UNIVERSITÀ DEGLI STUDI DI PADOVA

UNIVERSITA' DEGLI STUDI DI PADOVA

Dipartimento di Ingegneria Industriale DII

Dipartimento di Tecnica e Gestione dei Sistemi Industriali DTG

Corso di Laurea Magistrale in Ingegneria Meccanica

A Decision Support System for the Storage Space Allocation Problem under the Effect of Disturbances: a Case of the Port of Arica (Chile)

Relatore:

Ch. mo Prof. Maurizio Faccio [illegible]

Correlatori:

Dr. Jana Ries

Dr. Rosa Gonzàlez Ramirez

Laureando

Leonardo Maretto 1160016

Anno Accademico 2019/2020

Riassunto Esteso

Questo lavoro di tesi si è sviluppato nell'ambito della logistica portuale e, in particolare, dei terminal di container. Ciascun terminal può essere suddiviso in tre macro-aree: quay, yard e gate. Lo yard è il luogo deputato allo stoccaggio dei container ed al suo interno essi vengono disposti in blocchi (block); ciascun blocco è costituito da più pile (stack). L'obiettivo della tesi è quello di proporre una soluzione innovativa al Container Allocation Problem (CAP) che tenga contro degli eventi che possono interferire con il normale funzionamento del terminal. Il CAP consiste sostanzialmente nel determinare la migliore strategia di allocazione di un container all'interno dello yard: in questo lavoro non ci si è limitati a trovare una soluzione fissa al problema ma è stato possibile costruire un meccanismo (il Decision Support System: DSS) in grado di cambiare strategia a seconda dei possibili eventi

Come primo passo, è stata condotta un'approfondita analisi della letteratura scientifica esistente che ha prodotto come risultato una classificazione inedita dei criteri con cui si allocano i container nei blocchi, dei parametri di valutazione della bontà di una determinata strategia (KPI) e anche degli eventi che si possono verificare nel terminal e che possono interessarne le operazioni. È stato inoltre possibile correlare le tre classificazioni grazie al cosiddetto "matching principle": criteri, KPI ed eventi presentano tutti e tre le stesse macro-classi.

Si è poi passati all'analisi del caso studio, il porto di Arica in Cile. A partire dai dati reali e dalla classificazione proposta, sono stati sviluppati dei criteri di allocazione dei container. Tali criteri sono poi stati combinati insieme, costruendo quelli che sono stati definiti come Fuzzy System, per mezzo della logica fuzzy. Introdotta da Zadeh (1965), è una versione della logica booleana che impiega insiemi dai confini non definiti e che ben si presta ad essere applicata in un ambiente ad elevata icnertezza come un terminal di container. Un Fuzzy System è di fatto l'implementazione di una strategia di allocazione che, dato un container in ingresso, ne restituisce la migliore posizione possibile nello yard stante le condizioni attuali.

Tali Fuzzy System sono stati testati al fine di comprendere come reagissero a differenti eventi. Per fare ciò, è stato sviluppato un modello Matlab, in grado di simulare sia il funzionamento del terminal di container che la generazione di eventi dannosi per le operazioni. Il risultato di questa fase è stata la correlazione tra le varie tipologie di eventi

implementate nel modello e i Fuzzy System che davano le migliori prestazioni durante il loro svolgimento.

Da ciò nasce l'idea che è alla base del DSS: definire una strategia di allocazione dei container che scelga dinamicamente tra diversi Fuzzy System, selezionando quello che dà le prestazioni migliori a seconda dell'evento corrente.

Varie versioni del DSS sono state sviluppate, a seconda delle diverse modalità con cui si valutavano i migliori Fuzzy System. Tali versioni sono poi state testate simulando varie sequenze di eventi e comparandone i risultati con delle strategie fisse. In alcuni casi è stato possibile verificare un significativo miglioramento delle prestazioni. Inoltre, il DSS è stato anche comparato con l'attuale strategia attualmente in vigore nel porto di Arica, fornendo risultati notevolmente migliori.

Abstract

The aim of this thesis is to develop a Decision Support System (DSS) that is able to react in real time to events and disturbances that might happen in a container terminal. It is a novel approach to the real-time container allocation problem, which is mainly resolved by adopting fixed algorithms.

The DSS is based on a selector, which recognizes whether a disturbance is occurring or not and reacts accordingly, applying the most suitable stacking strategy. To define the different strategies used by the selector, Fuzzy Logic is employed: it allows to take into account the uncertainty that populates a terminal when disrupting events are happening. Considering the real-life data coming from a Chilean port, the Port of Arica, and the results of a thorough literature review, a series of stacking decision rules are developed: since they combine different criteria through Fuzzy Logic, they are called Fuzzy Systems.

A Matlab Model is developed to simulate the behaviour of the container terminal and the eventual events. After a training phase conducted with the aid of said model, the Fuzzy Systems are assembled to create the selector of the DSS. Its performances are finally evaluated during a campaign of testing where it is compared to more traditional, fixed, stacking strategies.

Acknowledgements

I would like to acknowledge my supervisors for the constant support and help provided throughout the work.

Many thanks to Prof. Faccio for having chosen me for an international experience, for the availability and the valuable suggestions.

Many hanks to Dr Jana Ries, for the interest shown towards my work, for putting up with my late submissions, even to the last minute, for all the Skype calls and for allowing my name to appear in an international conference.

Many thanks to Dr Rosa Gonzàlez Ramirez, for the constant insights, the patience and the positive attitude.

Grazie ai miei genitori, Fabio e Marina, per i sacrifici in silenzio e per avermi sopporto nei giorni preesame.

Grazie a Sara, per le rinunce e le attese, per essere nonostante ciò la mia prima sostenitrice, per essere una delle mie poche certezze e per avermi concesso di diventare amico di Argo

Grazie agli amici, per la stima, le risate e le telefonate

Index

…Take two people, romantic…

…Your memory stays, it lingers ever

Will fade away never

2SF

CHAPTER 1

Introduction

This chapter provides a brief overview on the environment of the container terminals and their characteristics. Then, it introduces the Container Allocation Problem. Finally, it states the aims and the objectives of the work, presenting an outline of the content of the following chapters.

1.1 The Container Space Allocation Problem

Seaborne trade volumes have been projected to expand at an annual growth rate of 3.8% between 2018 and 2023, and containerized trade at a 6.0% annual growth rate and world container port throughput is estimated on 752 million TEUs (Twenty-foot Equivalent Units) in 2017 (UNCTAD, 2018).

Seaports are intermodal facilities that provide transfer service of cargo. With the increasing trend of world trade volumes, seaports play an important role in the competitiveness of global supply chains. For this reason, container handling planning decisions at seaports have attracted significant attention in the literature, often through algorithms.

Subject to the time horizon under consideration, algorithmic approaches can be categorised as follows: long term, medium term, short term and real time or online (Borgman et al., 2010). Within each category, algorithmic designs may vary depending on the problem under consideration. Grötschel et al. (2001) outline the difference existing between an online optimisation problem and a real-time optimisation problem: it relates to the point of time when a decision has to be made. In the case of online and real-time, the decision has to be made before all the data are known or within very tight time frames, respectively.

As stated by Steenken et al. (2004), container terminals can be described as open systems of material flow with two external interfaces. These interfaces are the quayside with loading and unloading of ships, and the landside where containers are loaded and unloaded on/off trucks and trains. Figure 1.1 shows the general outline of a container terminal.

Between the two interfaces there is the yard, which acts as a buffer area that services both the quay and the gate. Therefore, a container terminal can be divided in three different areas: quay, yard and gate. A representation of this tri-partition of the terminal is shown in Figure 1.2.

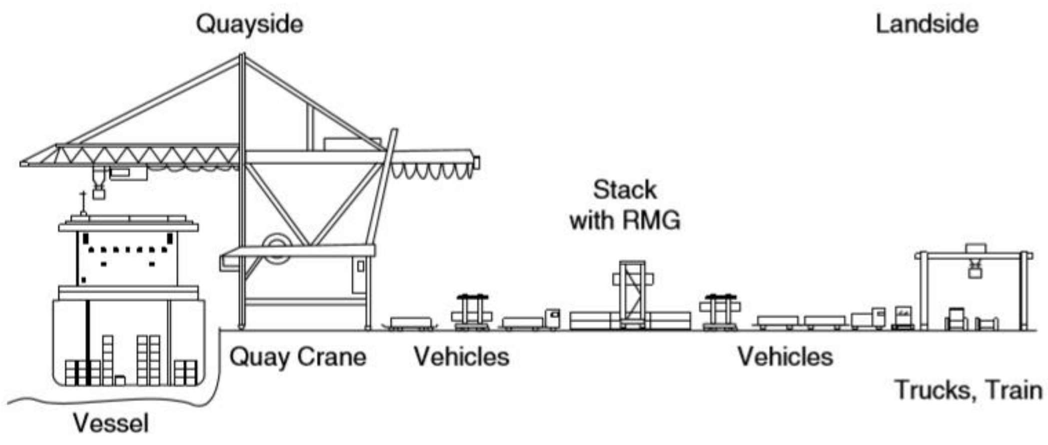


Figure 1.1 A representation of a container terminal and its equipment. Steenken et al. (2004)

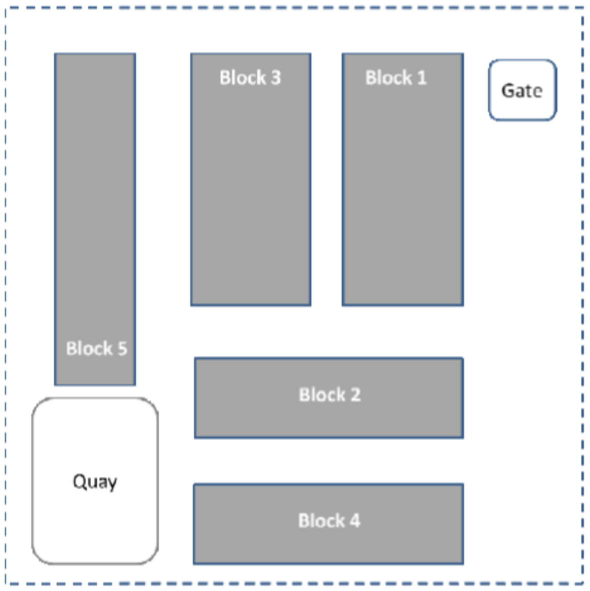


Figure 1.2 A container can be divided in three areas: quay, yard and gate. The yard, which contains the different blocks, is shown in grey. Ries et al. (2014)

This thesis considers the operations within a port container terminal in which containers are stacked on the ground and piled up vertically in the yard. The yard is divided into blocks formed by piles or stacks of containers. Each block is, in general, dedicated to either inbound or outbound containers. The location of a container is generally defined by the bay, row (or stack) and tier, also referred to as BAROTI scheme, which is represented in Figure 1.3. In this work, however, the bays, although present in the data, will not be considered since each stack of each block is identified by a progressive number. Therefore, the position of container is determined by the block, the stack identifying number and the tier.

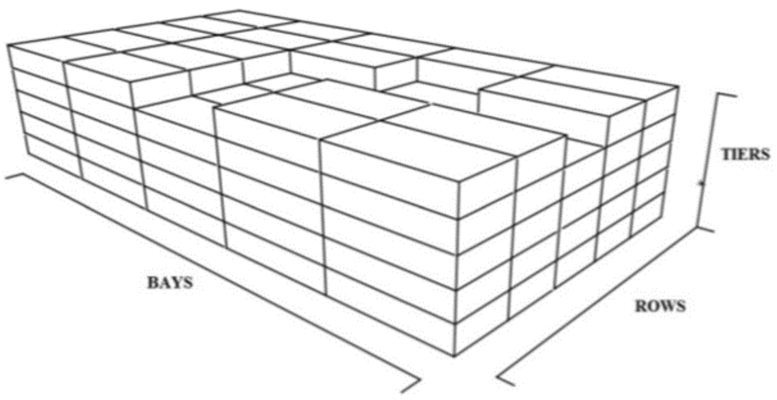


Figure 1.3 The BAROTI scheme. Guerra-Olivares et al. (2017)

Many different equipment resources find their place in a container terminal: following the flow of an import container, quay cranes (QC) unload the containers from the ships. Those containers are then transferred to various types of internal vehicles (internal trucks, multi-trailers, automated guided vehicles AGVs) that have the duty of transferring the containers to the yard. Once arrived in the yard, a container terminal is then transferred to a block by a crane. There are various types of cranes: rail mounted gantry cranes (RMG), rubber-tired gantries (RTG) and overhead bridge cranes (OBC). There are also vehicles that can move horizontally and lift a container at the same time: straddle carriers, forklifts and reach-stackers.

Several decision planning problems arise for container handling in the yard, and the strategies employed are diverse with efficiency and effectiveness depending on a variety of factors, including resource availability, infrastructure and uncertainty.

