP. 15 **Prepare for defence** (2 weeks)
The last work package sets aside time to prepare for the thesis defence. A presentation should be created/prepared that discusses the methodology and key findings of the research. Time will be used to rehearse the presentation and prepare for any possible questions that can be asked during the defense.

III

Supporting Work

The following appendices were written in collaboration with another MSc Aerospace Engineer student, Didier van der Horst, who is also doing his master thesis with Amsterdam Airport Schiphol (see Horst (2021) [25]). Together, we developed and extended the original AATOM simulator from Janssen et al. (2019) [30] to improve the security checkpoint process. Furthermore, we used the extended AATOM simulator to simulate several case studies for Amsterdam Airport Schiphol.

Appendix A

AATOM Simulator

This appendix provides a detailed description of the Agent Based Airport Terminal Operations Model (AATOM). AATOM was initially developed by Janssen et al. (2019). During this research the architecture was updated and improved to facilitate the implementation of a modular security checkpoint. The necessary architecture changes, along with the improved/added classes are presented in this appendix.

This appendix is structured in a top-down manner. Initially the highest level of hierarchy is covered, namely the procedure that initializes the simulation by 'building' its components (covered in Appendix A.3) and the procedure that dictates the start conditions, end conditions and the step procedure in the simulation (covered in Appendix A.4).

Then the main simulation components are covered, namely the environment and the agent. For the environment, see Appendix A.5, the approach used to generate and update the environment is presented. In Appendix A.6, covering the agents, a detailed description of the agent architecture, along with a detailed description of each of the activities that an agent can perform is provided.

A.1. Assumptions

This section presents the assumptions used in the agent-based model. They are categorized into three parts: passenger, operator and security checkpoint.

Passenger Assumptions

- **Ass**$_{Pax.}$ **01** All passenger activities are performed strictly sequentially. For example, passengers do not go back to divest another item once they have completed the divest activity.
- **Ass**$_{Pax.}$ **02** All luggage items have the same divest and reclaim distributions.
- **Ass**$_{Pax.}$ **03** All passengers go through the SSc (adopted from Janssen et al. (2019)).
- **Ass**$_{Pax.}$ **04** Passengers always automatically stop inside the SSc scanner.
- **Ass**$_{Pax.}$ **05** Passengers are never denied access to the restricted area (adopted from Janssen et al. (2019)).
- **Ass**$_{Pax.}$ **06** Passengers can start the reclaim as soon as their luggage item is on the belt, and they are at a reclaim location. I.e. passengers do not have to wait for the tray to be next to them.

Operator Assumptions

- **Ass**$_{Ops.}$ **01** The rotation of operators in the security checkpoint is neglected.
- **Ass**$_{Ops.}$ **02** Operators are always performing an activity, i.e. they do not slack.
- **Ass**$_{Ops.}$ **03** Operator communication failures are captured inside agent attribute distributions.
- **Ass**$_{Ops.}$ **04** Recheck operators only check luggage once it reaches the end of the reject belt.
- **Ass**$_{Ops.}$ **05** Recheck operators check luggage in the order that it arrives at the recheck.
- **Ass**$_{Ops.}$ **06** Divest operators are not modelled.

Security Checkpoint Assumptions

- **Ass**$_{Pax.}$ **01** Luggage is transferred instantaneously from divest belt to CT belt.
- **Ass**$_{Pax.}$ **02** Luggage is transferred instantaneously from decision belt to reclaim belt.
- **Ass**$_{Pax.}$ **03** "Stop the check" is not simulated. A scenario where the entire security lanes freezes due to a suspicious/dangerous item.

Ass$_{Pax.}$ **04** Luggage always acts as expected, i.e. it does not fall out of trays which can slow the CT process.

Ass$_{Pax.}$ **05** SSc rescans are not modelled, these are captured in the scan time.

A.2. Overview

At the highest level of hierachy, the *Simulator* dictates the main components of the simulation. In the *Simulator*, the environment is constructed by means of the *Map*. Anything that is added to the *Map*, is known as a *MapComponent*. Initially the user constructs the *Map* within the user interface by adding *MapComponents*. The procedure of generating agents is specified in the *AgentGenerator*, and the stop condition of the simulation is defined by the *EndingCondition*. Schematically a class diagram is depicted in Figure A.1.

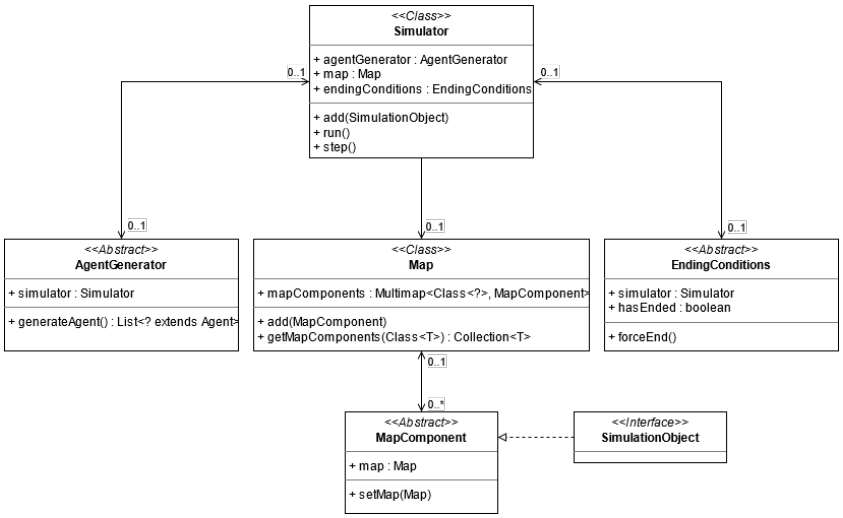


Figure A.1: UML Diagram of Simulator

A.3. User Interface

The user interface constructs the simulator such that a simulation can be run. This is done by initially instantiating a $Simulator$ object. Then, $MapComponents$ are created and added to the $Simulator$. The pseudo code is depicted in Algorithm 19.

Algorithm 19 Main File

Input airport layout A_{layout}, passenger attributes $P_{Att.}$, operator attributes $O_{Att.}$, security checkpoint characteristics $C_{char.}$, ending conditions $End_{cond.}$

for $episode$ in $episodes$ **do**
    $Sim \leftarrow$ new SIMULATOR($End_{cond.}$, new AGENTBASEGENERATOR($P_{Att.}$))
    Sim.ADD(new WALLS(A_{layout}))
    Sim.ADD(new ENTRANCEAREAS(A_{layout}))
    Sim.ADD(new CHECKIN(A_{layout}))
    Sim.ADD(new SECURITYCHECKPOINT(A_{layout}, $O_{Att.}$, $C_{char.}$))
    Sim.ADD(new GATEAREA(A_{layout}))
    Sim.RUN()
end for

A.4. Simulator

The $Simulator$ class can be regarded as highest hierachy in AATOM. It interfaces the end conditions and updates the simulation. To achieve this, three methods are highlighted. First the $add()$ method, which is used to add objects to the simulator. Second, the $run()$ method, which is used to start and end the simulation.

Third the $step()$ method, which updates the simulator. During this update, the $AgentGenerator$ checks if new agents must be generated. Furthermore, $step()$ updates all $MapComponents$ that are identified as $directlyUpdatable$. The pseudo-code for the $add()$, $run()$, and $step()$ methods are depicted in Algorithm 20, Algorithm 21 and Algorithm 22 respectively.

Simulator Attributes

Sim$_{Att.}$ **01** *Agent Generator*
DESCRIPTION: Generates passengers in the entrance area if generating conditions are satisfied.
INITIALIZATION: A passenger agent is generated when there are less than 20 passengers in the banklining of the security checkpoint. This ensure continuous demand for the security checkpoint.

Sim$_{Att.}$ **02** *Map*
DESCRIPTION: Contains all physical elements of the model.
INITIALIZATION: According to airport layout.

Map$_{Att.}$ **01** *Map Components*
DESCRIPTION: A collection of all map components on the map.
INITIALIZATION: According to airport layout.

Sim$_{Att.}$ **03** *Ending Conditions*
DESCRIPTION: The ending conditions of a simulation.
INITIALIZATION: Simulations ends after a fixed number of steps.

Algorithm 20 Simulator Method: $Sim.\text{ADD}(simulationObjects)$

```
function Sim.ADD(simulationObjects)
    for obj in simulationObjects do
        if obj instance of Concept.class then          ▷ Concept explained in Appendix A.5.2.
            c ← (Concept) obj
            for mapComponent in c.GETMAPCOMPONENTS() do
                Map.ADD(mapComponent)
            end for
        end if

        if obj instance of MapComponent.class then
            Map.ADD(obj)
        end if

        if obj instance of Agent.class then
            a ← (Agent) obj
            a.INIT()
        end if
    end for
end function
```

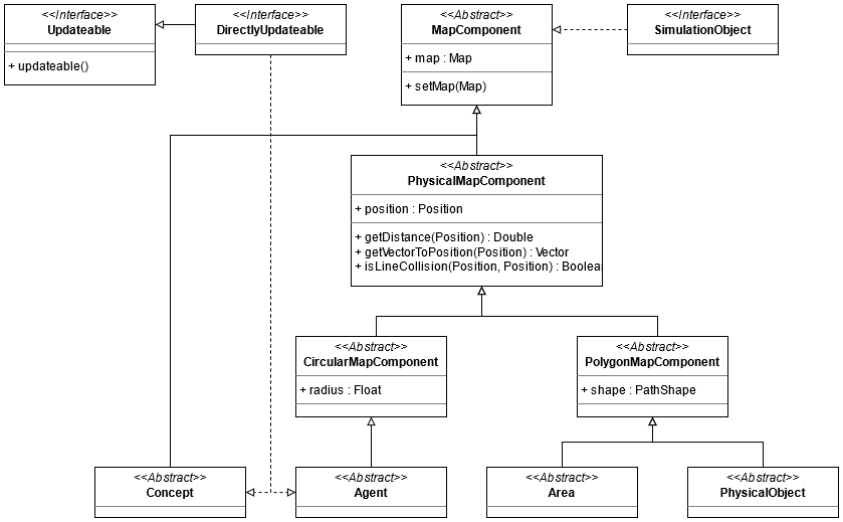


Figure A.2: UML Diagram of MapComponent.

Algorithm 21 Simulator Method: $Sim.\text{RUN}()$

function $Sim.\text{RUN}$
  $running \leftarrow$ True
  while $End_{cond.}$ not satisfied **do**
    $time \leftarrow 0$
    if $running$ **then**
      $time \leftarrow \text{STEP}()$
    end if
  end while
  $\text{LOGOUTPUT}()$
end function

Algorithm 22 Simulator Method: $Sim.\text{STEP}()$

function $Sim.\text{STEP}()$
  $agents \leftarrow AgentGenerator.\text{GENERATEAGENT}(P_{Att.})$
  $Sim.\text{ADD}(agents)$

  for $directlyUpdateable \leftarrow Map.\text{GETMAPCOMPONENTS}(directlyUpdateable.\text{class})$ **do**
    $directlyUpdateable.\text{UPDATE}()$
  end for
end function

A.5. Environment

The $Simulator$ contains a Map. The Map can be regarded as the environment which consists of different $MapComponents$. The four main $MapComponents$ that we highlight are the $Agents$, $PhysicalObjects$, $Areas$ and $Concepts$. A schematic class diagram of the $MapComponent$ is shown in Figure A.2. It can be seen that of these four $MapComponents$, the $Agent$ and $Concept$ are classes that are also $DirectlyUpdatable$, indicating that they are updated each step by the $Simulator$ class.