The container stacking problem (CSP) is a well-known operational problem in the yard and seeks to determine the best stacking position for a container that is arriving into a yard. The problem is solved considering specific constraints relating to the yard, the container and resources. It aims to optimize key performance indicators (KPI) such as container rehandles or reshuffles, traveling distance of vehicles operating within and outside of the yard for the horizontal transport of containers and congestion.

1.2 Aims and Objectives

Considering uncertainty and its implications on the efficiency and effectiveness of operations in container ports, this study seeks to design and apply an adaptive Decision Support System to address the Container Space Allocation problem. The design contributes to recent developments in algorithmic design, supporting the need to adjust algorithm-specific information to changes in problem-specific parameters.

More precisely, the work postulates the argument of moving from 'one size fits all' algorithm to a selecting strategy that chooses the best fitting algorithm for the problem. A decision support system (DSS) for real-time container allocation is proposed: it is able to select the most suitable container allocation strategy for an incoming container. A rule-based selector in combination with Fuzzy Logic has been implemented and computational results are presented for a real-work case. Uncertain conditions are simulated by means of disruptive events. Performance is discussed for a set of relevant KPIs, including rehandles and congestion within the yard.

The objectives of this study are:

- Exploring the diversity of decision criteria and key performance indicators (KPIs) being considered when solving the container space allocation problem
- Identifying a classification of disruptive events
- Implementation of a Fuzzy Selector to solve the Container Space Allocation problem
- Assessing the performances of the Fuzzy Selector using a real case study of the port of Arica (Chile)

The remainder of this study is structured as follow. Chapter 2 reviews recent studies using online and real-time decision support systems to solve the Container Space Allocation Problem. Chapter 3 identifies a mapping of decision criteria and key performance indicators in the existing literature, following by the integration of a mapping to potential disruptive events in a port. The case of Arica is introduced in Chapter 4. Chapter 5 details the application of Fuzzy Logic to the problem, resulting in a series of Decision Rules the combine different criteria following the Fuzzy Inference Process. Chapter 6 defines the Matlab model that has been developed to simulate the operations of the yard and the impact of the disturbances. Chapter 7 includes the computational results of the training and testing phase: the former aims at creating the dynamic DSS while the latter evaluates its performances. Finally, Chapter 8 reports the conclusions and the eventual ideas for further research.

CHAPTER 2

The Literature Review

The aim of this chapter is to present the literature outline that is related to the container allocation problem. A brief resume of the most important papers in the field of operational research applied to port logistics is presented in the first part of the chapter whereas in the second part the focus is shifted to the real-time container space allocation and the most recent papers on the subject. Finally, some considerations on the existing gaps in the literature are introduced, allowing to define the proposed framework.

2.1 Operations Research at Container Terminals: An Overview

Yard operations in container terminals is a topic of research that has been deeply studied and thoroughly examined in the existing literature. However, one aspect of it, the stacking problem, has been less scrutinized. Saanen and Dekker (2006) suggest that the reason for this may lay in the complexity of such a practical problem, not easily allowing analytical results that are relevant for practice.

Sculli and Hui (1988) were amongst the first to address the aforementioned issue, using a simulation approach that took in consideration stacking height, storage space utilisation and reshuffles. It is interesting to note that one of the very first papers that dealt with the container allocation problem focused on three parameters that are of the maximum importance in the proposed solution. Taleb-Ibrahimi et al. (1993) analysed the relation between those parameters both at a long term and operational level. In addition to this, a dynamic system to be used for real-time allocation was proposed. This strategy comprised a buffer zone, a rough pile, where the containers are supposed to be stacked before moving them to their dedicated storage are. De Castilho and Daganzo (1993) proposed two approaches that are valid for import containers and tried to estimate the number of clearing or retrieval moves based on them: the first one used stack height and avoided segregation while the other one exploited segregation based on the dwell time of the containers. However, they were not able to identify an optimal strategy. The importance of rehandles and the yard configuration is testified also by Kim (1997): the study aimed at predicting the number of rehandles depending on various stack heights and number of bays using a simulation program that applied regression equations. Kim and Kim (1998) acknowledged the influence of resources such as transfer cranes: a cost model was developed which examined the relation between available stacking space, stack height and number of necessary cranes. Another study on the importance of segregation and the link between stack height and number of rehandles was proposed by Kim and Kim (1999): a Lagrangian-relaxation-technique methodology is introduced in order to find an optimal solution. Moreover, the segregation approach offers one of the first examples of an allocation criterion: "stacking newly arrived containers on top of containers that arrived earlier is not allowed". Kozan and Preston (1999) developed a Genetic Algorithm which aimed at minimising the time ships spend at berth which is the sum of the travelling time and the retrieval time of each container from its stack. Hence, importance is given both to rehandles, that affect the retrieval time, and to distance, which defines the travelling time. Duinkerken et al. (2001) studied the container terminal in Rotterdam and proposed a simulation model that could recreate not only the yard configuration (length, width and height of the stacking area) but also the characteristics of AGVs. In addition, different stacking strategies were proposed and the issue of information on the container is discussed. Zhang et al. (2003) proposed a rolling horizon approach: the stacking allocation problem was divided in two levels, each represented via a mathematical model. The aim of the first level is to allocate containers in their respective block balancing the workload at the same time. The second level, on the other hand, aims at reducing the distance between block and berth. Therefore, this is an example of a multi-level stacking strategy which also takes distance in consideration.

The papers mentioned above represent the most historically important studies on the container allocation problem, the ones that should be considered fundamental and the foundation for everyone who approaches said problem. It is also interesting to note that many of them already introduce concepts related to criteria for the allocation of containers which is going to be the core of the current work. For a more comprehensive view on the subject, the reader is referred to the following papers: Vis and De Koster (2003), Steenken et al. (2004) and Stahlbock and Voß (2008). Each one of them offers an overview on the existing literature in the field of operations in container terminals and has a specific section where the space allocation problem is addressed.

2.2 Real Time Container Allocation: the existing literature

Many studies on the container stacking problem employ a static optimisation model, often recurring to a mathematical model able to obtain a proper optimal solution, which works using a rolling horizon. This requires knowing in advance a good degree of information about the incoming or outgoing containers which subsequently allows to put in place a form of planning or pre-planning of the allocation of containers. However, given the potential impact of internal or external disturbances, more flexibility is required in the allocation strategy. This has led to a recent interest in the application of real-time, or online, allocation strategies. For this reason, an in-depth inspection of eleven of the most recent scientific publications in the field of online systems for container stacking has been conducted: Kim et al. (2000), Saanen and Dekker (2006), Dekker et al. (2006), Borgman et al. (2010), Park et al. (2011), Ries et al. (2014), Petering (2015), Petering et al. (2017), Guerra-Olivares et al. (2017), Guven and Eliiyi (2018) and Rekik et al. (2018). Those papers were analysed under two different aspects: the combination of criteria that constitute the allocation strategy, called Decision Rule as a comprehensive denomination, and the parameters that are used to assess the performances of real-time system, which will be referred to as Key Performance Indicators (KPIs). Moreover, an exploration of the typology of events that may happen and may affect the operations of the port has been conducted, in order to have a literature-based list of possible events. The findings are presented in the following subsections.

2.2.1 Decision Rules/Policies

Kim et al. (2000) proposed a methodology to stack export containers depending on their weight via either a dynamic programming model or a decision tree. Containers are segregated into different groups depending on their weight. This segregation strategy is based on the assumption, confirmed in reality, that heavier containers are more likely to be allocated to upper tiers, above lighter containers, because this reduces the number of relocations since in the loading plan of the ship heavier containers are the first to be loaded, due stability reasons. Those groups are derived from past empirical data since the actual weight of the incoming containers is known only upon arrival. In the dynamic programming model, two concepts are introduced: stage and state. The former is defined as the number of empty slots in a bay while the latter consists of a combination of the number of empty slots in each stack and the letter representing the weight group of the heaviest container, stacked in each stack of a certain bay. An objective function is then defined, based on the number of relocations caused by the assignment of the incoming container, of a certain weight group, to a row with a defined input state. Moreover, the probability of arrival of a container belonging to a certain weight group is integrated in the equation. Furthermore, the equation is constructed in a recursive fashion: given a certain stage and state, the value of the objective function for an incoming container belonging to a particular weight group is calculated not only considering the potential number of relocations that the allocation of that container could bring but also adding the potential relocations that the new configuration (a new stage and a new state) could create. Therefore, the incoming container is positioned in the slot that has the lowest value of the objective function. The dynamic programming model, however, has proven to require a computational time that is too substantial for real-time allocation purposes. Thus, decision rules, organised in a decision tree, have been developed in order to speed up the decision-making process.

Saanen and Dekker (2006) stressed the importance of using intelligent stacking strategies as a way to deal with higher density (a synonym of utilisation) in the yard without facing a decline in performance. Moreover, they also addressed the issue of disturbances, pointing out that the allocation strategies need to face lacking or incorrect information and even stating that, according to practice, between 30 and 40% of the information regarding a container changes during its dwell time. They focused their attention on a transhipment terminal. After presenting an interesting list of rules that are valid for RTGs and RMGs terminals, some of them were chosen to be implemented in the proposed model:

- Allocation of the incoming container in a stack made up of containers that have the same port of discharge, will travel with the same ship and belong to the same weight class.
- Use of real time consolidation: this criterion consists of stacking container of the same category (combination of port of destination, size and/or weight class) in stacks that are close together. In this way, RTGs reduce their gantry travel, with the aim of increasing productivity.
- Use the workload of the RTGs, which is defined as the number of orders for a single RTG times their duration (for the following 15 minutes)
- Use of the position of the RTGs. In this criterion, the RTG allocates the container in the closest stack of the same category depending on its current position.
- Use of the expected dwell time of the containers. Another prescription related to position is proposed: containers with a short dwell time should be allocated closer to the quay
- Use of the position of the loading vessel when that information is available. Containers should be allocated closer to the expected quay in order to reduce truck driving time.

- Use of distance from the quay crane. Containers should be allocated close to the quay crane that has discharged them in order to reduce drive times.

Their model simulated the behaviour of a Terminal Operating System, and its inputs included vessel load lists, gate arrivals, berth schedule etc.

Dekker et al. (2006) worked again on categorisation, developing a category-based algorithm for containers and comparing it to the results coming from random stacking. First of all, some common rules regarding stacking are defined: they deal generally with the size of the containers, the prohibition of overhanging and the positioning of reefer containers. The random stacking strategy, used as a benchmark, works as follows: a pile (stack) that is not full is looked for in the yard. If it is empty or there are containers of the same size of the incoming one, it can be stacked there. If not, the lane is changed and the same process is repeated again. On the other hand, the algorithm is based on categories defined by weight class, destination and type of container. A specific variable is created in order to keep track of how many piles of containers exist, within a given lane, with only containers of a specific ship and category combination and an empty top position. Then, a pile not full and occupied by containers of the same category and for the same ship of an incoming one is searched for in a randomly picked lane (in order to spread the load evenly). This is signalled by the value of the aforementioned variable. If one or more piles like this exist within that lane, the program starts searching for one of these piles. When found, a container is stacked on the top of the pile. If not, the aim shifts to the next lane. If no piles are available, the container is stacked randomly. The algorithm is then enriched with other interesting features:

- Preference for ground locations. This feature aims to avoid stacking the incoming container belonging to a certain category on the top of a uniform pile of containers all belonging to a different category.
- Use of the empty pile closest to departure transfer point: When multiple empty piles are available in the same lane, the algorithm will select the pile that is closest to the point where the container will leave the stack.
- Use of the expected departure time of the containers. This feature is only used when all the piles of the same category of the incoming container are full. The container is then stack on top of another container which is expected to leave the yard later. This feature, however, does not require detailed information but only rough approximations.

- Workload control. A workload variable, the percentage of time of the current quarter that the Automatic Stacking Crane (ASC) is busy, is associated to every lane. If that variable exceeds a certain threshold, the lane is skipped in the search for a stacking position.
- Active use of the workload during the allocation process. For incoming containers, creating uniform piles takes precedence over the lowest workload. Thus, a container will be stacked on top of a uniform pile of the same category even if the ASC for that lane is very busy. However, if there are uniform piles in multiple lanes, then the lane with the lowest ASC workload is selected.

Other fewer interesting features were implemented regarding the exchange of containers from different lanes and specific procedures for reefer containers.

Borgman et al. (2010) produced a very important contribution in the field of online container allocation. They set up a series of experiments simulating a 15-week period with a yard modelled on the ECT Delta Terminal at the Port of Rotterdam. The yard is organised in lanes that are perpendicular to the berth site and each lane has two transfer points: one seaside and one landside. They started with setting up two benchmark algorithms:

- Random stacking: the incoming container is placed at a randomly chosen position, if allowed. The sequence starts with the random selection of a lane, followed by a random selection of a position in that same lane, then the availability of that position is checked: if positive the container is allocated there, otherwise the lane is changed and the sequence starts over.
- Levelling: the concept is to fill the yard in layers, ideally occupying all the ground locations before stacking container one upon the other. The process starts with the selection of a random lane with at least one available position; if true, the container is allocated in the first empty ground location, as close as possible to the seaside transfer point; if untrue, the container is stacked on the lowest existing pile of containers of the same size and type within said lane, with preference given to stacks close to the transfer point landside.

Moreover, other variations of the two benchmark algorithms were created for the purpose of comparison:

- RSDT (Random Stacking with Departure Times): it is a modified version of the random stacking. The algorithm searches for a random pile with a dwell time of the top container longer than the dwell time of the incoming container. If no such pile is found, the incoming container is stacked entirely in a random way, following the traditional random approach;

- RS-DTC (Random Stacking with Departure Time Classification) an algorithm very similar to the RSDT, with the only difference being the use of departure time classes instead of actual departure times;
- TPRL (Transfer Point Random Level): is an algorithm that finds the available spaces closest to the transfer point for every tier. After that, one of those positions is chosen randomly.

After that they introduced 6 different stacking strategies, based on various criteria, and developed five experiments in order to test and compare them to the benchmark algorithms. Those strategies are:

1. LDT (Levelling with Departure Time). This algorithm combines the use of predicted dwell time, distance, stack height and block utilization following this procedure:
    a. The incoming container is stacked on a pile where the top container departs later and the difference between the two departure times is minimal;
    b. If no such position is found, the incoming container is stacked on an empty ground location. (In case the container is sea-sea, it should be stacked as close as possible to the transfer point at seaside);
    c. If no empty ground position is found, the incoming container is stacked at the highest available pile (in order to reduce the number of reshuffles), as close to the transfer point as possible.
2. LDT-DTC (LDT with Departure Time Classification). The algorithm is almost identical to LDT. The difference regards how uncertain departure times are treated: containers are segregated in five different classes (from 1 to 5) by ascending residence time. The classes boundaries are calculated taking the quintiles or the 20th, 40th, 60th, 80th and 100th percentiles of the residence time. Therefore, at the step when time differences are calculated, classes are used instead, placing containers belonging to a lower class on top of containers of a higher class.
3. TVR (Travelling distance Versus Reshuffling). In this algorithm departure time is not taken into consideration but the focus is on finding an optimal position, defined as a trade-off between proximity to the transfer point and the number of potential rehandles that this location might cause. Stacking close to the transfer point, in fact,

is preferable because containers are close to where they depart; however, this might create an-extra number of reshuffles. On the other hand, creating lower piles might reduce the number of rehandles but containers might end up too far away from their transfer point. To solve this issue, the cost in time of every position is calculated as a combination of two time costs: extra ASC driving time and time due to reshuffling. The first one is a sum of the time spent moving the container from the transfer point to the selected position and vice versa and the relative lifting times. The latter is a sum of the estimated travel and lifting times caused by one reshuffle multiplied by the expected number of reshuffles generated by one container. This number is calculated without using dwell times but as a simple probability of extra rehandles caused by the incoming container which is a simple function of the occupied tiers in a given stack: the higher the stack the more probable it is to have additional rehandles. The algorithm then works in the following way:

   a. For every tier (called stacking level in the paper) the closest available position to the transfer point is found.
   b. For every one of these positions, the time cost is calculated.
   c. The container is allocated in the position that has the lowest total cost.

4. TVR-PA (Peak Adjusted TVR). This algorithm addresses the problem of big peaks in the workload of the cranes and is valid for blocks that accommodate containers bound for different destinations (sea-sea, sea-land and land-sea). The idea of placing sea-land and land-sea containers near the transfer point landside because distance is not important for them (since they have to move along the lane of the stack anyway) is valid only if the value of the time spent by the cranes moving along the lanes in always the same. However, this cannot be true during peaks of workload. To solve this, an extension of TVR is proposed: every lane is divided in two parts, one for sea-sea containers and on for all the others. When it is the moment to consider distance in the algorithm, every container is stacked as close as possible to the transfer point quayside but each type in his own part of the lane. The size of the two parts is a parameter than can be adjusted.
5. TVR-DTC (TVR with Departure Time Classes). With this algorithm, knowledge on residence times is used again, combined with TVR. Departure time classes are included in the TVR algorithm and used to calculate the expected number of reshuffles in a more precise way: if the departure time class of the incoming container is lower than the earliest class of the pile, the probability of reshuffles is set to zero.

If it is higher, the probability is set to one. If the classes are equal, the probability is determined through a function of the number of containers of the same class within that stack. With this additional information, the closest position to the transfer point might not be an optimal solution anymore, thus the need to calculate the time cost for every available position and not only to the ones within the proximity of the transfer points.

6. TVR-DTC-MD. This algorithm is an extension of the previous but also adds a feature: the minimisation of the difference between departure time classes.

The results showed that the algorithms that use either departure time classes or expected departure times tend to perform well, with performance indicators showing similar values. This indicates the importance of using even partial information. The TVR strategy also proved to be a good strategy, outperforming the random and levelling approaches. However, no significant improvement was obtained through the workload peak-adjusted algorithm in comparison to the standard TVR. Another important finding was that trying to minimise the difference between departure time classes of containers belonging to the same group is crucial to achieve good performances.

Park et al. (2011) proposed probably the first attempt to generate a stacking strategy that is not fixed but changes dynamically. This is done through a so-called Dynamic Policy Adjustment, which will be explained later. Focusing on an Automated Container Terminal (ACT) where the blocks are laid out perpendicularly to the seaside, with two ASCs for each block, the Decision Rule for container stacking is defined via a two-stage model: the first step consists in finding the best block for allocation while the second step regards stack assignment:

1. For Block Assignment of an incoming container, the workload distribution of the ASCs among the blocks in the yard is considered. The workload of ASCs associated with a certain block, at a given time and for an incoming container is defined as a weighted sum of two components: the short-term workload for that block (which consists in the number of containers scheduled to be handled by the ASCs dedicated to that block in a defined time interval) and the future workload at the block (defined as the estimated number of containers that belong to the same group of the incoming one already allocated in the block; this is based on the assumption that a segregation of the containers in groups has been put in place considering weight, size and

destination and that containers belonging to the same group are likely to be retrieved together around the same time, causing delays in ASCs operations). The incoming container is allocated in the block with the lowest workload.

2. For Stack Assignment of an incoming container, all the stacks in the block selected through the previous procedure are evaluated using an evaluation function which is a weighted sum of four elements:
    a. The stacking cost of storing an incoming container at a specific stack. It depends on the distance between the stack and the transfer point where the incoming container arrives and the delay caused by interference of the two ASCs that work in the same block. (the larger the distance from the transfer point the more probable is the interference because the landside ASC, which receives the incoming container, is more likely to meet the seaside ASC). The stacking cost is then modelled as a linear function of the distance between two different thresholds: below the lower distance threshold, the cost function is set at 0 since the interference between the ASCs is negligible while above the upper distance threshold the cost function is set at 1.
    b. The expected retrieval cost of retrieving the incoming container from the stack where it is going to be allocated. The retrieval cost is modelled exactly in the same way as the stacking cost, with the same dependence on distance and interference and the same thresholds.
    c. The need for rehandling. The relative parameter in the weighted sum depends on the group ID of the incoming container and the group ID of the containers already part of the considered stack, where the groups are defined as above. For export and transhipment containers, the parameter is set at 0 if the incoming container belongs to the same group of all the containers in the stack (since there are supposedly no constraints in the order of retrieval of containers of the same group which means that they can be retrieved in no specific order) or if the incoming container is expected to be loaded on a vessel which is scheduled to arrive earlier than the ones of the containers already in the stack. In all the other cases is set at 1. For import containers, the parameter is ignored because the authors suppose that it is not possible to know the arrival time of the external trucks, making it impossible to predict the need for rehandles.

d. The waste of space in a stack. This parameter is a function of both retrieval times and the height of the considered stack. The waste of space is calculated as the product of two ratios: the first one is defined as the remaining height of the stack after allocating the incoming container divided by the maximum available height of the stack itself while the second one is defined as the difference between the earliest retrieval time of the containers already in the stack and the retrieval time of the incoming container, all divided by the earliest retrieval time of the containers in the stack. The concept of waste of space is explained in the following way: the remaining height mentioned above is wasted for the aforementioned time difference. In fact, in order to avoid rehandles, only containers that depart earlier than the incoming one are allowed to be stacked of top of it, thus preventing all the containers that have a later delivery time to be allocated in the given stack and "wasting" the remaining stack height; seeing this from another angle, the shorter the time difference, the more containers can be allocated in the remaining space of the stack. This parameter can be used for both export and import containers, using for the former the scheduled arrival of the vessel while for the latter the average dwell time of the containers.

The container is then allocated in the stack with the lowest weighted sum.

The weights of the two weighted sums (one for Block Assignment and one for Stack Assignment) are determined through an algorithm called Dynamic Policy Adjustment (DPA). The algorithm generates a weight vector and applies it for a given period of time, evaluating the results of its application over a defined evaluation period. Once the application period is over, a new weight vector is generated from the best-so-far weight combination using a Gaussian mutation operator. The new weight vector is then applied for the same application time described above and evaluated during the evaluation period. The best-so-far vector is adjourned every time a new weight vector outperforms the current best combination. The performances of the weight vector are evaluated using a weighted sum that comprises the Quay Crane (QC) delay time, the AGV waiting time and the external trucks waiting time.

Ries et al. (2014) proposed a stacking strategy which aims to account for a high degree of uncertainty in the arrival of containers. The two-phase framework consists of a Stack

Assignment approach that is preceded by Block Assignment. Both processes are based on fuzzy logic. The process of Block Assignment is based on two criteria:

1. Block-Gate Distance: it is the distance between the external trucks entry point in the yard and the block under examination.
2. Block Utilization: it is a measurement of the space usage in the considered block. It is defined as the ratio between used allocation spaces and the available allocation slots in a single block.

The two criteria are then modelled as two input variables of a fuzzy inference system. For each one of them, three different subsets are created where each subset is a membership function which transforms crisp numerical information on the variable (for example, the distance of each block) into linguistic terms such as small, medium or high. The two input variables are then combined using a set of specific fuzzy rules in order to determine the value of the output variable of the fuzzy inference system. This variable is defined as the Value of Goodness of the Block, which is a way to assess the validity of stacking the incoming container in a given block. Therefore, the block with the highest Value of Goodness is chosen as the destination for the incoming container. For Stack Assignment, which represents the second phase of the framework, the procedure is very similar and is done for all the stacks in the block chosen in phase 1. It takes into consideration two different criteria:

1. Stack Height: it defines the current height of each stack.
2. Estimated Time of Departure: it is defined as the normalized difference between the estimated time of delivery of the container on top of each stack and the estimated time of delivery of the incoming container.

Exactly in the same way as for Block Assignment, the two criteria constitute the input variables of a second fuzzy inference system, where they are combined using a set of fuzzy rules which have the target to determine the value of the output variable, called Value of Goodness of the Stack. The stack that has the highest Value of Goodness is chosen to accommodate the incoming container. The framework also includes a slight variation of the Block Assignment to be used for relocated containers: when a container is to be retrieved from a stack and there are other containers on top of it, it is possible to examine the possibility of moving those blocking containers to other blocks by using the framework and evaluating the Value of Goodness of each block. In this case, instead of the Block-Gate Distance, the distance between the current block and other blocks is evaluated.

Moreover, in order to test and compare the performances of the proposed fuzzy framework, other stacking strategies were used. They were taken from Borgman et al. (2010): RSDT, LDT and slight variations of Random stacking and Levelling.

Petering (2015) focused on a land-scarce, transhipment terminal and developed a real-time stacking strategy for the incoming containers. The work is interesting because it also modelled a series of discrete events that are related to the transhipment operations, evaluating their effects. Before describing how the proposed system works, some necessary considerations and assumptions need to be made. The author defined that the proposed stacking policy follows the definition of homogeneous or sort-and-store strategy which means that containers are segregated in groups according to their weight, height, length, liner service on which they are supposed to be loaded and their port of destination. In this way, during the loading phase of a vessel, containers that belong to the same group are virtually interchangeable: the aim is to create stacks made of containers that belong to the same group, avoiding the need for relocations. This leads to the definition of two categories of containers: trailblazing and non-trailblazing: the former are incoming containers assigned to empty stacks because there are no existing stacks belonging to their same class, the latter are incoming containers assigned to partially full stacks in the yard comprised of containers belonging to their same group. Other important parameters and definitions that are used in the model are listed below:

- Simult20: it indicates the possibility of having two 20' containers unloaded from the ship onto the same yard truck. In this case the location of both containers is determined at the same time once they are both on the truck.
- Threhsold40: it is a lower bound that indicates the minimum number of empty slots dedicated to 40' containers in a block. In fact, a 20' container occupies one empty slot at the base of a stack while a 40' container occupies two adjacent slots. Once the number of empty slots falls below this threshold40, specific 40' stack conservation measures are put in place. Those measures imply forbidding new incoming 20' containers from occupying the slots for 40' containers. This allows to keep enough space to accommodate new 40' containers in the yard.
- Dispersion level: it measures the dispersion level in the yard of containers that are supposed to be loaded onto the same vessel. A high dispersion level means that containers that are supposed to be loaded onto a vessel are stored in a rather dispersed fashion from the home berth of said vessel: many containers are stored many columns

away from the berthing site. Hence, considering the dispersion level as a parameter, the higher it is the more sites and columns are available for allocation of incoming containers.