A.5.1. Physical Objects

One of the four main $MapComponents$ is the $PhysicalObject$. The $PhysicalObject$ is an object that has a physical representation on the Map and is also visualized in the user-interface. The classification of $PhysicalObjects$ is presented in the class diagram shown in Figure A.3. $Sensors$ are used to make observations that agents would not be able to do. To provide modularity in constructing security checkpoint

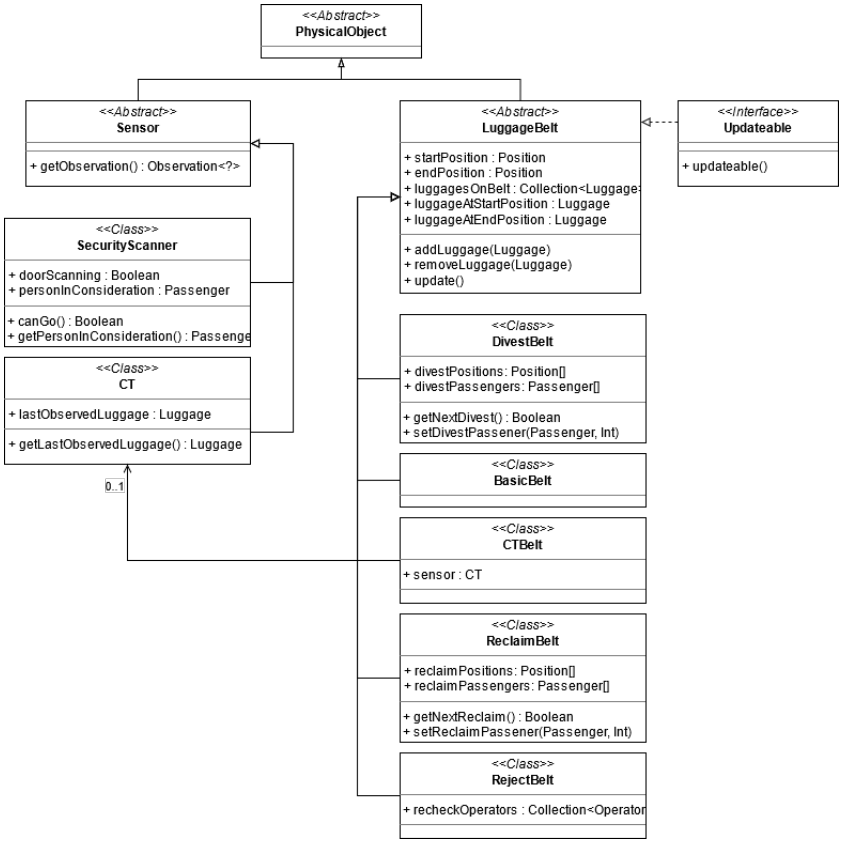


Figure A.3: UML Diagram of PhysicalObject.

configurations, the $LuggageBelt$ class was implemented. $LuggagBelts$ are able to transport the luggage of passenger agents. Users can construct a modular checkpoint by defining luggage belts individually.

$LuggageBelt$ attributes are presented below. The update of the luggage over the luggage belt is shown in Algorithm 23.

Luggage Belts

Luggage Belt Attributes

Belt$_{Att.}$ **01** *Start Position*
DESCRIPTION: A position indicating the start of the belt.
INITIALIZATION: According to airport layout.

Belt$_{Att.}$ **02** *End Position*
DESCRIPTION: A position indicating the end of the belt.
INITIALIZATION: According to airport layout.

Belt$_{Att.}$ **03** *Luggages On Belt*
DESCRIPTION: A collection of luggages that are on the belt.
INITIALIZATION: Empty.

Belt$_{Att.}$ **04** *Luggage at Start Position*
DESCRIPTION: The luggage that is at the end position of the belt.
INITIALIZATION: Null.

Belt$_{Att.}$ **05** *Luggage at End Position*
DESCRIPTION: The luggage that that is at the start position of the belt.
INITIALIZATION: Null.

Algorithm 23 Luggage Belt Method: $Belt$.UPDATE()

```
function Belt.UPDATE()
    for luggage in luggagesOnBelt do
        if luggage not at endPosition then
            if GETLUGGAGEINFRONT() = null then
                move luggage
            end if
        else
            SETLUGGAGEATENDPOSITION(luggage)
        end if
    end for
end function
```

A.5.2. Concepts

When a sub-process is added to the Map, a $Concept$ must be implemented. A $Concept$ describes the relationship between different $mapComponents$ and $Agents$ in a specific airport sub-process. E.g., in a larger sub-process such as the security checkpoint, the interaction between different $LuggageBelts$ must be defined.

Checkpoints

As this research focuses on security checkpoints, a $Concept$ was created for the security checkpoint. This is implemented in the $Checkpoint$ class, which extends the $Concept$ class. The $Checkpoint$ therefore provides the relationship between all $PhysicalObjects$ and $Agents$ that belong to the security checkpoint. These relationships include (1) the order in which luggage flows over the $Luggagebelts$, (2) the transfer of luggage between luggage belts, and (3) whether passengers and luggage items are suspicious and need corresponding additional checks.

The relevant attributes of the $Checkpoint$ are listed below. Each iteration the $Checkpoint$ is updated, the pseudo-code is presented in Algorithm 24. The pseudo-code for luggage transfers between belt is presented in Algorithm 25.

Checkpoint Attributes

Check$_{Att.}$ 01 *Luggage Flow*
DESCRIPTION: A mapping from each luggage belt to a collection of luggage belts that it is able to transfer to.
INITIALIZATION: According to airport layout.

Check$_{Att.}$ 02 *Not Checked Luggages*
DESCRIPTION: A collection of luggages that have been scanned by the CT sensor but not yet checked by a security operator.
INITIALIZATION: Empty.

Check$_{Att.}$ 03 *Luggages Rechecked*
DESCRIPTION: A collection of luggages that have to be rechecked.
INITIALIZATION: Empty.

Check$_{Att.}$ 04 *Passengers To Scan*
DESCRIPTION: A collection of passengers that have to be scanned.
INITIALIZATION: Empty.

Check$_{Att.}$ 05 *Passengers To Frisk*
DESCRIPTION: A collection of passengers that have to be frisked.
INITIALIZATION: Empty.

Algorithm 24 Checkpoint Method: $Checkpoint$.UPDATE()

```
function Checkpoint.UPDATE()
    beltsToTransfer ← []
    for belt in luggageBelts do
        belt.UPDATE()
        if belt instance of CTBelt.class then
            belt ← (CTBelt) belt
            luggage, observation ← belt.GETCTSENSOR().GETOBSERVATION()
            luggageNotChecked.ADD(luggage)
        end if
        if belt.GETLUGGAGEATENDPOSITION() != null then
            beltsToTransfer.ADD(belt)
        end if
    end for

    beltsToTransfer ← sort beltsToTransfer by time waiting for transfer
    for belt in beltsToTransfer do
        TRANSFERLUGGAGE(belt)
    end for
end function
```

Algorithm 25 Checkpoint Method: $Checkpoint$.TRANSFERLUGGAGE($belt$)

```
function Checkpoint.TRANSFERLUGGAGE(belt)
    luggage ← belt.GETLUGGAGEATENDPOSITION()
    targetBelts ← luggageFlow.GET(belt)
    for targetBelt in targetBelts do
        if targetBelt.ISFREE() then
            if targetBelt instance of ReclaimBelt.class then
                if luggage != suspicious and luggage can divert then
                    transferLogic ← True
                end if
            else if targetBelt instance of RejectBelt.class then
                if luggage = suspicious and luggage can divert then
                    transferLogic ← True
                end if
            else
                transferLogic ← True
            end if

            if transferLogic = True then
                belt.REMOVELUGGAGE(luggage)
                targetBelt.ADDLUGGAGE(luggage)
                break
            end if
        end if
    end for
end function
```

A.6. Agent

This section will describe the architecture of the agent in more detail. $Agents$ use the original AATOM agent architecture. This architecture is depicted in the class diagram in Figure A.4. As per original AATOM architecture, an $Agents$ is initialized with three models: (1) the $StrategicModel$, (2) the $TacticalModel$ and (3) the $OperationalModel$. Each of these models has several internal modules. When an agent is

generated at the airport terminal or security checkpoint, it initializes with goals that it wants to accomplish ($GoalModule$). Through reasoning, an agent generates a plan (e.g. a list of activities) to achieve these goals ($PlanningModule$). Based on the strategy, an agent will navigate ($NavigationModule$) through the environment and execute activities ($ActivityModule$). Agents perform activities by observing ($PerceptionModule$) and executing actions ($ActuationModule$). It is important to note agents continuously maintain their beliefs and goals, thereby potentially resulting in new plans. Furthermore, agents are only able to perform one activity at a time. The main sequence of operations followed by an agent is: observation → perception → interpretation → reasoning → activity control → actuation → action.

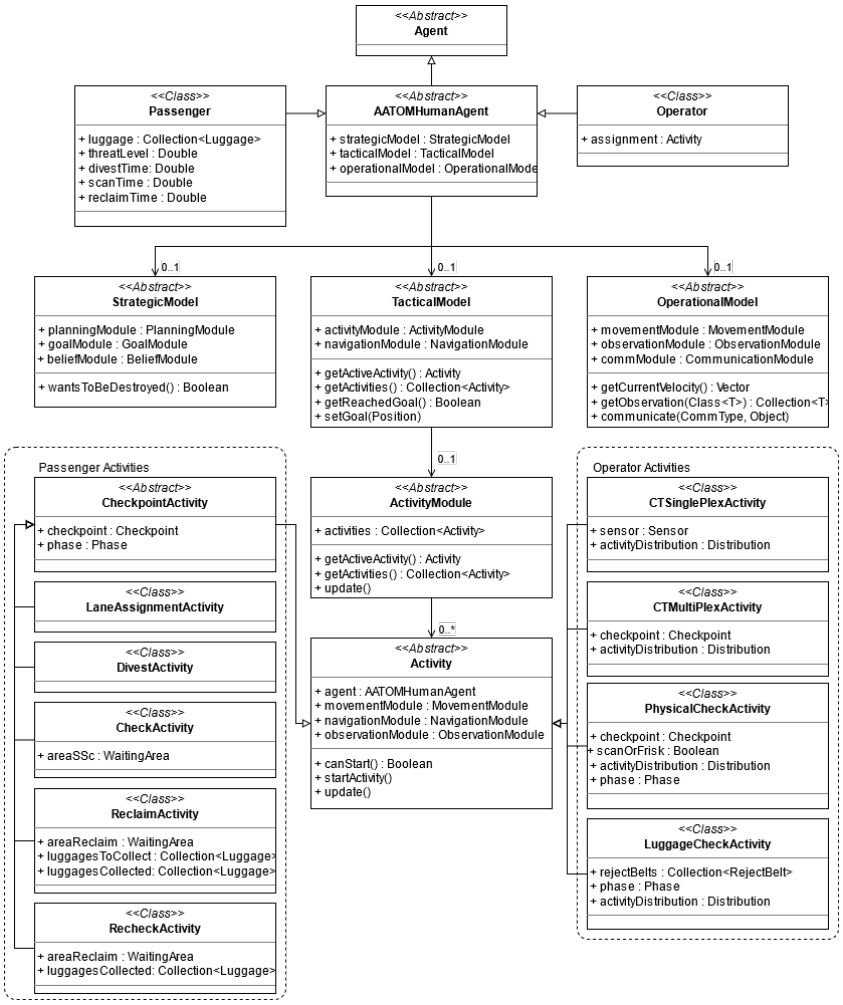


Figure A.4: UML Diagram of an Agent.

The modules that were improved such that agents are compatible with the new security checkpoint are the $GoalModule$, $PlanningModule$ and $ActivityModule$. These are also the modules that are covered in more detail in this appendix. The goal module is covered in Appendix A.6.1, the planning module in Appendix A.6.2, and the activity module in Appendix A.6.3.

For all agents the initialisation and updates are identical. The initialisation of the agents is dictated by the $AgentGenerator$. All internal modules of the agent are initialises simultaneously with the initialisation of the $Agents$. The pseudo-code for the $initialisation()$ of an agent is presented in Algorithm 26. An $Agent$ is $DirectlyUpdatable$. Therefore, every $Simulator$ step updates the $Agents$. This update also results in the update of the internal modules of the $Agents$. The pseudo-code for the $update()$ of an agent is presented in Algorithm 27.

Algorithm 26 Passenger Method: $Agent$.INIT()

```
function Agent.INIT()
    goalModule.INIT()                                  ▷ Initializes goals.
    planningModule.INIT(goalModule)                    ▷ Initializes planning based on goals.
end function
```

Algorithm 27 Agent Method: $Agent$.UPDATE()

```
function Agent.UPDATE()
    goalModule.UPDATE()                                ▷ Goals can change over the simulation.
    planningModule.UPDATE()                            ▷ Plans must match the new goals.
    beliefModule.UPDATE()
    activityModule.UPDATE()                            ▷ Activities must match the new plan.
    navigationModule.UPDATE()                          ▷ Navigation must match the new plan.
    movementModule.UPDATE()                            ▷ Movement dictated by destination.
end function
```

A.6.1. Goal Module

The $GoalModule$ is part of the agents $StrategicModel$. It manages the goals that an agent has. The user defines the goals with which an agents is initialised. In context of this research, only the goals required to successfully complete the security checkpoint were added. These are the check-in goal and the checkpoint goal. The pseudo-code for the initialisation of the $GoalModule$ is presented in Algorithm 28.

Goal Module Attributes

GoalMod$_{Att.}$ **01** *Goals*
DESCRIPTION: A collection of goals.
INITIALIZATION: Defined by the user.

Algorithm 28 Goal Module Method: $GoalModule$.INIT()

```
function GoalModule.INIT()
    if not checkedIn then
        goals.ADD(GOAL(checkInActivity))
    end if
    goals.ADD(GOAL(checkpointActivity))
end function
```

A.6.2. Planning Module

The $PlanningModule$ is part of the agent's $StrategicModel$. It manages the planning of activities required to satisfy the agent's goals. This requires the user to specify which activities need to be sequentially executed in order to satisfy a specific goal. Once the user has specified this order of sequential activities, the $PlanningModule$ is initialised by adding all activities that belong to a certain goal to the planning. In the pseudo-code, depicted in Algorithm 29, this is represented by the method $addActivitiesFromType(activity_i)$ which adds all activities required to satisfy $activity_i$ in the correct order to the agent's planning.

The $PlanningModule$ of the agent can be accessed at any moment in the simulation. By accessing the $PlanningModule$, the next activity that an agents needs to execute can be determined. Within the improved AATOM simulator, additional plans were implemented in order to fulfil the security checkpoint goal.

Planning Module Attributes

PlanMod$_{Att.}$ **01** *Planning*
DESCRIPTION: A collection of activities to perform.
INITIALIZATION: Populated depending on the goals.

Algorithm 29 Planning Module Method: $PlanningModule.\text{INIT}(goalModule)$

```
function PlanningModule.INIT(goalModule)
    for goal in goalModule.GETGOALS() do
        if goal instance of checkInActivity.class then
            planning.ADDACTIVITIESFROMTYPE(checkInActivity.class)
        else if goal instance of checkpointActivity.class then
            planning.ADDACTIVITIESFROMTYPE(checkpointActivity.class)
        end if
    end for
end function
```

A.6.3. Activity Module

The $ActivityModule$ is part of the $TacticalModel$. It defines the logic required to execute a specific activity. For every goal, a plan is derived. For this plan a set of activities are allocated to satisfy the plan. In context of this research, specifically the activities required for the security checkpoint were added to the existing AATOM simulator. This section will mainly discuss these new activities and present pseudo-code for each of these activities, both for the passenger agent and the operator agent.

A passenger agent can have the goal to pass the security checkpoint. To succeed, a plan is derived consisting of the following activities:

1. Lane Assignment activity
2. Divest activity
3. Check activity
4. Reclaim activity
5. Recheck activity

The operator agent can have the goal to perform a specific assignment, this assignment dictates the planning of that operator agent and hence the activity which is executed. The different possible activities are:

1. CT SinglePlex activity
2. CT MultiPlex activity
3. Physical Check activity
4. Luggage Recheck activity

As agents are $DirectlyUpdatable$, so are its models. Hence, the $ActivityModule$ is directly updatable which implies that its $update()$ method is updated every step by the $Simulator$. The $ActivityModule$ thus updates all the activities that are to be executed. It first checks if an activity is in progress, if not it starts the activity. If it is in progress the activity is updated. Refer to Algorithm 30 for the pseudo-code.

The general procedure for each activity is identical. Every activity has the following methods:

- $isInProgress()$: Defines if the activity is in progress
- $canStart()$: Defines if the activity can start
- $update()$: Defines the procedure of the respective activity

In the remainder of this section the $canStart()$ and the $update()$ function for each of the activities within the security checkpoint will be discussed. To understand the corresponding pseudo-code, it should be realised that activities are performed by an agents (they are linked to agents by the $ActivityModel$). Therefore, whenever a position is referenced, the agent position is implied. Furthermore, when an observation is done, it implies that the agent is observing.

Algorithm 30 Activity Module Method: $ActivityModule$.UPDATE()

```
function ActivityModule.UPDATE()
    activity ← planningModule.GETNEXTACTIVITY()
    if activity.ISINPROGRESS() then
        activity.UPDATE()
        return
    end if
    if activity.CANSTART() then
        activity.STARTACTIVITY()
        return
    end if
    activity.GOTOACTIVITY()
end function
```

Lane Assignment Activity

A passenger activity. Starts when a passenger reaches the end position of the banklining of the queue. A passenger observes the occupancy of the security lanes, decides which lane to go to based on the minimum number of passengers, and then goes to the entry position of that security lane. This is presented as pseudo-code in Algorithm 31 and Algorithm 32.

Algorithm 31 Lane Assignment Activity Method: $LaneAssignmentActivity$.CANSTART()

```
function LaneAssignmentActivity.CANSTART()
    if position = banklining.GETEXITPOSITION() then
        checkpoints ← OBSERVEALL(Checkpoint.class)
        bestCheckpoint ← SELECTLEASTOCCUPIED(checkpoints)

        if bestCheckpoint != null then
            navigationModule.SETGOAL(bestCheckpoint.GETWAITINGAREADIVEST())
            return True
        end if
    end if
    return False
end function
```

Algorithm 32 Lane Assignment Activity Method: $LaneAssignmentActivity$.UPDATE()

```
function LaneAssignmentActivity.UPDATE()
    if navigationModule.GETREACHEDGOAL() then    ▷ If passenger has reached waiting area divest.
        ENDACTIVITY()
    end if
end function
```

Divest Activity

A passenger activity. Starts when a passenger reaches the entry position of the security lane. A passengers waits until a divest position is assigned to them, then goes to the divest position and drops their luggage onto the belt. The total time to drop a luggage item is determined by the agents attributes. Once all luggage has been divested, the passenger continues to the entry position of the security scanner. The $canStart()$ and $update()$ methods are presented in Algorithm 33 and Algorithm 34 respectively. The specific methods called within the $update$ are illustrated in Algorithm 35, Algorithm 36 and Algorithm 37.

Divest Activity Attributes

DivestAct$_{Att.}$ **01** *Phase*
DESCRIPTION: Indicates what phase of the activity an agent is in.
INITIALIZATION: 0.

DivestAct$_{Att.}$ **02** *Checkpoint*
DESCRIPTION: Indicates the checkpoint the agent is in.
INITIALIZATION: The best checkpoint from the lane assignment activity.

Algorithm 33 Divest Activity Method: $DivestActivity$.canStart()

```
function DivestActivity.canStart()
    if navigationModule.getReachedGoal() then    ▷ If passenger has reached waiting area divest.
        return True
    end if
    return False
end function
```

Algorithm 34 Divest Activity Method: $DivestActivity$.update()

```
function DivestActivity.update()
    goToDivest()
    stopAtDivest()
    divestLuggage()
end function
```

Algorithm 35 Divest Activity Method: $DivestActivity$.goToDivest()

```
function DivestActivity.goToDivest()
    if phase = 0 then
        for divestBelt in observeAll(divestBelt.class in checkpoint) do
            divestPositions ← observeAll(divestPosition.class in divestBelt)
            divestPositions ← observe(pos if pos.unoccupied() for pos in divestPositions)
            if not divestPositions = Empty then
                divestPosition ← divestPositions.selectFirst()
                navigationModule.setGoal(divestPosition)
                phase = phase + 1
                break
            end if
        end for
    end if
end function
```

Algorithm 36 Divest Activity Method: $DivestActivity$.stopAtDivest()

```
function DivestActivity.stopAtDivest()
    if phase = 1 and position = divestPosition then
        movementModule.setStopOrder(passenger.getDivestTime())
        phase = phase + 1
    end if
end function
```

Algorithm 37 Divest Activity Method: $DivestActivity$.DIVESTLUGGAGE()

```
function DivestActivity.DIVESTLUGGAGE()
    if phase = 2 and position = divestPosition and not movementModule.ISINSTOPORDER() then
        for luggage in GETLUGGAGES() do          ▷ Gets the collection of luggages of a passenger.
            if divestBelt.ISFREE() and luggage.STILLONPASSENGER() then
                ADD(luggage to divestBelt)
            end if
        end for

        if getLuggages() are all divested then
            areasSSc ←OBSERVEALL(WaitingAreaSSc.class in checkpoint)
            areaSSc ←SELECTLEASTOCCUPIED(areas)
            navigationModule.SETGOAL(areaSSc)
            ENDACTIVITY()
        end if
    end if
end function
```

Check Activity

A passenger activity. Starts when a passenger reaches the entry position of the security scanner. A passenger observes if they can go into the security scanner and then proceeds to go to the scan point if it is unoccupied. When passenger reaches the scan point the passenger is added to the lists of passengers to scan. Operateor agents keep track of this list. An operator performs the passenger screen activity, where after the operator either notifies that the passenger is scanned. A passenger completes their activity when they have been scanned and are not suspicious. The $canStart()$ and $update()$ methods are presented in Algorithm 38 and Algorithm 39 respectively. The methods called within the $update$ are illustrated in Algorithm 40 and Algorithm 41.

Check Activity Attributes

CheckAct$_{Att.}$ **01** *Phase*
DESCRIPTION: Indicates what phase of the activity an agent is in.
INITIALIZATION: 0.

CheckAct$_{Att.}$ **02** *Checkpoint*
DESCRIPTION: Indicates the checkpoint the agent is in.
INITIALIZATION: The best checkpoint from the lane assignment activity.

CheckAct$_{Att.}$ **03** *Area SSc*
DESCRIPTION: Indicates the waiting area of the security scanner.
INITIALIZATION: The security scanner area observed from the divest activity.

Algorithm 38 Check Activity Method: $CheckActivity$.CANSTART()

```
function CheckActivity.CANSTART()
    if navigationModule.GETREACHEDGOAL() then      ▷ If passenger has reached waiting area SSc.
        SSc ←OBSERVE(SSc.CLASS IN areaSSc)
        return True
    end if
    return False
end function
```

Algorithm 39 Check Activity Method: $CheckActivity$.UPDATE()

```
function CheckActivity.UPDATE()
    GOTOSSC()
    SCANANDFRISK()
end function
```

Algorithm 40 Check Activity Method: $CheckActivity$.GOTOSSC()

```
function CheckActivity.GOTOSSC()
    if phase = 0 and SSc.CANGO() then
        checkpoint.GETPASSENGERSTOSCAN().ADD(passenger)
        navigationModule.SETGOAL(SSc)
        phase = phase + 1
    end if
end function
```

Algorithm 41 Check Activity Method: $CheckActivity$.SCANANDFRISK()

```
function CheckActivity.SCANANDFRISK()
    if phase = 1 and ISSCANNED() and not ISSUSPICIOUS() then
        areaReclaim ←OBSERVE(waitingAreaReclaim.CLASS IN checkpoint)
        navigationModule.SETGOAL(areaReclaim)
        ENDACTIVITY()
    end if
end function
```

Reclaim Activity

A passenger activity. Starts when a passenger is not suspicious. A passenger goes to the reclaim area and waits for their luggage to be scanned. Next, the passenger observes whether or not their luggage is rejected. If all of their luggage has been rejected, then the activity ends immediately. By contrast, if a luggage has not been rejected, then the passenger will observe if a reclaim position is unoccupied. When a reclaim position is unoccupied, they proceed to go to that reclaim position and collect that luggage. If a passenger has another piece of luggage, the process restarts with the passenger observing if their luggage has been rejected or not. The total time to collect a luggage is dictated by the passenger attributes as described in the research paper. Once all luggage has been collected, the activity is completed. The $canStart()$ and $update()$ methods are presented in Algorithm 42 and Algorithm 43 respectively. The methods called within the $update()$ are illustrated in Algorithm 44, Algorithm 45, Algorithm 46 and Algorithm 47. In the pseudo-code, restarting the reclaim process is managed by the attribute *Phase* and *LuggageToCollect*. The phase determines in which part of the reclaim process the agent is. The *LuggageToCollect* is a collection of all luggage that are still to be collected.

Reclaim Activity Attributes

ReclaimAct$_{Att.}$ **01** *Phase*
DESCRIPTION: Indicates what phase of the activity an agent is in.
INITIALIZATION: 0.

ReclaimAct$_{Att.}$ **02** *Checkpoint*
DESCRIPTION: Indicates the checkpoint the agent is in.
INITIALIZATION: The best checkpoint from the lane assignment activity.

ReclaimAct$_{Att.}$ **03** *Area Reclaim*
DESCRIPTION: Indicates the waiting area of the reclaim.
INITIALIZATION: The reclaim area observed selected in the check activity.

ReclaimAct$_{Att.}$ **04** *Luggages To Collect*
DESCRIPTION: A collection of luggages that are available for collection.
INITIALIZATION: Empty.

ReclaimAct$_{Att.}$ **05** *Luggages Collected*
DESCRIPTION: A collection of luggages that have been collected.
INITIALIZATION: Empty.