- Yard template: it accounts for the forms of pre-planning that are in use in the port. Usually the yard template is generated offline thanks to a mathematical modelling program and it serves as a guide for real-time allocation, defining a set of preferred stacks in the various blocks for containers belonging to each group and bound for each different vessel.
- Penalty weights: the weights of the weighted sum that constitutes the penalty score of each block. The meaning of the penalty score will be described later.

Given those necessary definitions, the real-time stacking algorithm works in the following way:

1. Checking whether the incoming container is trailblazing or not which means looking for stacks of the same group the of such container. When it is not, it needs to be stacked on top of a stack of containers that do not belong to the same group of its and the rest of the algorithm is unnecessary. On the other hand, when the container is trailblazing, the algorithm can continue.
2. If Simult20 is set at the logical value of TRUE and if the incoming container is the first of the pair of two 20' container that are about to be loaded onto the same yard truck, its location assignment has to wait until the second container is placed on the truck. In that moment, the stacking location of both containers is determined simultaneously
3. Evaluation of the length of the incoming container (20' or 40') and research of the empty spots dedicated to that length, compiling a list of candidate stacks.
4. If the incoming one is a 20' container and the empty slots for 40' containers are below threhsold40, all the empty slots that are part of a 40' dedicated stack shall be eliminated from the list of candidate stacks.
5. All the stacks that do not agree with the dispersion level associated with the incoming container and its group shall be eliminated from the list of candidate stacks.
6. All the stacks that do not agree with the yard template currently in use shall be eliminated from the list of candidate stacks.

7. Calculation of the penalty score for each block, considering the real-time conditions of the yard. The penalty score of each is then assigned to all the available stacks in that given block.
8. If two or more stacks have the same penalty score, the tie is broken randomly. Once a stack is chosen, it becomes the storing site for the incoming container.

The penalty score is a concept which is not far away from the opposite of the Value of Goodness of a block introduced by Ries et al. (2014): it is defined by the authors as the undesirability of storing a container in a certain block; the higher it is for a block, the worse it is to store a container in that block. The penalty score of a block is a weighted sum of ten penalty components:

- The total distance from the unloading berth to the block
- The horizontal distance from the unloading berth to the block (the horizontal distance is the distance travelled moving alongside the berthing site, without going in depth into the blocks)
- The total distance from the block to the loading berth (the berth where the incoming container is going to be loaded onto its outbound vessel at the end of its dwell time. It is usually based on a prediction)
- The horizontal distance from the block to the predicted loading berth
- The difference between the location of the block and the "ideal zone" of the container (the "ideal zone" is a concept that derives from a Duration-Of-Stay stacking policy in which containers with a short dwell time are supposed to be allocated close to the entrance/exit point while containers with a long dwell time are moved much farther away. The ideal zones for the containers are then determined depending on the predicted dwell time of the incoming containers)
- The number of yard trucks that are currently directed towards the block
- The number of containers that are currently travelling towards the block (this component counts the actual containers while the former counts just the truck: a truck carrying two 20' containers accounts for one the former component and for two in the latter)
- The ratio between the yard trucks directed towards the block and the yard cranes present in the block itself
- The ratio between the container travelling towards the block and the yard cranes present in the block itself

- The forecasted retrieval clashing in the block (the retrieval clashing is defined as the weighted sum of the overlap between the vessel where the incoming container is scheduled to be loaded and the different liner services associated with the other stacks in the considered block, where an overlap is defined as the number of minutes in which two or more liner services are present contemporarily at the port).

The weights of the penalty components in the penalty score are calculated through a calibration process.

Petering et al. (2017) used the same real-time stacking algorithm for another study. In this case, four different experiments were developed in order to address different issues:

1. Investigate the impact of the dispersion level on the Gross Crane Rate (see next subsection for the definition)
2. Investigate the effect of two different yard templates
3. Investigate the impact of the different berthing policies of the vessel on the Gross Crane Rate
4. Investigate the effect of different travelling speeds of the yard trucks

Guerra-Olivares et al. (2017) developed a two-phase allocation strategy while focusing on export containers and a yard where reach stackers are in operation. The first phase of the two is not a real-time allocation policy but is a mathematical model that works offline. This model, developed and presented in Tapia et al. (2013), works with the usual concept of container segregation, where the containers are separated into different groups according to their weight, size, ship and port of destination. Its outcome is the association between every container group and a bay in the yard: the bays in which the containers of a certain group are going to be allocated is defined. However, the actual final location, in terms of row and tier, is still unknown for all the containers. This is solved by the proposed heuristic which is based on segregation related to weight: containers are segregated into different categories according to their weights. Each category is given a number: the lower the number the lighter the containers. All the containers that belong to the same category are interchangeable in terms of allocation strategy. The idea is to store heavier containers on top of the lighter ones in the yard. This is done because the stowage plan of the vessels onto which the containers are going to be loaded, usually require heavier containers to be placed below the lighter ones, for the sake of stability. Therefore, heavier containers are required earlier than the lighter ones during the loading operations of the outbound vessel and if the weights are not taken into consideration, rehandles become a necessity. Therefore, the algorithm of the heuristic puts a great importance on the difference between the weight category of the last container that has been allocated in a given bay and the weight category of the incoming container. The incoming container is allocated in the bay in which this difference is negative, meaning that the incoming container is heavier than the containers already in the bay, thus likely reducing retrieval moves, and its absolute value is minimal, so to allocate the incoming container in the bay with the most similar weight category. When this difference is positive for all the already half-occupied bays, containers are either placed in an empty bay or, when that is not possible, are placed in the bay with the most similar weight category. In addition to the heuristic, the authors also developed a mathematical model which serves a lower bound to compare the test the validity of the heuristic. The mathematical model works on the assumption, not verified in reality, that the arrival sequence of the containers in the yard is known in advance.

Güven and Eliiyi (2018) developed a mathematical model to be used in an online fashion, based on the case study of the Port of Izmir, Turkey. Considering a mathematical model, which is usually run offline, might seem an odd choice given the target of the work. However, the authors of the paper state very clearly that the model is run dynamically, container by container as they arrive in the yard, through an algorithm because of the characteristics of the environment under examination, which changes quickly and very often. Some important assumptions were made:

- The size of the containers can be 20’ or 40’
- Containers of different sizes cannot be stacked on top of each other or not even placed in the same bay
- The maximum height of a stack is 4 tiers and the maximum difference in terms of weight between the heaviest and the lightest container in the stack should be below 3 metric tons. This constraint comes from guidelines of the local port authorities that put them in place in order to avoid damages to the containers in the lower tiers.

The categorisation is, again, an important part of this model: the containers are group into different categories according to their size (20’ or 40’), their trade type (import, export/transhipment, empty) and their destination (a vessel for export containers, the receiving company for import containers and the owner of the containers for the empty ones). Two important definitions are introduced:

- Sub-optimal stacking position: an available stacking position in the yard in which containers have the same size of the incoming one and where stacking the incoming container does not mean breaking the weight constraint (maximum difference of 3 metric tons)
- Optimal stacking position: an available stacking position that has the qualities of a sub-optimal position but also where stacking the incoming container does not increase rehandles. Avoiding rehandles can be obtained in three separate ways: the incoming container belongs to the same category of the containers already in the stack (which means that, belonging to the same category, they are supposed to leave the yard at the same time in no specific order), the stack is empty or the incoming container is supposed to leave the yard earlier than the container currently on top of the existing stack.

The authors also made the assumption that, at any given moment, that there is at least one optimal or sub-optimal position in the yard for each incoming container. Using the concept described above, three different stacking policies were developed:

1. Random stacking: a random stacking position is selected in the yard. If this position is sub-optimal, the container is stacked there. If not, another stacking position is randomly selected and its characteristics are compared to the requirements for a sub-optimal position. This process goes on until a feasible position is found
2. Attribute-Based Stacking (ABS): it corresponds to the implementation of the mathematical model. In this procedure, each lane is checked first, in ascending order. If one lane stores containers of the same trade type of the incoming one, then its bays are checked in ascending order. If one bay has containers that belong to the same trade type of the incoming container, the focus then shifts on its stacks which are checked, again, in ascending order. If one stack stores containers of the same category of the incoming one and the weight constraint is respected, then the incoming container can be stored in that stack.
3. 3-Tons Relaxation: it corresponds exactly to the ABS policy, only without applying the weight constraint. It is developed in order to have a lower bound for the proposed algorithm.

Rekik et al. (2018) produced probably the most interesting paper with comparison to the subject and the scope of this thesis. The authors developed a Case-Based heuristic for online container stacking which is able to address and react in real-time to unexpected events and disturbances. This can be considered the most similar work to this thesis currently existing in literature, especially regarding the interaction between events and stacking strategy. At the basis of the heuristic there is the so-called Case-Based Reasoning (CBR) which is defined as a form of Artificial Intelligence which uses information drawn from past experiences to solve problems: the solution of each current problem is obtained through the adaptation to the current environment of previous well-performing solutions to similar problems. In CBR, problems are known as cases and they are stored in a database named case base. For a container terminal, each case illustrates the "Situation" of an incoming container and is defined as a vector of three elements:

- Knowledge, which is made up of three components itself
    - Knowledge about the incoming container: it comprises the container ID, its origin and its destination, its entry date and its expected time of delivery and its type (which can be regular, open top, empty, tank, reefer or dangerous). In the case of a container belonging to the dangerous type, which requires extra focus and attention during handling, the class of dangerous goods is also recorded.
    - Knowledge about the yard: it comprises the number of containers already allocated and the type of each one of them at the moment of arrival of the incoming container.
    - Knowledge about Events and Disturbances. Each event is described by its type: allocation of a container, retrieval of a container and a disturbance event. When the event type corresponds to a Disturbance, two other attributes are introduced: the type of disturbance (which can be container related, resource related or equipment related) and its severity.
- Decision: it contains all the elements that are involved in finding a stacking location for each incoming container. First of all, the rules used for Block Assignment, Bay Assignment and Stack Assignment are recorded. Secondly, it records the final position of the incoming container in terms of block, bay and stack
- Performance: it is represented by a Boolean value, 1 meaning that the combination of stacking rules proved to be a good decision and 0 if it proved to be a bad decision.

The idea is to gather as much information as possible for each incoming container and to compile its Knowledge vector, called "Sit", which is a representation of the situation in the yard at the moment of the arrival of the incoming container. Once that is done, “Sit” vector is compared with all the other cases stored in the case base with the aim of finding the most similar case and adopting the same Decision, which is to say the same allocation strategy. In order to compare “Sit” with each one of the vectors, the distance between them is calculated. The Case-based heuristic is based on the concepts described above and works with a two-step methodology, in which the first step consists in using CBR to find the best stacking rule and the second step consists in applying that rule to stack the incoming container. With more detail, the algorithm is the following:

1. After gathering information about the incoming container and the yard, “Sit” vector is computed and it is compared in terms of distance with the cases with a Good Performance (Boolean value of 1) in order to check if the situation has happened before.
2. If the situation has already been encountered, the distance between “Sit” and the respective case is zero and the algorithm immediately skips to point 5. Otherwise, the closest case to “Sit” in terms of distance is retrieved and its Decision attributes are combined with “Sit”, forming a new vector. This new vector is then compared with the cases (Knowledge + Decision) that gave a Bad Performance (Boolean value of 0) by calculating their distance.
3. If the distance calculated at step 2 is below a certain threshold, the Decision is considered bad and discarded.
4. Steps 1-3 are repeated until a case with a distance above the threshold is found. If no such case exists, the most similar one (to the “Sit” vector) and its Decision are chosen
5. A stacking position for the incoming container is determined by using the rules for Block Assignment, Bay Assignment and Stack Assignment stored within Decision of the chosen vector.
6. The validity of the Decision is evaluated by calculating a function that the authors called Performance Index. It is a weighted sum of four components: block-gate distance, queue in front of the chosen block, stack-gate distance, remaining stack height. The lower the value of the sum is, the better the performance. The weights are chosen according to the type of the incoming container.
7. The Performance Index is compared with a specific threshold: if it is above it, then the case is considered Good and stored as such in the case base. Otherwise, the case is stored as Bad.