Algorithm 42 Reclaim Activity Method: $ReclaimActivity$.CANSTART()

```
function ReclaimActivity.canStart()
    if navigationModule.getReachedGoal() then  ▷ If passenger has reached waiting area reclaim.
        return True
    end if
    return False
end function
```

Algorithm 43 Reclaim Activity Method: $ReclaimActivity$.UPDATE()

```
function ReclaimActivity.update()
    observeLuggage()
    goToReclaim()
    waitAtReclaim()
    reclaimLuggage()
end function
```

Algorithm 44 Reclaim Activity Method: $ReclaimActivity$.OBSERVELUGGAGE()

```
function ReclaimActivity.observeLuggage()
    if phase = 0 then
        luggagesToCollect ←observe(l if l.availableToCollect() for l in getLuggages())
        luggagesSuspicious ←observe(l if l.isSuspicious() for l in getLuggages())
        if getLuggages() - luggagesCollected = luggagesSuspicious then
            endActivity()                ▷ Proceed to recheck activity if only suspicious luggages left.
        end if
    end if
end function
```

Algorithm 45 Reclaim Activity Method: $ReclaimActivity$.GOTORECLAIM()

```
function ReclaimActivity.goToReclaim()
    if phase = 0 and not luggagesToCollect = Empty then
        reclaimBelt ← observe(reclaimBelt.class containing luggagesToCollect)
        reclaimPositions ← observeAll(reclaimPosition.class in reclaimBelt)
        reclaimPositions ← observe(pos if pos.unoccupied() for pos in reclaimPositions)
        if not reclaimPositions = Empty then
            reclaimPosition ← reclaimPositions.selectFirst()
            navigationModule.setGoal(reclaimPosition)
            phase = phase + 1
        end if
    end if
end function
```

Algorithm 46 Reclaim Activity Method: $ReclaimActivity$.waitAtReclaim()

```
function ReclaimActivity.waitAtReclaim()
    if phase = 1 and position = reclaimPosition then
        movementModule.setStopOrder(passenger.getReclaimTime())
        phase = phase + 1
    end if
end function
```

Algorithm 47 Reclaim Activity Method: $ReclaimActivity$.reclaimLuggage()

```
function ReclaimActivity.reclaimLuggage()
    if phase = 2 and not movementModule.isInStopOrder() then
        for luggage in luggagesToCollect do
            remove(luggage from reclaimBelt)
            luggagesCollected.add(luggage)          ▷ Can only remove luggage that was stopped for.
        end for
        if getLuggages()=luggagesCollected then
            endActivity()
        end if

        luggagesToCollect ←observe(l if l.availableToCollect() for l in getLuggages())
        luggagesSuspicious ←observe(l if l.isSuspicious() for l in getLuggages())
        if not luggagesToCollect = Empty then
            phase = phase − 1                       ▷ Go back and collect luggage.
        else
            navigationModule.setGoal(areaReclaim)   ▷ Free up reclaim position.
            phase = phase − 2                       ▷ Go back and wait for luggage to be scanned.
        end if
    end if
end function
```

Recheck Activity

A passenger activity. Starts when a passenger has rejected luggage. A passenger goes to the recheck area and observes if their luggage has been selected for examination by a security operator. Once selected, the passenger goes to the recheck position and waits for the security operator to perform the luggage recheck activity. When the recheck is finished, the passenger observes if they collected all of their luggage. If all luggage has been collected, the activity is completed. Otherwise, the passenger goes back to the recheck area and the process restarts. The $canStart()$ and $update()$ methods are presented in Algorithm 48 and Algorithm 49 respectively. The methods called within the $update()$ are illustrated in Algorithm 50, Algorithm 51, and Algorithm 52. Similar to the reclaim activity, restarting the recheck activity when not all luggage's have been collected is managed by moving back in the attribute *Phase*.

Recheck Activity Attributes

RecheckAct$_{Att.}$ **01** *Phase*
DESCRIPTION: Indicates what phase of the activity an agent is in.
INITIALIZATION: 0.

RecheckAct$_{Att.}$ **02** *Checkpoint*
DESCRIPTION: Indicates the checkpoint the agent is in.
INITIALIZATION: The best checkpoint from the lane assignment activity.

RecheckAct$_{Att.}$ **03** *Area Recheck*
DESCRIPTION: Indicates the waiting area of the recheck.
INITIALIZATION: The reclaim area observed selected in the check activity.

RecheckAct$_{Att.}$ **04** *Luggages Collected*
DESCRIPTION: A collection of luggages that have been collected.
INITIALIZATION: The luggages collected during the reclaim activity.

Algorithm 48 Recheck Activity Method: *RecheckActivity*.CANSTART()

```
function RecheckActivity.canStart()
    if getLuggages()=luggagesCollected then
        endActivity()
    else                                   ▷ Perform recheck because only suspicious luggage left.
        return True
    end if
end function
```

Algorithm 49 Recheck Activity Method: *RecheckActivity*.UPDATE()

```
function RecheckActivity.update()
    goToRecheck()
    waitAtRecheck()
    finishRecheck()
end function
```

Algorithm 50 Recheck Activity Method: *RecheckActivity*.GOTORECHECK()

```
function RecheckActivity.goToRecheck()
    if phase = 0 then
        recheckArea ←observe(recheckArea.class in checkpoint)
        navigationModule.setGoal(recheckArea)
        phase = phase + 1
    end if
end function
```

Algorithm 51 Recheck Activity Method: *RecheckActivity*.WAITATRECHECK()

```
function RecheckActivity.waitAtRecheck()
    if phase = 1 and navigationModule.getReachedGoal() then
        operators ← observeAll(operator.class assigned to recheck in checkpoint)
        operators ← observe(o if o.hasLuggage() for o in operators)
        if not operators = Empty then
            operators ← operators.selectFirst()
            luggage ← operator.getLuggage()
            navigationModule.setGoal(operator)
            phase = phase + 1
        end if
    end if
end function
```

Algorithm 52 Recheck Activity Method: $RecheckActivity$.FINISHRECHECK()

```
function RecheckActivity.finishRecheck()
    if phase = 2 and operator.gaveBackLuggage() then
        luggagesCollected.add(luggage)
        if getLuggages()=luggagesCollected then
            endActivity()
        else
            navigationModule.setGoal(recheckArea)
            phase = phase − 1
        end if
    end if
end function
```

CT SinglePlex Activity

An operator activity. The operator is allocated to one specific CT sensor. This is identified by the Operator attribute *Sensor*. Starts when there is cabin luggage to be checked. An operator selects the first luggage waiting to be checked. Then, they proceed to analyze whether or not the luggage should be rejected. The time to make a decision is determined by taking a random sample from the operator performance distribution. Decisions are based on the information obtained from the CT sensor. If the CT sensor raises an alarm, then the luggage is rejected and otherwise it is cleared. Once a decision has been made, the luggage is able to divert and the activity ends. The $canStart()$ and $update()$ methods are presented in Algorithm 53 and Algorithm 54 respectively.

CT SinglePlex Activity Attributes

SinglePlexAct$_{Att.}$ **01** *Sensor*
DESCRIPTION: The CT sensor that the operator is assigned to.
INITIALIZATION: According to airport layout.

SinglePlexAct$_{Att.}$ **02** *Activity Distribution*
DESCRIPTION: A time distribution representing the performance of the operator.
INITIALIZATION: According to operator attributes.

Algorithm 53 CT SinglePlex Activity Method: $CTSinglePlexActivity$.CANSTART()

```
function CTSinglePlexActivity.canStart()
    if not sensor.getNotCheckedLuggage() = Empty then
        luggage ← sensor.getNotCheckedLuggage().selectFirst()
        sensor.getNotCheckedLuggage().remove(luggage)
        decisionTime ← activityDistribution.sample()
        observation ← observe(luggage through sensor)
        if observation = alarm then
            luggage.setSuspicious(True)
        else
            luggage.setSuspicious(False)
        end if
        return True
    end if
    return False
end function
```

Algorithm 54 CT SinglePlex Activity Method: $CTSinglePlexActivity$.UPDATE()

```
function CTSinglePlexActivity.UPDATE()
    if luggage.CANDIVERT() then          ▷ Luggage can divert once decisionTime has passed.
        ENDACTIVITY()
    end if
end function
```

CT MultiPlex Activity

An operator activity. The CT MultiPlex activity is similar to the CT SinglePlex activity. THe difference is that in the SinglePlex activity, an operator is only able to check the CT images of a single sensor while in the MultiPlex activity the operator is able to check the CT images of the entire checkpoint. This differences arises in the $canStart()$ method of the MultiPlex activity. Instead of obtaining the collection of $notCheckedLuggage$ from a single sensor, this collection is determined from the entire checkpoint. The $canStart()$ and $update()$ methods are presented in Algorithm 55 and Algorithm 56 respectively.

CT MultiPlex Activity Attributes

MultiPlexAct$_{Att.}$ **01** *Checkpoint*
DESCRIPTION: The checkpoint that the operator is assigned to.
INITIALIZATION: According to airport layout.

MultiPlexAct$_{Att.}$ **02** *Activity Distribution*
DESCRIPTION: A time distribution representing the performance of the operator.
INITIALIZATION: According to operator attributes.

Algorithm 55 CT MultiPlex Activity Method: $CTMultiPlexActivity$.CANSTART()

```
function CTMultiPlexActivity.CANSTART()
    if not checkpoint.GETNOTCHECKEDLUGGAGE() = Empty then
        luggage ← checkpoint.GETNOTCHECKEDLUGGAGE().SELECTFIRST()
        checkpoint.GETNOTCHECKEDLUGGAGE().REMOVE(luggage)
        decisionTime ← activityDistribution.SAMPLE()
        observation ← OBSERVE(luggage through sensor)
        if observation = alarm then
            luggage.SETSUSPICIOUS(True)
        else
            luggage.SETSUSPICIOUS(False)
        end if
        return True
    end if
    return False
end function
```

Algorithm 56 CT MultiPlex Activity Method: $CTMultiPlexActivity$.UPDATE()

```
function CTMultiPlexActivity.UPDATE()
    if luggage.CANDIVERT() then          ▷ Luggage can divert once decisionTime has passed.
        ENDACTIVITY()
    end if
end function
```

Physical Check Activity

An operator activity. Starts either when there is a passenger to scan or to frisk. The operator will always first perform a frisk procedure is that operator is able to start a frisk procedure, else the operator will start the scan procedure. If there is a passenger to scan, the operator begins the security scanner. Once scanned, the operator checks whether or not the security scanner raise an alarm. If no alarm was raised,

then the passenger is considered as unsuspicious and is cleared. By contrast, if an alarm was raised, the passenger is considered suspicious and must be frisked. If there is a passenger to be frisked, then the operator commands the passenger to move to the frisk position. It is important to note that an operator can only frisk a passenger if they have the same gender. Once the position has been reached, the passenger is searched for forbidden items. The time of the search is determined by taking a random sample from the operator performance distribution. When the search is done, the passenger is marked as not suspicious and the activity is completed. The $canStart()$ and $update()$ methods are presented in Algorithm 57 and Algorithm 58 respectively. The methods called within the $update()$ are illustrated in Algorithm 60 and Algorithm 59.

Physical Check Activity Attributes

PhysCheckAct$_{Att.}$ **01** *Checkpoint*
DESCRIPTION: The checkpoint that the operator is assigned to.
INITIALIZATION: According to airport layout.

PhysCheckAct$_{Att.}$ **02** *Scan or Frisk*
DESCRIPTION: A boolean indicating whether to frisk or scan.
INITIALIZATION: Null.

PhysCheckAct$_{Att.}$ **03** *Activity Distribution*
DESCRIPTION: A time distribution representing the performance of the operator.
INITIALIZATION: According to operator attributes.

PhysCheckAct$_{Att.}$ **04** *Phase*
DESCRIPTION: Indicates what phase of the activity an agent is in.
INITIALIZATION: 0.