Unfortunately, the authors did not seem to describe which stacking strategies and decision rules were implemented within the case-based heuristic. They only provided a categorisation of stacking rules:

- Block stacking rules which include Role separation of blocks (each block is assigned only to inbound or outbound containers), dedicated areas, Different Priorities on Blocks for Different Berths, Role Separation of Bays, Maximum Number of Internal Trucks and Road Trucks in a Block, No Restriction.
- Bay stacking rules which include Concentrated Location Principle (assigning containers to non-empty bays even if they are from different groups) and Sequence Rule (selecting an empty bay for incoming containers)
- Slot stacking rules which include Levelling Rule, Random Rule, Maximum Remaining Stack Height and Closest Position (choosing the stack with the closest position among candidate stacks).

The model was developed and tested using real data coming from King Abdul Aziz Port in Dammam, Saudi Arabia.

2.2.2 Key Performance Indicators (KPIs)

Kim et al. (2000) did not provide absolute results from the model but only a comparison between the performances of the decision tree and the optimal solution given by the dynamic programming model: the number of wrong decisions, taken by the decision tree, varies between 1.0% and 5.5%. Hence, the decision tree can be used as an online tool. However, it is possible to consider the number of relocation movements as the main KPI since the model was built explicitly to reduce them.

Saanen and Dekker (2006) drew a final comparison in terms of moves per hour performed by the quay crane between the proposed stacking strategy and a more traditional (for transhipment terminals with RTGs) random assignment: the two models gave very similar results but the random approach was more sensitive to variations of overall yard utilisation.

In Dekker et al. (2006), the proposed algorithm and the random stacking approach were tested and compared on the grounds of reshuffles occasions (defined as a situation when one or more rehandles are required in order to retrieve a container that is supposed to leave the yard), total number of rehandles, workload of the ASCs and the degree of occupation of empty ground locations, with the category stacking strategy yielding better results in the first three KPIs but likely decreasing the percentage of empty piles.

Borgman et al. (2010) used four different KPIs to compare the performances of their proposed stacking strategies:

- Exit Time: it is defined as the time in hours that it takes to remove a container from the stack and have it ready for transport (either at quayside or landside). It is the main performance indicator and is influenced by the distance of the stacking position, the eventual number of reshuffles and the workload of the ASCs;
- ASC workload: the percentage of time in which an ASC is busy;
- Reshuffles: the number of reshuffles are measured as a percentage of the total number of container movements. Moreover, reshuffle occasions are also taken in consideration: a reshuffle occasion happens when one or more reshuffles are needed to retrieve one departing container;
- Ground Position Usage: the average of the percentage of ground locations that are occupied at each given time;

Park et al. (2011) compared their proposed DPA (Dynamic Policy Adjustment) against nine other different stacking strategies. Each of those nine used the same approach for Block Assignment, which is the one described in the previous subsection for the current paper, while they differed for Stack Assignment, where each strategy employed a different subset of the four criteria that are part of the weighted sum that evaluates the goodness of each stack. One of the strategies used the complete set of criteria for Stack Assignment while employing a fixed weight combination. The different strategies were compared on the grounds of three different Performance Indicators: Quay Crane delay, AGV waiting time and external trucks waiting time. The DPA, while increasing the waiting time of the external trucks, obtained a 4.7% reduction of the Quay Crane delay, which the authors judged as a significant improvement in quayside productivity.

Ries et al. (2014) used two different Performance Indicators to test the validity of the proposed fuzzy framework:

- Distance: it is measured as the distance travelled by each container that is retrieved from its block of residency to the gate which is the landside exit. To each of those distances it is required to add the eventual distances of the relocated containers in case of the need for rehandles.

- Relocation ratio: it is defined as the ratio between the effective relocations and the total moves during the rehandling phase (which is the sum of relocations and the number of efficient/effective moves where the latter are the moves that effectively lead to the retrieval of the required container).

The results show that the fuzzy framework does not outperform all the other staking strategies proposed but the performances are still top-tier and, above all, show a very low variability.

Petering (2015) measured the performances of the proposed real-time stacking algorithm using the Gross Crane Rate (or GCR), which is defined as the average number of lifts performed by a Quay Crane during one working hour. For a container terminal, a higher GCR means serving more vessels while for the ship liner it means spending less time at a berth. The author postulated that the target of maximizing the GCR can be achieved by pursuing four different subobjectives:

1. Minimizing the quay-yard distance travelled by the container during unloading
2. Minimizing the yard-quay distance travelled by the container during loading (once the container has to leave the yard)
3. Minimizing the congestion of the yard trucks in the proximity of the stacking site during unloading
4. Minimizing the congestion of the yard trucks in the proximity of the stacking site during retrieval

Subobjectives 2 and 4 were considered to be more difficult to achieve since they require to work on forecasted data. Moreover, the author states that it is not possible to pursue some of the subobjectives simultaneously. The results show that the proposed algorithm can achieve a 1-7% improvement in terms of CGR compared to the results of a random stacking strategy. Furthermore, a deeper analysis of the results showed that subobjectives 3 and 4 are more important than 1 and 2.

Petering et al. (2017) used again the Gross Crane Rate as the main KPI in their work and also the four related subobjectives. The findings coming from the four experiments they conducted show that a highly dispersed allocation strategy is superior to a more condensed distribution of containers belonging to the same groups (which contradicts subobjective 2 which aims at minimizing yard-quay distance during loading operations). Secondly, using a yard template does not improve performances and using a normal real-time decision without any sort of pre-planning is equally good. Another interesting result is the fact that slower yard trucks impact performances in a sensible way, reducing the advantage of pursuing each one of the four subobjectives. Finally, a ranking of the four subobjectives in terms of importance can be drawn: 3, 4, 1, 2. This means that, generally, minimizing congestion is more important than minimizing the travelled distance.

Guerra-Olivares et al. (2017) used the ratio of rehandle movements as the main Performance Indicator in their work. They defined the ratio in the same way as it was defined by Ries et al. (2014): the number of rehandle movements (or more simply, rehandles) divided by the number of total moves which is the sum of rehandle and efficient moves. The paper is also interesting because it clearly defines the way in which the rehandling effort is calculated. In order to retrieve a container which is located at the bottom of a stack, it is required to move the containers that are placed on top of it and to then move them back to the stack: when calculating the number of rehandling moves, only the retrieving ones are considered while the moves needed to place the containers back are neglected. Moreover, the containers are moved back to the stack in the same order or configuration in which they were prior to the retrieval. The results of the heuristic were compared to an adaptation of a previous work by Chen and Lu (2012) where the original RTGs were substituted with reach stackers, showing a better performance. Moreover, the comparison with the mathematical model showed gaps in terms of relocation ratio that variated between 0 and 42.5%.

Güven and Eliiyi (2018) used four different Performance Indicators in their paper in order to test their ABS stacking policy:

1. Total Number of Rehandles
2. Total Number of Reshuffle occasions. (A reshuffle occasion is defined as in Borgman et al. (2010))
3. The distance travelled by an empty Automatic Stacking Crane, which is considered an unproductive feature.
4. An estimation of the total time lost due to reshuffles and empty travels performed by an empty Automatic Stacking Crane

The results showed that the ABS policy outperformed sensibly the random stacking approach, with a reduction of the total number of reshuffles of almost 85%. Moreover, the 3-Tons Relaxation policy showed an improvement in comparison to the ABS, which lead the authors to question the validity of the weight constraint.

Rekik et al. (2018) defined an Average Performance Index to evaluate the validity of their Case-Based heuristic. This represents the only example found in literature of a combined Performance Indicator, which is a KPI that takes into consideration multiple Performance Indicators at the same time. Firstly, a Performance Index is defined for each incoming container as the weighted sum of four components:

- Block-Gate distance (considering the block where the incoming container is allocated)
- The queue of containers in front of the chosen block
- Stack-Gate distance (considering the stack where the incoming container is allocated)
- The remaining stack height

Then, the Average Performance Index is calculated by Averaging the Performance Index across all the incoming containers. The weights of the sum depend on the type of the incoming container, as stated in previous subsection. The heuristic proved to be reliable and outperforming other stacking approaches taken from the existing literature.

2.2.3 Events and Disturbances

Without having the chance to talk with port managers, in order to examine what kind of events might happen in a container terminal and they affect operations, a literature analysis has been done. In particular, some of the most recent papers were reviewed to find description of events, list of possible events or a description of the effect of disturbances. The results of this review are presented here.

Saanen and Dekker (2006) stressed the importance of information about the container flow and in particular its availability and quality. There might be issues regarding information, especially when it is late, of low quality or completely missing. Moreover, information is not stable over time but it might change. The authors estimated that between 30 % and 40 % of the available information might change during the dwell time of a container in the yard. In addition to that, they stated that the quality of information also depend on the type of container terminal: in a transhipment terminal, in fact, information is considered to be 50% better than in a more traditional import-export terminal since the final destination, the vessel and possibly the scheduled departure are already known upon a container arrival. Therefore, stacking decision rules have to adapt to such an environment where information is not complete and is prone to change. The authors also listed a very brief set of dynamic effects that might have an impact on the port: delayed vessels (with no reason stated for this sort of event), arrival of external trucks without pre-notice and late arrival of information.

Meydanoglu (2009) did not focus specifically on container terminals and the relative stacking strategies but worked on a more general level, evaluating the impact of SCEM (Supply Chain Event Management) systems support risk management in a supply chain. SCEM systems monitor, record and evaluate the impact of disruptions of the supply chain in real time, aiding the decision-taking process. The author introduced some interesting definitions for events which might serve as an inspiration for a classification of the possible disturbances:

- Negative events: events that cause negative deviations and require actions to be taken
- Positive events: events that cause positive deviations and allow to have more available time.
- Late events: events which happen after the moment when they were expected to occur (e.g. late start of a production).
- Early events: events which happen before the moment when they were expected to occur.
- Unexpected events: events that happen as the result of unplanned situations and for which there are no foreseeable countermeasures (e.g. a traffic jam, a machine breakdown)
- Unreported events: events which were expected to occur but in reality they did not (e.g. missed confirmation of the handing over of goods by the transporter)

Finally, a very basic list of examples of disturbances that might affect the performances of a supply chain was reported: « late delivery, breakdown of IT-systems or production machines, variations in demand, supply, transportation ».

Borgman et al. (2010) also underlined the effect of information, focusing their work on an import container terminal where the departure time of the containers are subject to high uncertainty, resulting in missing, imperfect or corrupted information. To take this issue into account in their model, two classes of departure times of the containers were created: the planned/expected departure time and the actual/real departure time. The variation between the two types of departure times was generated using a specific perturbation function which depends on the mode of later transport (train, truck, another vessel etc.)

Zhen (2014) proposed a decision support system that deals with transhipment terminals working under uncertain conditions. He stated that many of the most advanced allocation system do not take uncertainty and unexpected events into consideration. Those events, as a matter of fact, might seriously affect efficiency and performances in the yard, to the point of rendering pre-planned strategies entirely unfeasible. As an example of those uncertain conditions the author cited unexpected changes in the loading or unloading time of the vessels and variations in the workloads across different time shifts. He went even more into detail by distinguishing uncertainties by the impact they have, respectively, on:

- Unloading plan of the incoming vessels. It comprises all the events which might affect the voyages and the arrival times of the arriving vessels. As an example of those kind of events, the author mentioned «unforeseen changes» in weather conditions, sea routes or unpredicted engine problems. The results of these changes are vessels arriving in the port outside their pre-determined time window.
- Loading plan of the outgoing vessels. In the same way as for the unloading plan, the same kind of events that can interfere with the programmed voyage of the ships are going to affect the loading plan since the considered system is a transhipment hub.

Petering (2015) defined very briefly two types of disturbances while explaining the reason for an uneven distribution over time of the workload of the cranes: bad sea conditions and delays at previous ports which can result in a later vessel arrival. As anticipated in Section 2.2.1, the author also defined a set of events which were implemented in the simulation model. However, for the purposes of the thesis, they could be neglected since they are not proper disturbances or external events that may alter the usual operations in the yard but can be interpreted as features of the model instead. Just to give a quick example, some of those events are: arrival of a vessel, ending of the berthing of the vessel, ending of the cross gantry by a yard crane, ending of the handling operations by a yard crane etc.

Rekik et al. (2016) proposed and early version of the Case-Based heuristic they developed in their 2018 work. They attempted for the first time a classification of events and disturbances that might happen in a container terminal while adding some example to explain the meaning of each class:

- Resource related disturbances. E.g. the breakdown of a yard crane

- Equipment related disturbances. E.g. the breakdown of a block
- Container related disturbances. E.g. the breakdown of a container, a misplaced container, the change in a container time of delivery

Later on, the authors added the arrival of dangerous containers with flammable or toxic content as another example of disturbance. Moreover, they stated that many of the existing decision support systems for stacking allocation only work with a pre-determined set of rules that remains the same throughout time, without adapting to the changes of the real environment. In addition to that, those works which try to deal with disturbances and external events have never focused on the interaction between the containers and all the type of disturbances that might happen.

Rekik et al. (2018) proposed again the same classification of disturbances and reiterated their position on the absence of an in-depth study of the effect of disturbances in a container terminal and on the fact that disturbance management is still an unsolved problem.

Gharehgozli and Zaerpour (2018) proposed a shared stacking policy for a transhipment terminal, focusing on outbound container which arrive in the port travelling on barges and are scheduled to be loaded on deep-sea vessels. They introduced the causes for a late barge arrival: congestion, bad weather conditions or accidents.

CHAPTER 3

Decision Criteria Classification in Container Space Allocation

This chapter represents another output of the literary review. Here is presented a rational classification of the criteria used for stacking containers in real time, of the KPIs used to assess the performances of a certain decision rule and of the events and disturbances that might happen in the yard. To the best of the author's knowledge, this is the first classification of this kind in the area of container space allocation. The overview aims to establish relevant links between the classifications to provide recommendations for strategic comparative studies.

3.1 Introduction and rationale

The idea for a more structured version of the literature review came about considering the scope of the thesis. Since the aim is to define a decision support system that reacts to uncertain events in the yard, a strong focus is put on finding similarities between the different elements which are involved in stacking a container in the yard. In particular, the stacking policies were deconstructed into the single criteria that compose them. Then, the criteria were examined in depth in order to understand which ones used the same attribute or characteristic (of the yard, of the container etc.), grouping them together. Those subgroups were again put together in classes according to the similarities between them. The same approach was adapted to events, which, again, were split into classes. With the purpose of creating a reactive stacking system, it became extremely interesting to examine the link between the groups or classes of the events and the classes of the criteria that make the stacking policies, in order to see whether those events might have an impact on certain policies and if that link has ever been taken into consideration in the existing literature. This process was then applied to the Performance Indicators, in order to understand the effect of the events on how performances are assessed. Moreover, the new-born classes of Performance Indicators were compared with the Criteria classes, hence highlighting the close relationship between the two: if a feature is at the basis of both a criteria and a Performance Indicator, adding that criteria to the stacking policy affects (usually in a positive manner) the Performance Indicator. In order to keep all of these considerations together, a complete matching of criteria, Performance Indicators and events was attempted with different methods. This also laid the basis for the idea that is behind a dynamic stacking policy, which will be explained later on.

3.2 Criteria Classification

Before describing how literature was classified, a brief nomenclature introduction might be needed in order to fully comprehend how the work was carried out. A distinction needs to be made between Decision Rule and Criteria.

- Decision Rule: a combination of criteria which constitutes a stacking/allocation policy. The output of a Decision Rule is the final stacking position in the yard: it allows to find where to stack an incoming container. It could be considered a synonym of stacking policy.
- Criteria: principles based upon whi1ch the stacking of an incoming container happens. They represent the use of a certain attribute or characteristic belonging to the container, to the yard, to the resource etc.) in order to define the stacking slot. However, this does not mean that a single criterion univocally determines the final stacking position (e.g. within a Decision Rule, one criterion is responsible for block allocation and another one for stack allocation).

The scientific papers presented in the previous 2.2 subsection were reviewed in a successive way. A list of criteria was created and each time a new different criterion appeared it was added to the list. Then, the criteria were examined in order to find commonalities between them. The results of the review are presented in the form of a table: Table 3.1. In the table, each column represents a criterion (aside for the last three). Criteria were grouped in three main classes which are related to the environment of the port and other two classes which account for the presence of a certain degree of randomness and the various forms of pre-planning. Classes and their respective criteria are presented below:

- CLASS: Container-related. It comprises all the criteria that are built using attributes that belong to the incoming container.
  - CRITERION: Time. It represents the use of the departure time of the containers to define the final stacking position (e.g. containers with an earlier estimated time of departure should be stacked on top of containers with a later departure time). It comprises the case in which departure times are segregated into classes/categories (see Petering, 2015 and Petering et al., 2017).

- CRITERION: Weight. It represents the use of the weight of the containers to define the final stacking position (e.g. lighter containers on top of heavier ones for the sake of the stability of the stack or heavier containers on top of lighter ones in order to respect the loading sequence of the departing vessel in a transhipment terminal). It also includes allocation based on weight related categories, a consequence of segregation which represents the majority of the cases. See Kim et al. (2000), Saanen and Dekker (2006), Saanen et al. (2007), Park (2011), Petering (2015) and Guerra-Olivares et al. (2017).
- CRITERION: Type. It represents the use of the type of the incoming container to define the final stacking position. Within the considered papers it coincides with the practice of segregation according to the type. By container type, it is meant the trade type of the container (import, export or transhipment), whether it is full or empty, the specific constructive typology of the container (dry van, reefer etc.) or any attribute related to its content (dangerous or toxic content etc.)
- CRITERION: Destination. It represents the use of the destination of the incoming container to define the final stacking position. It includes the practice of segregation of the containers according to their destination. The proposed definition of destination is the following: the port of destination for export or transhipment containers or the land destination plus the relative mean of transport (truck, rail etc.) for import containers (for example, see Guven and Eliiyi, 2018)
- CRITERION: Ship. It represents the use of the ship where the incoming container is scheduled to be loaded after its dwell time has expired (e.g. stacking the incoming container only in stacks where the residing containers are bound for the same ship). It comprises the practice of segregating the incoming containers in groups according to their future vessel. It is a practice only valid for export and transhipment containers.
- CRITERION: Size. It represents the use of the size of the incoming container to define the final stacking position. With size it is generally meant the standardised length of the incoming container which can be either 20’ (or a TEU, Twenty-feet Equivalent Unit) or 40’ (FEU, Forty-feet Equivalent Unit). In some cases, it may also mean the height of the container, as it happens for

Petering (2015) and Petering et al. (2017), where height is taken into consideration in the segregation process of the container.

- CLASS: Yard-related. It comprises all the criteria the are built using attributes that belong to the yard and, more generally, to the layout of the stacking area. Therefore, in this category are included all the distances from the elements of the yard to the other areas of the port (gate and quay) as well as the characteristics of the blocks and stacks.
    - CRITERION: Distance-related. It represents the use of distance to define the final stacking position of the incoming container. This criterion comprehends the use of all the possible distances in the yard: block to gate, quay to gate, stack to the transfer point at the end of the lane etc.
    - CRITERION: Height. It represents the use of the height of the stacks to define the final stacking position of the incoming container. In order to be considered in this category, a criterion based on stack height must use it in an active way, which means using the height to define a preferable stacking slot (e.g. see Ries et al. (2014) where the stack height is associated with a Stack Value of Goodness) instead of simple passive constraints (e.g. a stack should not be higher than four tiers).
    - CRITERION: Utilisation. It represents the use of the share of capacity of one of the elements of the yard: in particular, for the examined papers, the block or the stack. Using the share of capacity of the block has been implemented in Ries et al. (2014). Moreover, a criterion which assigns a preference for empty stacks (see Borgman et al. (2010) and Petering (2015)) is considered part of the Utilisation sub-group since an empty stack can seen as occupied at 0% of its capacity. This is done also in order to separate criteria which give preference to the lowest stacks from criteria where the preference is given to an empty stack specifically.
- CLASS: Resource-related. It comprises all the criteria that are built using attributes related to the resources that serve the yard. Resources are intended as the vehicles or the infrastructures working in the yard such as AGVs, Yard Trucks, external trucks, Yard Cranes, Automatic Stacking Cranes etc.
    - CRITERION: Workload. It represents the use of the workload of the resources to the define the final stacking position of the incoming container Workload is defined as the amount of work that a given resource is able to

perform. It is important to note that workload can be assessed in various ways different ways while referring to different vehicles: Saanen and Dekker (2006) considered the number of orders and their duration for the RTGs, Dekker et a. (2007) defined the workload for ASCs as the percentage of time in the current quarter in which they are busy.

  - CRITERION: Position. The use of the position of the resources to define the final stacking position of the incoming container. It is used only by Saanen and Dekker (2006), where, among other criteria, a preference is given to the closest stack to the current position of the RTG.
  - CRITERION: Congestion. The use of the congestion level of the resources to define the final stacking position of the incoming container. This subcategory has been added to take into consideration a specific criterion applied by Petering (2015) and Petering et al. (2017): a part of the weighted sum that defines the opportunity of stacking the incoming container in a given block is represented by the number of containers currently heading to a block. This is considered a measure of congestion, being different from workload: congestion can be seen as a queue or a bottleneck of one of the resources which happens at a specific point in time and is measured in real time, instant by instant, while workload is a quantity that is measured over a longer time interval.
- CLASS: Random. It is not properly a class of attributes but it accounts for the presence of certain elements of randomness in the Decision Rule.
- CLASS: Pre-Planning. Again, it is not properly a class of attributes but it accounts for the presence of pre-planning, where the allocation of the incoming container do not happen entirely in real time but is based on a pre-determined path which reserves specific areas to certain groups of incoming containers.

Meanwhile, each line of the table corresponds to a Decision Rule. Each Decision Rule has been assigned a code. As an example:

Ries 14 – 13000

The name, in this case "Ries", is the name of the first author of the paper where the Decision Rule was presented for the first time. The following number, "14", is an indication of the year when the paper was published, 2014. Finally, the final 5 numbers are a brief representation of the criteria used in the decision rule: the first number corresponds to how many container-related criteria were used, the second to how many yard-related criteria were used, the third to how many resource-related criteria were used, the fourth is 1 whether some elements of randomness were used, in the same way as the fifth which is 1 when a form of pre-planning was used. Hence, for the example the code can be read in the following way: in the paper publish by Ries et. al in 2014, one container-related criteria and three yard-related criteria were used, while resource-related criteria, randomness and pre-planning were not employed. When a Decision Rule adopts a certain Criteria, this is marked in the table by a cross: ✗. A specific column of the table also reports the name of the proposed Decision Rule, as stated in the relative paper. When no name is proposed, a blank space is left.

3.2.1 Discussion on the findings

Before addressing the considerations that can be advanced after looking at the proposed table, a little explanation is needed for the decision rule Guerra-Olivares 17 – 10001. In the relative live there are four crosses in brackets. They represent the criteria that were used for the pre-planning phase hence they are not considered for the evaluation of the real-time model which only works with weight. This could be the same thing for Petering 15/17 – 52211 and Kim 00 – 10001 but those models use in real-time all the criteria that were used for the forms of pre-planning they employ so there is no need to indicate crosses in brackets.

In the table there are 22 Decision Rules coming from 10 different scientific studies. It was not possible to add the work by Rekik et al. (2018) since the authors did not express in an explicit way the type of Decision Rules they implemented in their Case-based heuristic. A simple frequency count shows that the two most used criteria are Time, which is container-related, and Distance, which is yard-related. The most used resource-related criterion is Workload while Congestion and Distance are used only once. The Decision Rule that employs the highest number of criteria is Petering 15/17 – 52211. Only four Decision Rules apply criteria that belong to the three main classes, container-related, yard-related and resource-related: Saanen 06 – 41200, Dekker 07 – 61110, Park 11-42100 and Petering 15/17 – 52211. It is also interesting to note that all the Decision Rules, except for four of them, always adopt at least one container-related criterion and that no Decision Rules is based only on a resource-related criterion. The most complete of the Fuzzy Systems that were developed for the purpose of the thesis is going to follow the trail set by the group of four aforementioned Decision Rules which use at least one criterion belonging to each one of the main criterion-classes. Moreover, the dynamic features that will be explained in the next chapters are common only to one of the 22 Decision Rules, Park 11 – 42100.