Algorithm 57 Physical Check Activity Method: $PhysicalCheckActivity$.CANSTART()

```
function PhysicalCheckActivity.canStart()
    if not checkpoint.getPassengersToFrisk() = Empty then
        for pax in checkpoint.getPassengersToFrisk() do
            if isSameSex(pax) then
                checkpoint.getPassengersToFrisk().remove(pax)
                scanOrFrisk = Frisk
                return True
            end if
        end for
    end if
    if not checkpoint.getPassengersToScan() = Empty then
        for pax in checkpoint.getPassengersToScan() do
            checkpoint.getPassengersToScan().remove(pax)
            scanOrFrisk = Scan
            return True
        end for
    end if
    return False
end function
```

Algorithm 58 Physical Check Activity Method: $PhysicalCheckActivity$.UPDATE()

```
function PhysicalCheckActivity.update()
    if scanOrFrisk=Scan then
        scanning()
    else if scanOrFrisk=Frisk then
        frisking()
    end if
end function
```

Algorithm 59 Physical Check Activity Method: $PhysicalCheckActivity$.SCANNING()

```
function PhysicalCheckActivity.SCANNING()
    if pax.GETREACHEDGOAL() and not pax.ISSCANNED() then
        scanTime ← pax.GETSCANTIME()
        sensor ← OBSERVE(SScSensor.class containing pax)
        COMMUNICATE(WaitCommand of scanTime for pax)
        pax.SETSCANNED(True)
    end if
    if pax.ISSCANNED() and not pax.ISINSTOPORDER() then
        observation ←OBSERVE(pax THROUGH sensor)
        if observation = alarm then
            pax.SETSUS(True)
            checkpoint.GETPASSENGERSTOFRISK().ADD(pax)
        else
            pax.SETSUS(False)
        end if
        ENDACTIVITY()
    end if
end function
```

Algorithm 60 Physical Check Activity Method: $PhysicalCheckActivity$.FRISKING()

```
function PhysicalCheckActivity.FRISKING()
    if phase = 0 then
        COMMUNICATE(GoToCommand to position for pax)
        phase = phase + 1
    else if phase = 1 and pax.GETPOSITION()=position then
        friskTime ← activityDistribution.SAMPLE()
        COMMUNICATE(WaitCommand of friskTime for pax)
        phase = phase + 1
    else if phase = 2 and not pax.ISINSTOPORDER() then
        pax.SETSUS(True)
        ENDACTIVITY()
    end if
end function
```

Luggage Recheck Activity

An operator activity. Starts when there is a luggage on the end position of the reject belt. An operator can be assigned to multiple luggage belts and is not allocated to a single belt. An operator picks up the luggage and motions for the owner to move to the recheck position. Once the owner has arrived at the position, the operator begins the search. The time of the recheck is determined by taking a random sample from the operator performance distribution. When the search is complete, the luggage is handed over to the passenger and the activity is completed. The $canStart()$ and $update()$ methods are presented in Algorithm 61 and Algorithm 62 respectively.

Luggage Recheck Activity Attributes

LugRecheckAct$_{Att.}$ **01** *Reject Belts*
DESCRIPTION: A collection of reject belts that the operator is assigned to.
INITIALIZATION: According to airport layout.

LugRecheckAct$_{Att.}$ **02** *Activity Distribution*
DESCRIPTION: A time distribution representing the performance of the operator.
INITIALIZATION: According to operator attributes.

LugRecheckAct$_{Att.}$ **03** *Phase*
DESCRIPTION: Indicates what phase of the activity an agent is in.
INITIALIZATION: 0.

Algorithm 61 Luggage Recheck Activity Method: $LuggageRecheckActivity$.canStart()

```
function LuggageRecheckActivity.canStart()
    luggages ←observe(belt.getLuggageAtEnd() if belt.endOccupied() for belt in rejectBelts)
    luggage ← selectLongestWaiting(luggages)
    if not luggage = Null then
        pax ← luggage.getOwner()
        Remove(luggage from belt)
        return True
    end if
    return False
end function
```

Algorithm 62 Luggage Recheck Activity Method: $LuggageRecheckActivity$.update()

```
function LuggageRecheckActivity.update()
    if phase = 0 and pax.getPosition()=position then
        searchTime ← activityDistribution.sample()
        communicate(WaitCommand of searchTime for pax)
        phase = phase + 1
    else if phase = 1 and not pax.isInStopOrder() then
        give(luggage to pax)
        endActivity()
    end if
end function
```

Appendix B

Case Studies for Amsterdam Airport Schiphol

The agent-based simulator was developed in collaboration with Amsterdam Airport Schiphol (AAS). Alongside the research as presented in the research paper, the developed simulator was also used in the operational planning of AAS. Furthermore, the simulator was applied in an investigative study of future security checkpoint configurations. These case studies aimed to answer operational questions considered by Schiphol airport managers. Next to that, these case studies show the potential of using agent-based modelling in an operational setting, as it can provide detailed insights into the operational performance.

The first case investigates the sensitivity of the conventional security checkpoint configuration (see Appendix B.1). The second case studies the effect of different operator allocations at the AAS security checkpoint during the COVID-19 pandemic (see Appendix B.2). The third case study regards a preliminary sizing and sequential analysis of a novel checkpoint configuration (see Appendix B.3). Finally, a graphical user interface of the AATOM simulator was created (see Appendix B.4).

B.1. Sensitivity of conventional checkpoint configuration

A brief local sensitivity study is done to understand the sensitivity of the agent-based model's input parameters. This local sensitivity also suffices as a means for validation. This is because the security experts from AAS have experienced which parameters tend to have the largest effect on the throughput performance of the checkpoint.

A tornado plot is shown in Figure B.1. This plot shows the impact of a 10% increase/decrease on the throughput of the security checkpoint. Most remarkable is the effect of the image factor, scan time and divest time. When increasing these parameters with 10%, the throughput degrades with at least 5%. Schiphol experts indicate that the image factor is one of the factors that influence the performance of a checkpoint highly.

For understanding, the image factor is the average number of trays that a passenger has. If the image factor increases, then there will be more trays on the luggage belts. This increases the chance of *dieback* which causes a significant decrease in checkpoint performance.

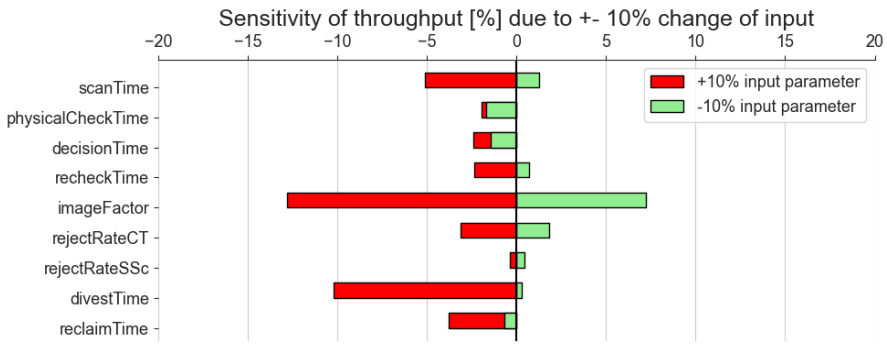


Figure B.1: Tornado plot with the +-10% sensitivity on the throughput

B.2. Security operator allocation under COVID-19 restrictions

March 2020 marked the outbreak of the global COVID-19 pandemic. To reduce the spread of this virus, countries introduced multiple regulations. A major regulation was the restriction of international air travel,

which led airline passenger numbers to drop by 60% (United Nations 2021). To limit losses due to this outbreak, AAS was forced to downsize in operations and personnel. As national vaccination programs started picking up, the COVID restrictions were slowly alleviated throughout 2021. This led air travel to increase again.

In April 2021, AAS was faced with limited security operator availability as earlier COVID restrictions led to downsizing of personnel. This introduced a challenging situation in which predicted operator availability would not be sufficient for the expected demand in air travel, this imbalance could potentially lead to bottlenecks at the airport security checkpoint. To overcome this challenge, AAS aimed to improve operator allocation within the security checkpoints, resulting in a higher throughput. As decision support, the agent-based simulator was used to provide performance insights into different security operator allocation scenarios. The goal of this case study is to determine the allocation of security operators that would increase the throughput of the security checkpoint, while adhering to the limited security operator availability.

To determine a security operator allocation that would increase the throughput, a two-phase approach was considered. The initial phase aims to determine the performance of the security checkpoint as is at AAS, the method and results for this phase are presented in Appendix B.2.1. Thereafter, phase 2 leverages on the performance insights of the initial phase to improve operations, both the method and results are presented in Appendix B.2.2.

B.2.1. Phase I: Performance of the AAS security checkpoint

To accurately redesign the operator allocation of the security checkpoint it is necessary to understand the performance of the existing checkpoint. Below the method to establishing this understanding, along with the results are presented.

Method

Initially, data was acquired to establish a baseline value for the performance of the security checkpoint. This data was gathered through both manual measurements at the security checkpoint, and by means of the data warehouse sources of AAS. The configuration of existing security checkpoint is depicted in Figure B.2. In this checkpoint bay, 4 operators are allocated to the security scanners, 2 operators are allocated the CT scanners, and 2 operators are allocated to luggage recheck.

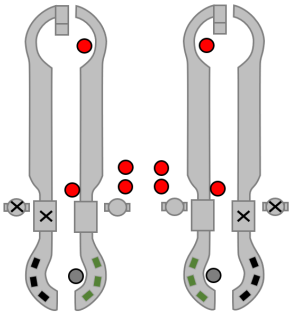


Figure B.2: Current security checkpoint concept at terminal 3 of *Schiphol Airport*

The performance of the security checkpoint is scored by quantifying five performance indicators:

1. **Throughput:** The main indicator for the capacity of the security lane. This is quantified as passengers/min/lane. This performance indicator can be derived both through manual measurements as well as simulator output.
2. **Operator throughput:** An indicator for the efficiency of the operator. It is quantified as passengers/min/operator. This performance indicator can only be derived from simulator output.
3. **Operator utilisation:** Quantified as the percentage of time that the operators are actually productive. This performance indicator can only be derived from simulator output.
4. **Time in critical operations:** This performance indicator provides insights into which security checkpoint sub process causes the bottleneck. It is quantified as percentage of time that a sub process is in critical operations. A more detailed description of this indicator is found in the research paper. This performance indicator can only be derived from simulator output.

5. **Spread in simulation performance output:** An indicator for the stability of a security checkpoint performance. If all of the simulations lie within a small range, it indicates that that specific configuration is predictable and resilient. It is determined by the difference in maximum and minimum achieved throughput. A small range is favourable as it indicates a smaller spread.

Results

The performance of the baseline configuration provides insights into what measures should be taken to improve the current configuration. The performance of the baseline configuration is depicted in Table B.1. It can be seen that the simulated throughput aligns with measured throughput.

Table B.1: The performance of the baseline configuration of the security checkpoint

	Baseline configuration
Measured throughput [pax/min/lane]	2.33
Simulated throughput [pax/min/lane]	2.35
Simulated operator throughput [pax/min/operator]	0.29
Spread in simulation performance [pax/min/lane]	1

Furthermore, the time in critical operations as well as the density of the simulation throughput are visualised in Figure B.4 and Figure B.3 respectively. Noteworthy is the large tail on the lower end of the density of simulation throughput. This indicates that there are numerous simulations in which a low throughput is achieved. In reality, this property is also observed and referred to as dieback: a situation where the luggage belts is completely occupied, which leaves no space for operations to proceed. From the time in critical operations it is observed that the SSc operator is not in critical operations at all, while the recheck process and CT process are in critical operations for larger parts of the simulations.

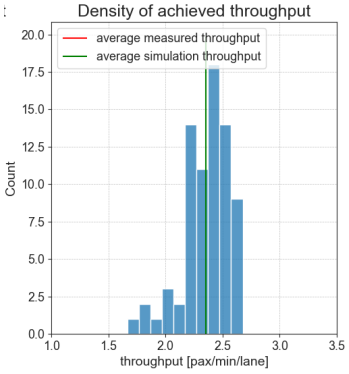


Figure B.3: Density of the achieved simulation throughput for the baseline configuration.

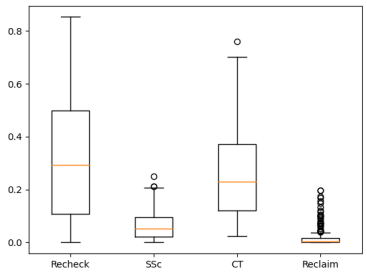


Figure B.4: The time in critical operation for each sub process for the baseline configuration.

These performance insights can be used to determine new security operator allocations. This is done in phase two.

B.2.2. Phase II: Improving the AAS security checkpoint

Together with experts from AAS, three different operational concepts were developed. These were designed with the goal to improve checkpoint performance. The method and results are presented below.

Method

The new operational concepts with different operator allocations were designed using the insights from phase 1, i.e. the baseline configuration. From these results it was concluded that the SSc operator is not in critical operations, while the CT and recheck show high critical operations. Three alternative concepts were derived that aim to limit the bottlenecks at the CT and recheck. This is done by using resources from the well performing SSc process. These new concepts are depicted in Figure B.5, Figure B.6 and Figure B.7.

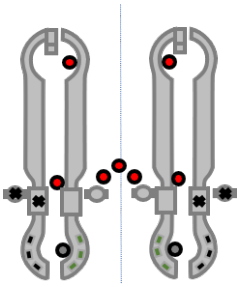


Figure B.5: Concept 1: removing a SSc operator.

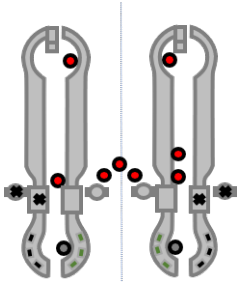


Figure B.6: Concept 2: reassigning a SSC operator to CT positions.