Decision Rule Code	Paper	Year	Model Name	Container related						Yard related			Resource related			Random	Pre-planning	Number of Criteria in a Decision Rule
				Time	Weight	Type	Destination	Ship	Size	Distance	Height	Utilization	Workload	Position	Congestion	-	-	
Kim 00-10001	*"Deriving decision rules to locate export containers in container yards"* Kim, Park, Ryu	2000	Dynamic Programming Model		✗												✗	2
Saanen 06-41200	*"Intelligent stacking as a way out of congested yards?" Pt. 1 and Pt. 2,* Saanen, Dekker	2006		✗	✗		✗	✗		✗			✗	✗				7
Dekker 07-20010	*"Advanced methods for container stacking"* Dekker, Voogd, Van Asperen	2007	Random stacking	✗					✗							✗		3
Dekker 07-61110			Category stacking	✗	✗	✗	✗	✗	✗	✗			✗			✗		9
Borgman 10-00010	*"Online rules for container stacking"* Borgman, van Asperen, Dekker	2010	Random stacking													✗		1
Borgman 10-01000			Levelling								✗							1
Borgman 10-13000			LDT	✗						✗	✗	✗						4
Borgman 10-10010			RSDT	✗												✗		2
Borgman 10-13000b			LDT-DTC	✗						✗	✗	✗						4
Borgman 10-10010			RS-DTC	✗												✗		2
Borgman 10-02000			TVR							✗	✗							2
Borgman 10-01010			TPRL							✗						✗		2
Borgman 10-12000			TVR-PA			✗				✗	✗							3
Borgman 10-13000c			TVR-DTC	✗						✗	✗	✗						4
Borgman 10-13000d			TVR-DTC-MD	✗						✗	✗	✗						4
Park 11-42100	*"Dynamic adjustment of container stacking policy in an automated container terminal"* Park, Choe, Kim, Ryu	2011		✗	✗		✗		✗	✗	✗		✗					7

Decision Rule Code	**Paper**	**Year**	**Model Name**	**Container related**						**Yard related**			**Resource related**			**Random**	**Pre-planning**	**Number of Criteria in a Decision Rule**
				Time	**Weight**	**Type**	**Destination**	**Ship**	**Size**	**Distance**	**Height**	**Utilization**	**Workload**	**Position**	**Congesti on**	-	-	
Ries 14-13000	*"A Fuzzy Logic Model for the Container Stacking Problem at Container Terminals"* Ries, Gonzàlez Ramirez, Miranda	2014	Fuzzy Logic Framewo rk	✗						✗	✗	✗						4
Petering 15/17-52211	*"Real-time container storage location assignment at an RTG-based seaport container transshipment terminal: problem description, control system, simulation model, and penalty scheme experimentation"* Petering	2015		✗	✗		✗	✗	✗	✗		✗	✗		✗	✗	✗	11
	"Real-time container storage location assignment at a seaport container transshipment terminal: dispersion levels, yard templates, and sensitivity analyses" Petering, Wu, Li, Goh, de Souza, Murty	2017																
Guerra Olivares 17-10001	*"A heuristic procedure for the outbound container space assignment problem for small and midsize maritime terminal"* Guerra Olivares, Gonzàlez Ramirez, Garcìa Mendoza, Cardenas Barròn	2017			✗	(✗)	(✗)	(✗)	(✗)								✗	2
Guven 18-60010	*"Modelling and optimisation of online container stacking with operational constraints"* Guven, Eliiyi	2018	Random	✗	✗	✗	✗	✗	✗							✗		7
Guven 18-60000			ABS	✗	✗	✗	✗	✗	✗									6
Guven 18-50000			3 tons Relaxatio n	✗		✗	✗	✗	✗									5
The Proposed Model				✗						✗	✗	✗			✗			
Overall Frequency of Criteria (Out of 22 Decision Rules)				15	8	5	7	6	7	12	9	6	4	1	1	8	2	

Table 3.1 Decision Rules Framework table

3.3 Performance Indicators Classification

For a review of the Performance Indicators, the same approach used for the Decision Rules was adopted, scrutinising the literature first in a successive order and then analysing the existing similarities between the newly found Key Performance Indicators. Again, the results of the review are presented with the aid of a table: Table 3.2. Columns show the groups of Performance Indicators (PIs) that are based on the same attributes. Those groups are then put together in classes which match with the three main classes proposed for the Criteria Classification. Groups and classes are presented below:

- CLASS: Container-related. It comprises all the Performance Indicators that are built using one of the attributes that belong to the containers or a metric which is closely linked to container-related attributes.
    - PI group: Rehandles/Reshuffles. It represents the use of Rehandles to evaluate the performance of a stacking strategy. Rehandles are defined as the unproductive moves needed to retrieve a container from a stack. They are considered to be part of the container-related class since they are generated by properties of the containers (weight, dwell time etc.) that determine their stacking order.
- CLASS: Yard-related. It comprises all the Performance Indicators that are built using of the attributes that belong to the yard, where yard assumes the same meaning as for yard-related criteria.
    - PI group: Distance. It represents the use of one or more distances between elements in the yard to evaluate the performances of a stacking strategy. The distances considered in the analysed papers are block-gate distance (twice), stack-gate distance and stack-transfer point distance.
    - PI group: Space Utilisation. It represents the use of the share of utilised capacity of the elements of the yard as well as more general considerations on how the space in the yard is employed. In the reviewed literature, the focus is on the remaining stack height and the occupation of ground locations.
- CLASS: Resource-related. It comprises all the Performance Indicators that are built using attributes that belong to the resources (vehicles and infrastructures) that serve the yard.
    - PI group: Workload. It represents the use of the workload of the resources to assess performances. As mentioned in the previous section for criteria

classification, workload is defined as a measurement of the amount of work performed by a resource.

- PI group: Congestion. It represents the use of congestion in the yard to measure operations in the yard. As mentioned in the previous subsection, congestion is defined as a real-time metric of queues and bottlenecks in the port.

Another additional column is added to show the difference between a Performance Metric and an Objective Function. In two of the reviewed papers the stacking strategy is represented by a mathematical model: Kim et al. (2000) and Guven and Eliiyi (2018). In the former, a dynamic programming model is defined offline with the Number of Relocations as the objective function. Then, as the computational time of the model is too long, a decision tree based on the former is developed to be used in real-time. Since the authors did not provide any sort of evaluation of the model or the decision tree, but just a comparison of the two, the Number of Relocations was added in the table as the Performance Indicator, since it is the only way in which performance is assessed in the paper, adding their status as objective function in the respective column. In the latter, a mathematical model is run dynamically, each time a container arrives in the yard, being formalised through a recursive algorithm. The additional column allows to differentiate the objective function of the model (the number of sub-optimal allocations which corresponds to the number of reshuffle occasions) from the four performance metrics (number of reshuffle occasions, number of reshuffles, travelled distance by empty ASCs and time lost due to empty travels and reshuffles) which are used ex-post to evaluate the goodness of the algorithm. With the last example, the difference between objective function and performance metric might appear slightly clearer: a same Performance Indicator might be used as an objective function or a performance metric, where in the former case it contributes actively to the determination of the optimal stacking sequence (all the choices for block and stack assignment are made in order to maximise or minimise said Performance Indicator), while in the latter it simply acts as a passive way to measure the validity of a certain allocation strategy, with no interference in the stacking process.

On the other hand, each line represents a single Performance Indicator. If more PIs are introduced in the same paper, they are grouped together. At the intersection between lines and columns there might be a cross (✗), signalling that the considered PI belongs to the group of the column.

3.3.1 Discussion on the findings

The table shows 24 different Performance Indicators. Reshuffles and the different forms in which they are assessed represent the most frequent group of Performance Indicators, according to a very quick examination. This evidence highlights the importance of Reshuffles as an indicator of the performances in the yard: reducing the number of rehandles, in fact, allows to reduce the vessel turnaround times for export containers as well as the external trucks turnaround times for import containers, thus increasing the container throughput, which represents the principal source of revenue for the company that manages the port. It is intriguing to note that Congestion shows the same frequency as Workload and Distance but, while Workload and Distance are present in four different papers each, Congestion is used to define a KPI only two times, in particular by Park et al. (2011) and Rekik et al. (2018). This might be an indication that Congestion is not considered as often as the others as a viable mean to construct a Performance Indicator. Another interesting observation that can be made regards the fact that the vast majority of Performance Indicators only focus on one aspect, one attribute, with the result of having many papers in which the allocation strategy is evaluated from different perspectives, using different KPIs, one by one. Only in three cases it is possible to observe a "multiple" KPI, a Performance Indicator that is able to consider more attributes at the same time:

- The Exit Time, proposed by Borgman et al. (2010), defined as the time it takes to retrieve a container from a stack and have it ready for the successive transport. It can be considered a "multiple" KPI because it is the result of three other metrics that belong to three different PI groups: the distance from the stacking position to the transfer point (Distance), the eventual number of reshuffles (Reshuffles) and the workload of the ASCs (Workload).
- Time lost due to reshuffles and empty ASC travel, proposed by Guven and Eliiyi (2018). The final time is a function of the amount of rehandles (which is metric that belongs to the Reshuffles group) and the distance travelled by ASCs when they are empty (so a metric that belongs to the group Distance).
- The weighted sum introduced by Rekik et al. (2018), which depends on block-gate and stack-gate distance (Distance), the queue of containers in front of a block (Congestion) and the remaining stack height (Space Utilisation).

Only one of those three KPIs, Exit Time, uses attributes belonging to each one of the three main classes and only the first two consider the most commonly used Performance Indicator, Rehandles. According to the proposed classification, it is possible to find a gap in the existing literature: there are not enough elaborate KPIs which are able to capture the situation in a port with a single index, employing different attributes belonging to different main classes at the same time. This allows to evaluate the overall performance of a stacking strategy without focusing only on one KPI at a time, which might cause difficulties in determining the eventual trade-offs. In this thesis, in order to overcome this issue, a new way to measure performances has been developed: two approaches, Utopia Point and Ranking, which consider multiple attributes to form a KPI which is able to assess and measure different metrics at the same time. They will be explained in detail in the next chapters.

Paper	Year	Performance Indicator	PM/ OF	Brief description	Container related	Yard related		Resource related		Number of attributes for every Performance Indicator
					Relocations/ Reshuffles	Distance	Space Utilisati on	Workload	Congestion	
"Deriving decision rules to locate export containers in container yards" Kim, Park, Ryu	2000	Number of relocations	OF		✗					1
"Intelligent stacking as a way out of congested yards?" Pt. 1 and Pt. 2, Saanen, Dekker	2006	Quay Crane Productivity	PM	Moves per hour performed by a Quay Crane				✗		1
"Advanced methods for container stacking" Dekker, Voogd, Van Asperen	2007	Reshuffles	PM	Calculated as a percentage over the number of containers that leave the stack	✗					1
		Number of reshuffle occasion	PM	A reshuffle occasion is defined as one or more reshuffle operations needed to retrieve a container	✗					1
		Workload of ASCs	PM	The share of the total available time when a resource (ASC) is busy				✗		1
		Occupation	PM	Share of occupied ground locations			✗			1
"Online rules for container stacking" Borgman, van Asperen, Dekker	2010	Exit Time	PM	The time in hours that it takes to remove a container from the stack and have it ready for transport	✗	✗		✗		3
		ASC Workload	PM	The percentage of time that the ASC's are busy				✗		1
		Reshuffles	PM		✗					1
		Ground Position Usage	PM	The average percentage of ground positions that are in use			✗			1
"Dynamic adjustment of container stacking policy in an automated container terminal" Park, Choe, Kim, Ryu	2011	Quay Crane delay time	PM						✗	1
		AGV waiting time	PM						✗	1
		External truck waiting time	PM						✗	1
		Evaluation value	PM	A weighted sum of the three previous Performance Indicators					✗	1
"*A Fuzzy Logic Model for the Container Stacking Problem at Container Terminals*" Ries, Gonzàlez Ramirez, Miranda	2014	Ratio of relocation moves	PM		✗					1
		Distance travelled by containers	PM			✗				1

Paper	Year	Performance Indicator	PM/ OF	Brief description	Container related	Yard related		Resource related		Number of attributes for every Performance Indicator
					Relocations/ Reshuffles	Distance	Space Utilisation	Workload	Congestion	
"Real-time container storage location assignment at an RTG-based seaport container transshipment terminal: problem description, control system, simulation model, and penalty scheme experimentation" Petering	2015	GCR	PM	Gross Crane Rate: the average number of lifts achieved at a terminal per QC working hour.				✗		1
"Real-time container storage location assignment at a seaport container transshipment terminal: dispersion levels, yard templates, and sensitivity analyses" Petering, Wu, Li, Goh, de Souza, Murty	2017									
"A heuristic procedure for the outbound container space assignment problem for small and midsize maritime terminal" Guerra Olivares, Gonzàlez Ramirez, Garcìa Mendoza, Cardenas Barròn	2017	Number of rehandles	PM		✗					1
"Modelling and optimisation of online container stacking with operational constraints" Guven, Eliiyi	2018	Number of sub-optimal assignments (Number of reshuffles)	OF	The objective function minimises reshuffle occasions by minimising the chance of assigning incoming container c to a sub-optimal position (which respects only size and weight constraints)	✗					1
		Number of reshuffles	PM		✗					1
		Number of reshuffle occasions	PM		✗					1
		Travelled Distance by an ASC	PM	It is considered an unproductive feature		✗				1
		Time lost due to reshuffles and empty ASC travels	PM		✗	✗				2
"A case-based heuristic for container stacking in seaport terminals" Rekik, Elkosantini, Chabchoub	2018	Weighted Sum	PM	The objective function is a weighted sum of four components: the distance separating the considered block and the gate, the waiting queue in front of the given block, the distance separating the given stack and the gate, the remaining stack height		✗	✗		✗	3
The Proposed Ranking and Utopia Point Approach					✗	✗	✗		✗	
Overall Frequency of Target attribute (Out of 24)					11	5	3	5	5	

Table 3.2 Performance Indicators Framework table

3.4 Events Classification

In the same way as for Criteria and Performance Indicators, a progressive review of the existing literature has been done, trying to find a codification of the events and disturbances that might happen in the yard, with the underlying target of defining a classification that is similar to the one proposed for Criteria and PIs. Paper were examined one by one and the results were collected in a visual fashion shown in Figure 3.1.