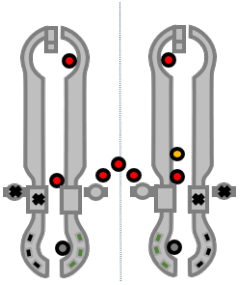


Figure B.7: Concept 3: reassigning a SSC operator to CT or recheck.

Concept 1 reduces the total amount of operators by removing an operator from the SSc. In this scenario, 3 SSc operators operate 2 SSc assets. Concept 2 does not remove this operator, rather it reassigns this operator to a more critical area within the checkpoint: the CT position. This operator is able to assist at both CT positions. Concept 3 is considered more dynamic. It also reassigns the task of one of the SSc operators, however, the task of this reassigned operator can either be assisting at the CT check or luggage recheck. A business rule is established: *if there are more than 4 luggages on the reject belt, the operator will be assigned to assist in luggage recheck. Else, the operator will assist in CT check*.

Each of these configurations were ran in the agent-based simulator. The number of required simulations is established by ensuring that the coefficient of variation is stable. This occurs after $n = 50$ simulations. The performance of these checkpoints was determined by using the same performance indicators as used in phase one. To determine which configuration is to be implemented, a comparison of the results between the baseline and proposed configurations is done. The results of this comparison is the topic of the next section.

Results & Discussion

The performance results of all tested concepts is compared against the performance results of the baseline configuration, this is depicted in Table B.2, in bold the best scoring concept for that specific KPI is identified. Concecpt III shows the best performance in both throughput and spread in simulation performance. This indicates that this concept has the highest capacity and is also the most stable in operations. Furthermore, when comparing the operator throughput Concept III only scores slightly worse than Concept I, while Concept I has 7 operators and Concept III has 8 operators. This is explained by the fact that the operators are more effectively utilised. From the operator utilisation it can be seen that by removing a SSc operator from the baseline concept, and re-assigning it elsewhere, the operator utilisation is more balanced. This is especially apparent in Concept III. In this concept the spare operator is dynamically assigned to either CT or recheck, this evens out the operator utilisation. As the operator utilisations are more evenly distributed over the operators, it can be stated that the workload of the operators is more evenly distributed.

Table B.2: Simulation results of the baseline concept and proposed operator allocations.

	Baseline configuration	Concept I	Concept II	Concept III
Measured throughput [pax/min/lane]	2.33	NA	NA	NA
Simulated throughput [pax/min/lane]	2.36	2.36	2.58	**2.66**
Simulated operator throughput [pax/min/operator]	0.29	**0.34**	0.32	0.33
Spread in simulation performance [pax/min/lane]	0.88	1.2	0.51	**0.45**
Utilisation CT operator [%]	86%	86%	79%	81%
Utilisation male SSc operator [%]	66%	70%	74%	74%
Utilisation female SSc operator [%]	20%	33%	36%	37%
Utilisation recheck operator [%]	86%	86%	83%	79%

To further investigate the spread in simulation performance, the density of the achieved throughput was plotted. These are visualized in Figure B.8, Figure B.9, Figure B.10, and Figure B.11. Especially noteworthy in these figures is the phenomenon of simulations with lower achieved throughput, which is apparent in the baseline configuration, concept 1, and to a smaller extend in concept 2 as well. This indicates that there are multiple simulations in which the throughput drastically decreases. This is unfavourable, as it makes the configuration of the checkpoint less resilient and less predictable. The reason that Concept 3 does not

show this property is due to the dynamic changing of the spare operator, therefore bottlenecks are resolved very locally within the checkpoint lanes. To substantiate this claim the time in critical operations of each different concept is visualised. This is depicted in Figure B.12, Figure B.13, Figure B.14, and Figure B.15. With respect to all other configurations, Concept 3 has very little time in critical operations, hence very little bottlenecks. Also, when comparing the spread of the boxplots, its is much lower for Concept 3 - again indicating the reliability of this concept.

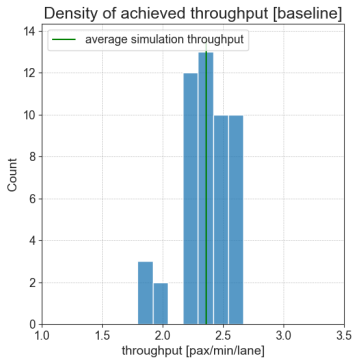


Figure B.8: Baseline

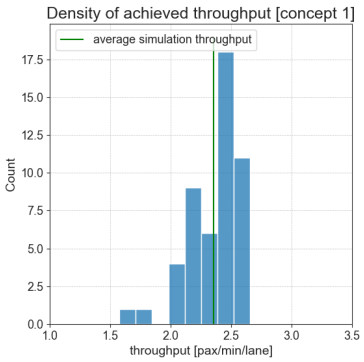


Figure B.9: Concept 1

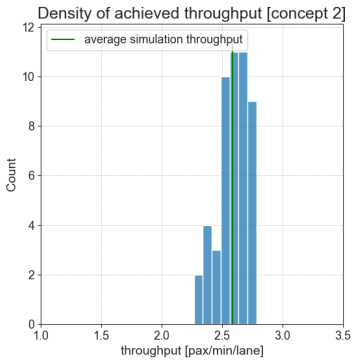


Figure B.10: Concept 2

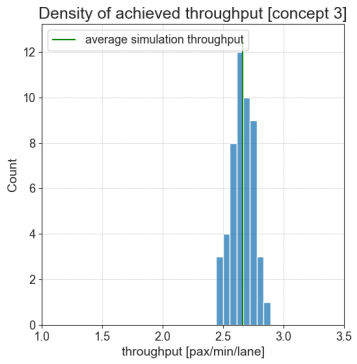


Figure B.11: Concept 3

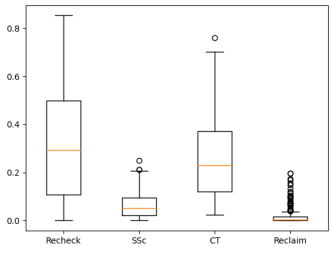


Figure B.12: Baseline

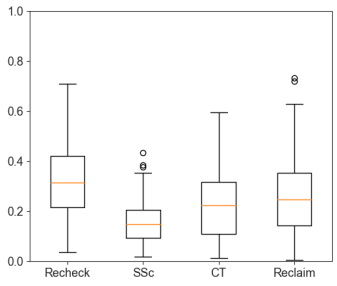


Figure B.13: Concept 1

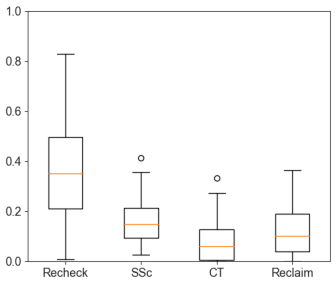


Figure B.14: Concept 2

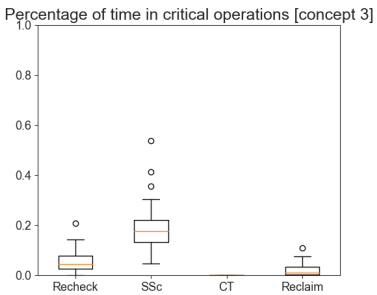


Figure B.15: Concept 3

Conclusion

In line with the results, the simulated performance of Concept 3 was considered most promising. To conclude, Concept 3 was chosen as it was able to achieve the highest passenger throughput [pax/min/lane] while showing very little decrease in operator throughput [pax/min/agent]. Furthermore the reliability of this configuration is higher than all other configurations, as the range in which the simulated throughput fall is much more narrow. Finally, the operator utilisation is more balanced indicating that the workload is more evenly distributed over the different agents. For these reasons, this concept was selected for implementation at the security checkpoints of AAS. As of writing this report, the working instructions of the security personnel

at AAS have been altered such that these newly developed instructions are incorporated.

B.3. Future security checkpoints

The conventional checkpoint configuration, which is currently also operated by AAS, consists of a lane configuration. The entire security checkpoint consists of multiple security lanes. Inherent to the lane configuration is the one-to-one relationship of security assets; each lane has one CT scanner, one Security Scanner and one Luggage Recheck position. The airport security experts at AAS believe that this one-to-one relationship of assets will bound the efficiency performance of the security checkpoint. Reason being that the lane configuration reduces the flexibility in optimally utilising individual assets, as the lane is always restricted by the worst performing asset.

Therefore, AAS is exploring novel checkpoint configurations. Specifically, AAS is analysing the configuration where individual security processes are grouped together. An example of this configuration in shown in Figure B.16. Airport experts believe that this novel checkpoint configuration introduces multiple advantages. First and foremost, the disregard of the one-to-one asset relationship, which would allow better utilisation of assets. Second, passengers and luggage can overtake, meaning that the throughput of a checkpoint is no longer dictated by its slowest passenger. And third, airport security experts believe that such configuration could also reduce the required area, which is considered as scarce and expensive resource at airport terminals.

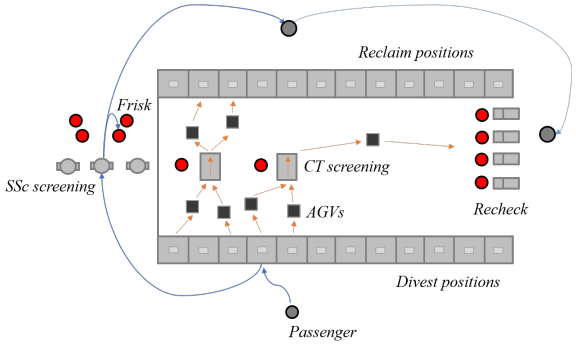


Figure B.16: A schematic of the novel non-lane checkpoint configuration.

This case study aimed to quantify the difference in performance between a conventional lane concept and a novel non lane concept at departure terminal 3 at AAS, while keeping the number of security assets and operators equal.

Method

In line with the aim of this case study, a three step approach was used to determine the performance difference between conventional and novel checkpoint configuration. First, a preliminary performance estimation of future security assets based on expert opinion was executed. Then, the novel checkpoint is placed within the available area. Lastly, both the conventional and novel checkpoint configuration are simulated after which a comparison is executed.

As the development of a novel checkpoint configuration will require time, it was deemed necessary to estimate the performance of future security assets to account for the innovations that are feasible in the near future. To do so, security experts from AAS quantified potential improvements to the current technology. Using these estimations, four scenarios were derived. In each of these scenarios multiple technological advancements were considered. The four scenarios are:

1. **Scenario 1: Improved CT, recheck, and SSc algorithms:** The improved CT algorithms reduce the reject rate of the CT and decreases the time a CT operator requires to make a decision on a specific luggage tray. Due to improved recheck algorithms, the recheck operator deals with less complex luggage rechecks, leading to a lower recheck time. Due to improved SSc algorithms the SSC reject rate goes down. Also, passengers no longer have to divest complex items like belts and shoes. This results in a lower amount of trays per passenger, i.e. image factor, and a lower divest time and reclaim time.

2. **Scenario 2: Walk through SSc:** This scenario is identical to scenario 1, however, it consists of walk through SSc. This reduces the scan time per passenger.
3. **Scenario 3: Improved CT with AUTOCLEAR:** This scenario is very similar to scenario 1. However, a new promising technology is (AUTOCLEAR) is implemented. This technology results in CT operators not having to check every luggage tray. Instead, a predetermined amount of luggage trays is automatically cleared (hence AUTOCLEAR) by a CT algorithm. Therefore, the amount of trays the CT operator has to check is reduced.
4. **Scenario 4 Improved CT with AUTOCLEAR and walk through SSc:** This scenario is identical to scenario 3, however, it consists of walk through SSc. This reduces the scan time per passenger.

With these scenarios defined and quantified, a preliminary sizing was done by means of previous research executed at AAS and individual process rates of the security process. It should be noted that the novel configuration that was developed has exactly the same amount of operators as the conventional configuration. The checkpoint configuration for the conventional setup and novel setup are depicted in Figure B.17 and Figure B.18 respectively. Both of these checkpoint configurations were simulated for scenario one to four. To compare the performance of each of these configurations a total of 3 performance indicators were employed: (1) throughput [pax/min/checkpoint], (2) individual operator utilisation [%], and (3) time in critical operations.

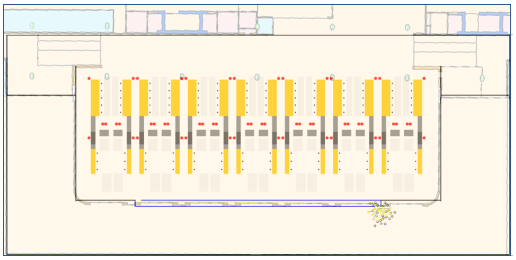


Figure B.17: The layout of the security checkpoint at departure filter 3 of AAS.

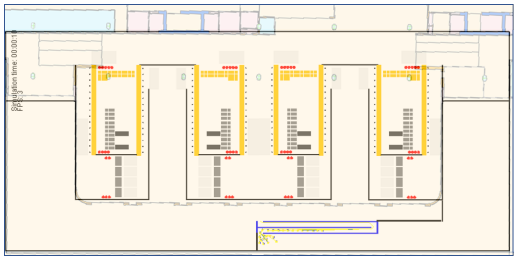


Figure B.18: The proposed novel configuration of the security checkpoint at departure filter 3 of AAS.

Results & Discussion

To analyse the performance of the different security checkpoints the throughput, operator utilisation and time in critical operations were considered. As indicated in Table B.3, the throughput of the novel checkpoint is higher than that of the conventional checkpoint for all simulated scenarios. This is due to the increase in number of divest positions in the novel concept. Due to this increase in divest positions, the total inflow of passengers is larger, which ultimately determines the maximum theoretical throughput that can be achieved by a checkpoint.

Although the throughput is higher for the novel concept, it is still believed that the potential gain in throughput is larger than that of the conventional concept. This is substantiated by the percentage of time in critical operations as depicted in Table B.4. From this table it is clear that the conventional checkpoint has very low time in critical operations, thus almost no bottlenecks are present. On the other hand, the novel concept is on average 24% of the time in critical operations. With an average of 24%, quite some bottlenecks are still to be resolved in this configuration, meaning that the throughput can theoretically still increase. It is exactly the low amount of critical operations which indicates that there is no room for improvement in the configuration of the conventional checkpoint, as there are no bottlenecks to resolve.

Table B.3: Throughput of conventional and novel checkpoint configuration for different scenarios.

throughput [pax/min/checkpoint]	**S1**	**S2**	**S3**	**S4**	**mean**
conventional checkpoint	41.7	42.8	42.2	45.2	**43.0**
novel checkpoint	49.8	53.0	49.7	53.1	**51.4**

Table B.4: Mean percentage of time in critical operations of conventional and novel checkpoint configuration for different scenarios.

critical operations [%]	**S1**	**S2**	**S3**	**S4**	**mean**
conventional checkpoint	10	2	9	2	**6**
novel checkpoint	32	14	35	14	**24**

Table B.5: Mean operator utilisation of conventional and novel checkpoint configuration for different operator types.

Mean operator utilisation [%]	**CT**	**Male SSc**	**Female SSc**	**Recheck**	**mean**
conventional checkpoint	52	72	25	32	**45**
novel checkpoint	54	87	68	25	**59**

Furthermore, for both the conventional and the novel checkpoint, the operator utilisation is unevenly distributed between operator types. For the conventional checkpoint both the female SSc operator and recheck operator show poor utilisation. For the novel checkpoint only the recheck operator shows poor utilisation. The discrepancy between these operator utilisation indicates that both checkpoints have not optimally allocated the operators. Hence, changing the allocation might improve the throughput. This is not always possible as certain security assets require a minimum amount of operators. This is especially the case in the conventional concept for the recheck operator. With a utilisation of 32% this operator is poorly utilised. However, it is not possible to remove this operator as the recheck position requires at least one operator. On the other hand, multiple recheck operators man the recheck station in the novel concept, it therefore becomes much easier to remove recheck operators and improve the recheck operator utilisation which is currently 25%. This phenomenon is inline with the expectation of security experts, who believe that the one-to-one relationship of the conventional configuration reduces the flexibility to optimise the lane.

Conclusion

In conclusion, the expectations of the AAS security experts fall in line with the simulated performance results when comparing the conventional checkpoint and novel checkpoint configuration. The quantification shows that the novel checkpoint performs better, while having a larger throughput potential than the conventional checkpoint. Furthermore, more flexibility in operator allocation is introduced in the novel configuration, allowing to evenly distribute the workload between security operators.

B.4. Application for the checkpoint simulator

An application has been developed such that airport managers can use the simulator without programming experience. The application was developed in python using the *tkinter* package. Airport managers are able to select four different security checkpoint configurations. For each configuration, there are different input variables available. When the *simulate* button is pressed, the application launches the AATOM simulator and simulates the security checkpoint configuration selected. Once the simulation terminates, the results are automatically visualized in the application. A representation is shown in Figure B.19.

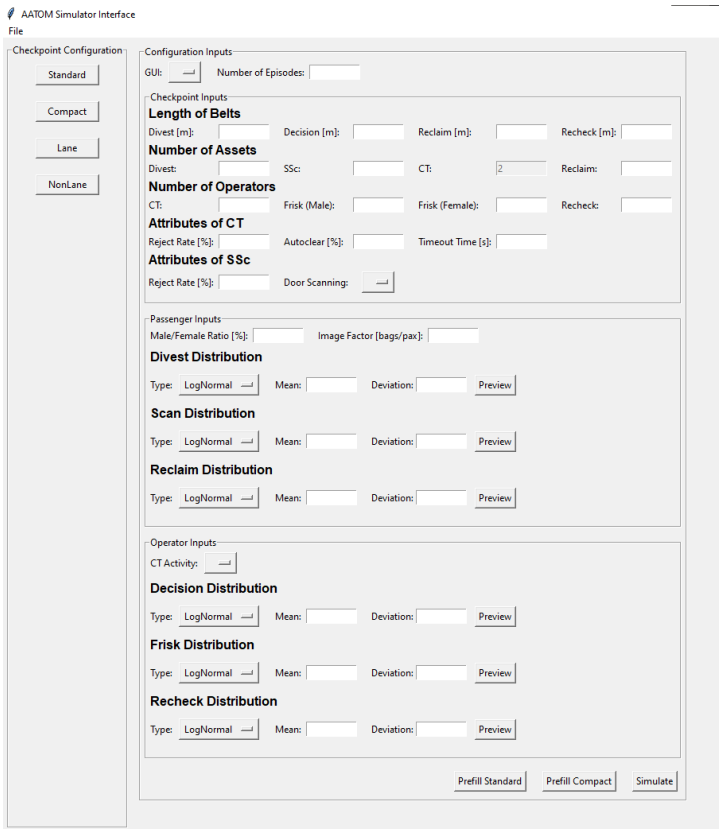


Figure B.19: The interface of the application that has been developed for the security checkpoint simulator.

Bibliography

[1] Airport Research Center. *CAST Terminal*. URL: https://arc.de/cast-terminal-simulation/. [Accessed 20-11-20].

[2] I. Akgun, A. Kandakoglu, and A.F. Ozok. "Fuzzy integrated vulnerability assessment model for critical facilities in combating the terrorism". In: *Expert Systems with Applications* 37.5 (May 2010), pp. 3561–3573. DOI: 10.1016/j.eswa.2009.10.035.

[3] M. Andrychowicz et al. "Hindsight experience replay". In: *Advances in neural information processing systems*. 2017, pp. 5048–5058.

[4] S. Appelt et al. "Simulation of passenger check-in at a medium-sized US airport". In: *2007 Winter Simulation Conference*. IEEE. Dec. 2007, pp. 1252–1260. DOI: 10.1109/WSC.2007.4419729.

[5] V.L.L. Babu, R. Batta, and L. Lin. "Passenger grouping under constant threat probability in an airport security system". In: *European Journal of Operational Research* 168.2 (Jan. 2006), pp. 633–644. DOI: 10.1016/j.ejor.2004.06.007.

[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*. IEEE. Macao, China, Dec. 2010, pp. 1621–1625. DOI: 10.1109/IEEM.2010.5674286.

[7] J.P. Braaksma and W.J. Cook. "Human orientation in transportation terminals". In: *Transportation engineering journal of the American Society of Civil Engineers* 106.2 (1980), pp. 189–203.

[8] J.P. Brashear and J.W. Jones. "Risk analysis and management for critical asset protection (RAMCAP plus)". In: *Wiley handbook of science and technology for homeland security* (Feb. 2008), pp. 1–15. DOI: 10.1002/9780470087923.hhs003.

[9] M. Brown et al. "One size does not fit all: A game-theoretic approach for dynamically and effectively screening for threats". In: *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI)*. Phoenix, Arizona, USA: ACM, Feb. 2016, pp. 425–431. DOI: 10.5555/3015812.3015877.

[10] L. Brunetta, L. Righi, and G. Andreatta. "An operations research model for the evaluation of an airport terminal: SLAM (simple landside aggregate model)". In: *Journal of Air Transport Management* 5.3 (July 1999), pp. 161–175. DOI: 10.1016/S0969-6997(99)00010-1.

[11] P.K. Chawdhry. "Risk modeling and simulation of airport passenger departures process". In: *Proceedings of the 2009 Winter Simulation Conference (WSC)*. Austin, TX, USA: IEEE, Dec. 2009, pp. 2820–2831. DOI: 10.1109/WSC.2009.5429244.

[12] S. Crook. "The use of simulation and virtual reality in the design and operation of airport terminals". In: *International Conference on Simulation* (1998), pp. 8–10. DOI: 10.1049/cp:19980609.

[13] *ECAC Common Evaluation Process of Security Equipment*. European Civil Aviation Conference. URL: https://www.ecac-ceac.org/cep-main. [Online; Accessed 14-10-20].

[14] *Economic performance of the airline industry*. End-Year Report. International Air Transport Association, Dec. 2019. URL: https://www.iata.org/en/iata-repository/publications/economic-reports/airline-industry-economic-performance---december-2019---report/. [Accessed 08-10-20].

[15] S. Eilon and S. Mathewson. "A simulation study for the design of an air terminal building". In: *IEEE Transactions on Systems, Man, and Cybernetics* 4 (July 1973), pp. 308–317. DOI: 10.1109/TSMC.1973.4309241.

[16] M.S. Fayez et al. "Managing airport operations using simulation". In: *Journal of Simulation* 2.1 (Dec. 2008), pp. 41–52. DOI: 10.1057/palgrave.jos.4250030.

[17] *FY 2020 Budget in Brief*. Finance Report. Homeland Security, 2019. URL: https://www.dhs.gov/sites/default/files/publications/fy_2020_dhs_bib.pdf. [Accessed 08-10-20].

[18] M.R. Gatersleben and S.W. van der Weij. "Analysis and Simulation of Passenger Flows in an Airport Terminal". In: *Proceedings of the 31st Conference on Winter Simulation: A Bridge to the Future*. Vol. 2. Phoenix, Arizona, USA: Association for Computing Machinery, Dec. 1999, pp. 1226–1231. DOI: 10.1145/324898.325045.

[19] F. Glover and M. Laguna. "Tabu search". In: *Handbook of combinatorial optimization*. Boston, MA, USA: Springer, 1998, pp. 2093–2229. DOI: 10.1007/978-1-4613-0303-9_33.

[20] M. Gongora and W. Ashfaq. "Analysis of passenger movement at birmingham international airport using evolutionary techniques". In: *2006 IEEE International Conference on Evolutionary Computation*. IEEE. Vancouver, BC, Canada, July 2006, pp. 1339–1345. DOI: 10.1109/CEC.2006.1688464.

[21] S.G. Hamzawi. "Management and planning of airport gate capacity: a microcomputer-based gate assignment simulation model". In: *Transportation Planning and Technology* 11.3 (July 1986), pp. 189–202. DOI: 10.1080/03081068608717341.

[22] H.V. Hasselt. "Double Q-learning". In: *Advances in neural information processing systems*. Vancouver, British Columbia, Canada: Curran Associates Inc., 2010, pp. 2613–2621. DOI: 10.5555/2997046.2997187.

[23] John H Holland. "Genetic algorithms". In: *Scientific american* 267.1 (1992), pp. 66–73.

[24] R. Horonjeff et al. *Planning and design of airports*. McGraw-Hill Companies, 2010.

[25] D.J. van der Horst. "Simulation optimisation approach to configuration of a novel security checkpoint". MA thesis. Delft University of Technology, Nov. 2021.

[26] HUB Performance. *PAX2Sim Applications*. URL: http://www.hubperformance.com/products/pax2sim-applications. [Accessed 20-11-20].

[27] *Implementing Regulation (EU) 2015/1998*. Legislation Report L 299/1. European Comission, Nov. 2015. URL: http://data.europa.eu/eli/reg_impl/2015/1998/oj.. [Accessed 14-10-20].

[28] INCONTROL Airport Simulation Solutions. *Digital Brochure - Passenger Flows*. URL: https://support.incontrolsim.com/en/pd-solution-areas/265-digital-brochure-passenger-flows-hq/download.html. [Accessed 20-11-20].

[29] M. Jaderberg et al. "Reinforcement learning with unsupervised auxiliary tasks". In: *arXiv preprint arXiv:1611.05397* (2016).

[30] S. Janssen et al. "AATOM: An Agent-Based Airport Terminal Operations Model Simulator". In: SummerSim '19. Berlin, Germany: Society for Computer Simulation International, July 2019, p. 12. DOI: 10.5555/3374138.3374158.

[31] H.K. Jim and Z.Y. Chang. "An airport passenger terminal simulator: A planning and design tool". In: *Simulation Practice and Theory* 6.4 (May 1998), pp. 387–396. DOI: 10.1016/S0928-4869(97)00018-9.

[32] P.E. Joustra and N.M. Van Dijk. "Simulation of check-in at airports". In: *Proceeding of the 2001 Winter Simulation Conference*. Vol. 2. Arlington, VA, USA: IEEE, Dec. 2001, pp. 1023–1028. DOI: 10.1109/WSC.2001.977409.

[33] W.J Dunlay Jr et al. "A system analysis procedure for estimating the capacity of an aiport: system deinfition, capacity definition and review of available models." In: (Oct. 1975).

[34] W.J. Dunlay Jr and C.H. Park. "Tandem-queue algorithm for airport user flows". In: *Journal of Transportation Engineering* 104 (Mar. 1978), pp. 131–149.

[35] W.J. Dunlay Jr and C.H. Park. "Tandem-queue algorithm for airport user flows". In: *Journal of Transportation Engineering* 104.2 (1978).

[36] Y. Ju, A. Wang, and H. Che. "Simulation and optimization for the airport passenger flow". In: *2007 International Conference on Wireless Communications, Networking and Mobile Computing*. Shanghai, China: IEEE, Sept. 2007, pp. 6605–6608. DOI: 10.1109/WICOM.2007.1621.

[37] J. Kennedy and R. Eberhart. "Particle swarm optimization". In: *Proceedings of ICNN'95-International Conference on Neural Networks*. Vol. 4. IEEE. Perth, WA, Australia, Aug. 1995, pp. 1942–1948. DOI: 10.1109/ICNN.1995.488968.

[38] A. Kierzkowski and T. Kisiel. "Simulation model of security control system functioning: A case study of the Wroclaw Airport terminal". In: *Journal of Air Transport Management* 64 (Sept. 2017), pp. 173–185. DOI: 10.1016/j.jairtraman.2016.09.008.

[39] Byeoung-su Kim et al. "A method of counting pedestrians in crowded scenes". In: *International Conference on Intelligent Computing*. Springer. Berlin, Heidelberg, Germnany, 2008, pp. 1117–1126. DOI: 10.1007/978-3-540-85984-0_134.

[40] W. Kim, Y. Park, and B.J. Kim. "Estimating hourly variations in passenger volume at airports using dwelling time distributions". In: *Journal of Air Transport Management* 10.6 (Nov. 2004), pp. 395–400. DOI: 10.1016/j.jairtraman.2004.06.009.

[41] A.S. Kiran, T. Cetinkaya, and S. Og. "Simulation modeling and analysis of a new international terminal". In: *2000 Winter Simulation Conference Proceedings*. Vol. 2. Orlando, FL, USA: IEEE, Dec. 2000, pp. 1168–1172. DOI: 10.1109/WSC.2000.899081.

[42] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. "Optimization by simulated annealing". In: *science* 220.4598 (May 1983), pp. 671–680. DOI: 10.1126/science.220.4598.671.

[43] D.B. Koch. "3d visualization to support airport security operations". In: *IEEE Aerospace and Electronic Systems Magazine* 19.6 (June 2004), pp. 23–28. DOI: 10.1109/MAES.2004.1308826.

[44] LWW Laing. "A computer simulation model for the design of airport terminal buildings". In: *Computer-Aided Design* 7.1 (Jan. 1975), pp. 37–42. DOI: 10.1016/0010-4485(75)90138-4.

[45] R. Lui, R. Nanda, and J.J. Browne. "International passenger and baggage processing at john f. kennedy international airport". In: *IEEE Transactions on Systems, Man, and Cybernetics* 2 (Apr. 1972), pp. 221–225. DOI: 10.1109/TSMC.1972.4309096.

[46] S. Luke. *Essentials of Metaheuristics*. second. Lulu, Feb. 2013.

[47] I.E. Manataki and K.G. Zografos. "A generic system dynamics based tool for airport terminal performance analysis". In: *Transportation Research Part C: Emerging Technologies* 17.4 (Aug. 2009), pp. 428–443. DOI: 10.1016/j.trc.2009.02.001.

[48] I.E. Manataki and K.G. Zografos. "Assessing airport terminal performance using a system dynamics model". In: *Journal of Air Transport Management* 16.2 (Mar. 2010), pp. 86–93. DOI: 10.1016/j.jairtraman.2009.10.007.

[49] M.J. Mataric. "Reward functions for accelerated learning". In: *Machine learning proceedings 1994*. Elsevier, July 1994, pp. 181–189. DOI: 10.1016/B978-1-55860-335-6.50030-1.

[50] L. McCabe and M. Gorstein. "Airport Landside". In: 1 (June 1982).

[51] F.X. McKelvey. "Use of an analytical queuing model for airport terminal design". In: *Transportation Research Record* (1988), pp. 4–11.

[52] B.L. Metais. "Bechtel Airport Computer Simulation Model". In: *unpublished), June* (1974).

[53] E. Miller, G. LaFree, and L. Dugan. *Global Terrorism Database*. National Consortium for the Study of Terrorism and Responses to Terrorism. May 2018. URL: https://start.umd.edu/data-tools/global-terrorism-database-gtdg. [Online; Accessed 08-10-20].

[54] V. Mnih et al. "Playing atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602* (2013).

[55] S.A. Mumayiz. *A methodology for planning and operations management of airport passenger terminals: a capacity/level of service approach*. Jan. 1985. URL: https://hdl.handle.net/2134/7403.

[56] R. De Neufville and A. Odoni. *Airport Systems. Planning, Design and Management*. 2003.

[57] G.F. Newell. *Applications of queueing theory*. Chapman and Hall, 1982.

[58] X. Nie et al. "Passenger grouping with risk levels in an airport security system". In: *European Journal of Operational Research* 194.2 (Apr. 2009), pp. 574–584. DOI: 10.1016/j.ejor.2007.12.027.

[59] A.R. Odoni and R. de Neufville. "Passenger terminal design". In: *Transportation Research Part A: Policy and Practice* 26.1 (Jan. 1992), pp. 27–35. DOI: 10.1016/0965-8564(92)90042-6.

[60] D. Pathak et al. "Curiosity-driven exploration by self-supervised prediction". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2017, pp. 16–17.

[61] D.R. Pendergraft, C.V. Robertson, and S. Shrader. "Simulation of an airport passenger security system". In: *Proceedings of the 2004 Winter Simulation Conference, 2004.* Vol. 1. Washington, DC,USA: IEEE, Dec. 2004, p. 878. DOI: 10.1109/WSC.2004.1371402.

[62] J. Pita et al. "Using game theory for Los Angeles airport security". In: *AI magazine* 30.1 (Jan. 2009), pp. 43–43. DOI: 10.1609/aimag.v30i1.2173.

[63] I. Rechenberg. "Evolutionsstrategien". In: *Simulationsmethoden in der Medizin und Biologie*. Berlin, Germany: Springer, 1978, pp. 83–114. DOI: 10.1007/978-3-642-81283-5_8.

[64] *Regulation (EC) No 185/2010*. Legislation Report L 55/1. European Comission, Mar. 2010. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32010R0185&qid=1605260106074&from=EN. [Accessed 8-10-20].

[65] *Regulation (EC) No 2320/2002*. Legislation Report L 355/1. European Comission, Dec. 2002. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32002R2320&qid=1605253622133&from=EN. [Accessed 8-10-20].

[66] *Regulation (EC) No 300/2008*. Legislation Report L 97/72. European Comission, Mar. 2008. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32008R0300&from=EN. [Accessed 08-10-20].

[67] E. Roanes-Lozano, L.M. Laita, and E. Roanes-Macías. "An accelerated-time simulation of departing passengers flow in airport terminals". In: *Mathematics and Computers in Simulation* 67.1-2 (2004), pp. 163–172.

[68] E. Ruiz and R.L. Cheu. "Simulation model to support security screening checkpoint operations in airport terminals". In: *Transportation research record* 2674.2 (2020), pp. 45–56. DOI: 10.1177/0361198120903242.

[69] M. Saffarzadeh and J.P. Braaksma. "Optimum design and operation of airport passenger terminal buildings". In: *Transportation Research Record* 1703.1 (Jan. 2000), pp. 72–82. DOI: 10.3141/1703-10.

[70] P. Schouten. "Security as controversy: Reassembling security at Amsterdam Airport". In: *Security Dialogue* 45.1 (2014), pp. 23–42. DOI: 10.1177/0967010613515014.

[71] M. Schultz and H. Fricke. "Managing passenger handling at airport terminals". In: *9th Air Traffic Management Research and Development Seminars*. 2011.

[72] S. Solak, J.P.B Clarke, and E.L. Johnson. "Airport terminal capacity planning". In: *Transportation Research Part B: Methodological* 43.6 (July 2009), pp. 659–676. DOI: 10.1016/j.trb.2009.01.002.

[73] A.A. Soukour et al. "Staff scheduling in airport security service". In: *IFAC Proceedings Volumes* 45.6 (2012). 14th IFAC Symposium on Information Control Problems in Manufacturing, pp. 1413–1418. ISSN: 1474-6670. DOI: 10.3182/20120523-3-RO-2023.00169. URL: https://www.sciencedirect.com/science/article/pii/S1474667016333493.

[74] *Strategic objectives of ICAO for 2005-2010*. Strategic Report. International Civil Aviation Organization, Dec. 2004. URL: https://www.icao.int/Documents/strategic-objectives/strategic_objectives_2005_2010_en.pdf. [Accessed 08-10-20].

[75] *Study on civil aviation security financing*. Strategic Report TREN/F3/51-2002. Avia Solution, Sept. 2004. URL: https://ec.europa.eu/transport/sites/transport/files/themes/security/studies/doc/2004_09_study_financing_aviation_security_en.pdf. [Accessed 14-10-20].

[76] R.S. Sutton and A.G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[77] Patrick Taillandier et al. "GAMA: a simulation platform that integrates geographical information data, agent-based modeling and multi-scale control". In: *International Conference on Principles and Practice of Multi-Agent Systems*. Springer. 2010, pp. 242–258.

[78] S. Takakuwa, T. Oyama, and S. Chick. "Simulation analysis of international-departure passenger flows in an airport terminal". In: *Winter Simulation Conference*. Vol. 2. New Orleans, LA, USA: IEEE, Dec. 2003, pp. 1627–1634. DOI: 10.1109/WSC.2003.1261612.

[79] Seth Tisue and Uri Wilensky. "Netlogo: A simple environment for modeling complexity". In: *International conference on complex systems*. Vol. 21. Boston, MA. 2004, pp. 16–21.

[80] V. Tosic. "A review of airport passenger terminal operations analysis and modelling". In: *Transportation Research Part A: Policy and Practice* 26.1 (Jan. 1992), pp. 3–26. DOI: 10.1016/0965-8564(92)90041-5.

[81] V. Tosic, O. Babic, and M. Janic. "Airport passenger terminal simulation". In: *Annals of Operations Research in Air Transportation, Faculty of Transport and Traffic Engineering. University of Belgrade* (1983), pp. 83–103.

[82] United Nations. *Air travel down 60 per cent, as airline industry losses top $370 billion: ICAO*. 2021. URL: https://news.un.org/en/story/2021/01/1082302 (visited on 10/13/2021).

[83] J. William W.J. Dunlay Jr. "Model of airport employee access traffic". In: *Journal of Transportation Engineering* 104.3 (May 1978).

[84] C. Watkins and P. Dayan. "Q-learning". In: *Machine learning* 8.3-4 (1992), pp. 279–292. DOI: 10.1007/BF00992698.

[85] H.H. Willis et al. *Estimating terrorism risk*. Rand Corporation, 2006.

[86] D. Wilson, E.K. Roe, and S.A. So. "Security checkpoint optimizer (SCO): An application for simulating the operations of airport security checkpoints". In: *Proceedings of the 2006 Winter Simulation Conference*. Monterey, CA, USA: IEEE, Dec. 2006, pp. 529–535. DOI: 10.1109/WSC.2006.323126.

[87] P.P.Y. Wu and K. Mengersen. "A review of models and model usage scenarios for an airport complex system". In: *Transportation Research Part A: Policy and Practice* 47 (2013), pp. 124–140. DOI: 10.1016/j.tra.2012.10.015.

[88] K.G. Zografos and M.A. Madas. "Development and demonstration of an integrated decision support system for airport performance analysis". In: *Transportation Research Part C: Emerging Technologies* 14.1 (Feb. 2006), pp. 1–17. DOI: 10.1016/j.trc.2006.04.001.