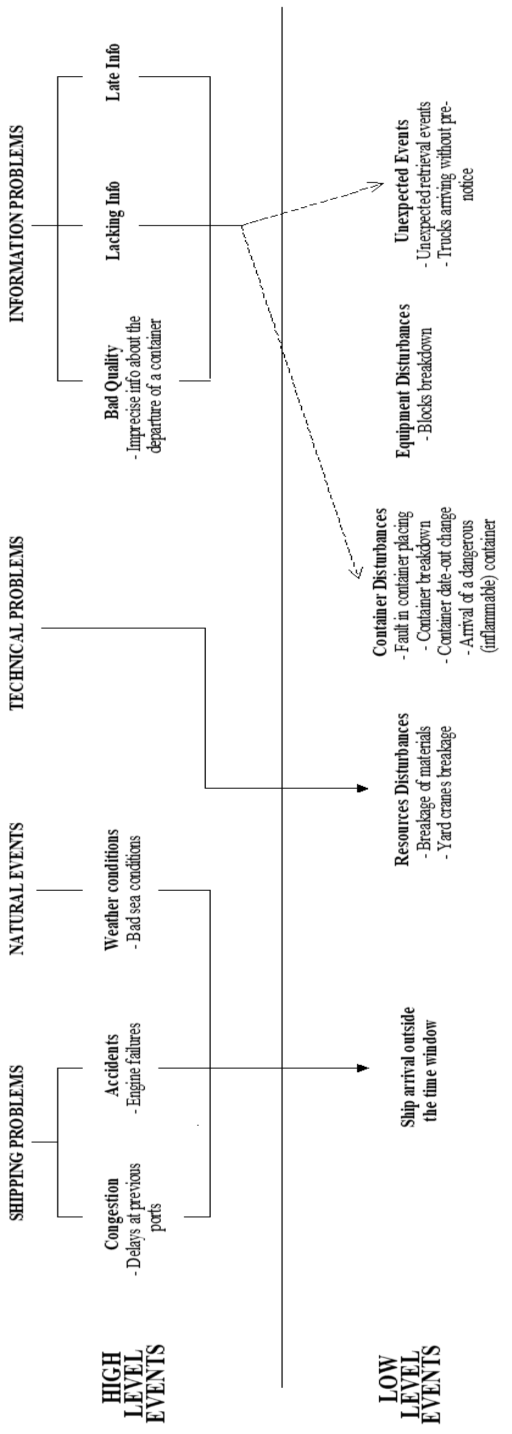


Figure 3.1 Events and Disturbances Classification Template

A first main distinction can be made between High Level and Low Level events:

- High Level: it comprises events that might happen inside or outside the container terminal and are not necessarily related to it. Hence, they may or may not have consequence for the port but when they do, they are the origin of the Low Level events, the reason which causes Low Level events
- Low Level events: it comprises all the events that happen inside the yard and effectively represent a form of disturbance to the usual operations of the yard. In general terms, they can be interpreted as the consequence on the yard of one of the High Level events.

Then, it is possible to investigate much further within those two groups. The disturbances are now presented in an indented fashion in order to show the relationship between them. Whether an event belongs to the High or Low Level is represented by the letters HL or LL respectively:

1. Information Problems (HL). It comprehends all the disturbances that might affect the flow of information to and from the port. It includes information on the incoming containers as well as the vessels or the external trucks. The importance of information and the effects of its changes were first formalised in a paper by Saanen and Dekker (2006). The types of events that interfere with information that were found in the literature are:
   - Bad Quality (HL). It is the case in which information is corrupted, wrong or inaccurate. An example of this is represented by:
     - Imprecise information about the departure of a container (HL). Defined by Borgman et al. (2010), it refers to an environment where there is high uncertainty regarding the delivery times of the containers.
   - Lacking Information (HL). In this case information is completely missing and the port has to keep functioning without it.
   - Late Information (HL). It refers to the case in which information is available but only after the moment in which it was required. For example, in a yard where a stacking strategy that uses weight is in operation, the weight itself is not available at the moment when the stacking location is being decided.

The effects in the yard of those types of disturbances are described below. It is important to note that this cause-effect relationship is not derived from the literature but it is an hypothesis advanced for the first time in the current work. This is signalled in the template in Figure 3.1 by a dotted line.

- Container Disturbances (LL). It includes the effects of High Level events that might tamper with any operation dealing with a container in the yard environment. Examples of this type of disturbance are:
    - Fault in container placing (LL): a container is stacked in a wrong location. (e.g. in a yard where a segregation policy is in place, due to the lack of information a container is stacked randomly and ends up in a stack that belongs to a different group).
    - Container breakdown (LL): the definition of a container breakdown is not clear but it was explicitly cited as a form of Container disturbance by Rekik et al. (2018)
    - Container date-out change (LL): it represents a variation in the expected time of delivery of the containers that are stacked in the yard. It may be considered a rather common situation if between 30 and 40% of information changes during the dwell time of a container, as stated by Saanen and Dekker (2006).
    - Arrival of a dangerous container (LL): it is cited by Rekik et al. (2016) and Rekik et al. (2018) as a disturbance that can cause disruption in normal containers operations. It is safe to assume that a container which carries dangerous goods is subjected to different stacking and allocation procedures.
- Unexpected Events (LL): it comprises all the events in the yard that do not happen in the moment in which they are supposed to happen, due to lacking or imperfect information. Two examples are:
    - Unexpected retrieval events (LL). Cited by Rekik et al. (2018), it refers to any generic dispatching of containers, either by vessel (export) or by truck/train etc. (import)
    - Trucks arriving without pre-notice (LL). Also introduced by Rekik et al. (2018), it is a focus on the retrieval of import containers. The authors underlined that retrieval requests for containers in the yard are issued in a random order because the external trucks arrival follows again a random pattern.

2. Shipping Problems (HL). It is a comprehensive definition that includes all the issues that might affect the vessels (intended as vehicles) that serve the port. There are two main examples:
   - Congestion (HL). Not to be confused with the Congestion in the yard, it is a concept introduced by Gharehgozli and Zaerpour (2018) while referring to the reason for a late arrival of a barge in a transhipment terminal. It can be seen as the queues or delays that a ship might be facing during its voyage to the port. An example of this is:
     - Delays at previous ports (HL). The arriving vessel is delayed by events that happened in ports located before the considered one on its respective sea route.
   - Accidents (HL). It includes all the physical and mechanical issue that might affect a vessel, including:
     - Engine failures (HL). Cited by Zhen (2014).
3. Natural Events (HL). It represents all the natural occurrences which can happen in, around or even far away from the port and its yard but have a significant impact on the operations in the yard itself.
   - Weather Conditions (HL). Amongst the Natural Events that might affect the port, one of the most commonly cited (see Petering (2015) or Gharehgozli and Zaerpour (2018)) is the weather, because of obvious reasons. In more detail:
     - Bad-sea Conditions (HL). A bad sea is likely going to affect the travelling speed and the route of the vessels which are carrying containers to the port or which are bound to receive a new load.

The impact of Natural Events and Shipping Problems on the yard, as stated in the existing literature, is mainly represented by one aspect:

- Arrival of a vessel outside its time window (LL): contrary to external trucks, vessels and ships usually have a very well-defined time slot in which they are supposed to berth. Any disruption on their path to the port might cause a late arrival.

4. Technical Problems (HL). It includes all the issues that might affect the resources or the infrastructures of the yard. These Problems have a direct link and with Low Level Events, which interfere with the normal stacking operations. In literature, this was described by Rekik et al. (2016) and Rekik et al. (2018). The effects on the yard of those High Level events are:

- Resource Disturbances (LL): they are defined as an alteration of the normal functioning of the resources of the yard. Two examples found in literature are:
    - Breakage of Materials (LL): cited by Rekik et al. (2018), it refers to failures that may alter the normal functioning of the vehicle that work in the yard.
    - Yard Crane Breakage (LL): a failure of a Yard Crane, as stated again by Rekik et al. (2018).

Finally, the categorisation of disturbances is completed by a class of Low Level events which is not linked to any sort of High Level event, according to the literature:

- Equipment Disturbances (LL): following the same rationale used by Rekik et al. (2018), equipment may be seen as a synonym for yard, so this kind of disturbance is an event which is impacting one of the elements of the yard (block, stacks etc.) The relative example, in fact, is:
    - Blocks Breakdown (LL): not explained by the authors but it can be interpreted as a temporary exclusion of one or more blocks from the possible stacking destinations.

3.5 The Final Matching and Literature Gap

With the aim of bringing this work of categorisation together, the Events classification has been examined much further. In particular, the focus was on the Low Level Events. Each Low Level event has been grouped according to the answer to the following question: which element of the yard is most affected by this event? This has led to splitting the Low Level Events into three main classes that are inspired by the ones mentioned by Rekik et al. (2018) and, incidentally, are the same that were defined for Criteria and Performance Indicators:

- Container – related Disturbances: all the events and disturbances that affect mainly the container and some of its attributes, especially in relation to the stacking process. Container Disturbances, as mentioned by Rekik et al. (2018), obviously belong to this class. It is worth mentioning that also Unexpected Events and Ship Arrivals Outside their Time Window are also considered as Container-related events. Unexpected Events are related to time and situations that occur not in the correct time slot. Time is also one of the criteria that is used to allocate incoming containers. Therefore, having an event which is likely to change the dwell times of the containers, or more simply, their delivery times, is affecting directly how the they are being allocated. At the same time, Ship Arrivals Outside the Time Window have the same impact on containers if their flow is land-sea or in the case of a transhipment terminal.

- Yard – related Disturbances: there is only one example of this kind of event in the literature and it is something which is related to the yard layout: a block breakdown. The exclusion of one or more blocks from the possible stacking destinations of an incoming container can be seen as a direct modification of the yard layout with an implication on the distances: for example, in a terminal where a stacking strategy that looks for the shortest block-gate distance is in place, shutting down the closest block to the gate impacts the allocation of an incoming container, forcing it to another yard location.
- Resource – related Disturbances: it includes all the disturbances which directly affect the operations of the resources (vehicles and infrastructures) that are serving the yard. This class incorporates the event type of the same name proposed by Rekik et al. (2018). When a disturbance of this kind occurs, the direct impact is on the affected resource itself: a crane breakdown, for example, immediately influences its productivity and the workload it is able to absorb. Hence, the creation of a class that encompasses all the type of occurrences that might alter the normal functioning of said resources.

This classification shows that there is a pattern which links Criteria, Performance Indicators and Events: all of the three can be classified in classes called Container-related, Yard-related and Resource-related, which share the same definition. This is highlighted in colours in Figure 3.2. This link, however, is more robust than a simple name-sharing bond because it is based on a cause-and-effect relationship: each time an event of a certain class occurs, the relative Performance Indicator is affected. On the other hand, if the target is to improve a certain Performance Indicator, it is safe to assume that adopting the relative criterion, which belongs to the same class of the PI, results in an increase of the PI itself. For this reason, the Performance Indicators play a central role, connecting Criteria and Events, and this is represented by their position in the figure. In particular, when an event of a certain class is happening, affecting the PI of the same class, it appears reasonable to adopt a stacking strategy that employs criteria belonging to the same class of the event and the PI, allowing to respond directly to the change of performances. From these considerations arises the idea of a new real-time stacking policy that changes between criteria dynamically according to the event that is happening and, thanks to the link described above, while “rescuing” the affected Performance Indicator.

To the best of my knowledge, this approach to creating a stacking Decision Rule has never been featured in scientific literature. The only authors that defined a reactive allocation model, a properly structure stacking strategy that changes over time depending on the environment, were Rekik et al. (2018). Despite creating the event categorisation that inspired the one proposed in this work, they did not thoroughly examine the link that exists between Performance Indicators, Events and Allocation Criteria. As a matter of fact, they did not even specify the type of stacking strategies they implemented. Moreover, their model was based on a case-based heuristic. Instead, this work uses a well-defined group of stacking criteria and employs fuzzy logic to construct them. Furthermore, the selection of the Decision Rules to apply when a certain event happens is not based on previous events but on their categorisation, allowing to have a direct change of policy, more suitable for an environment that evolves in real-time. The is another example of dynamic allocation strategy, the one proposed by Park et al. (2011). This model, however, does not categorise events nor considers them as an active factor in the yard. In addition to that, the employed Decision Rule uses fixed Criteria, that do not change over time: the only dynamic aspect is represented by the weights of the sum that brings all the criteria together. All the other reviewed papers propose fixed Decision Rules and have never attempted or stated explicitly the link between criteria and Performance Indicators.

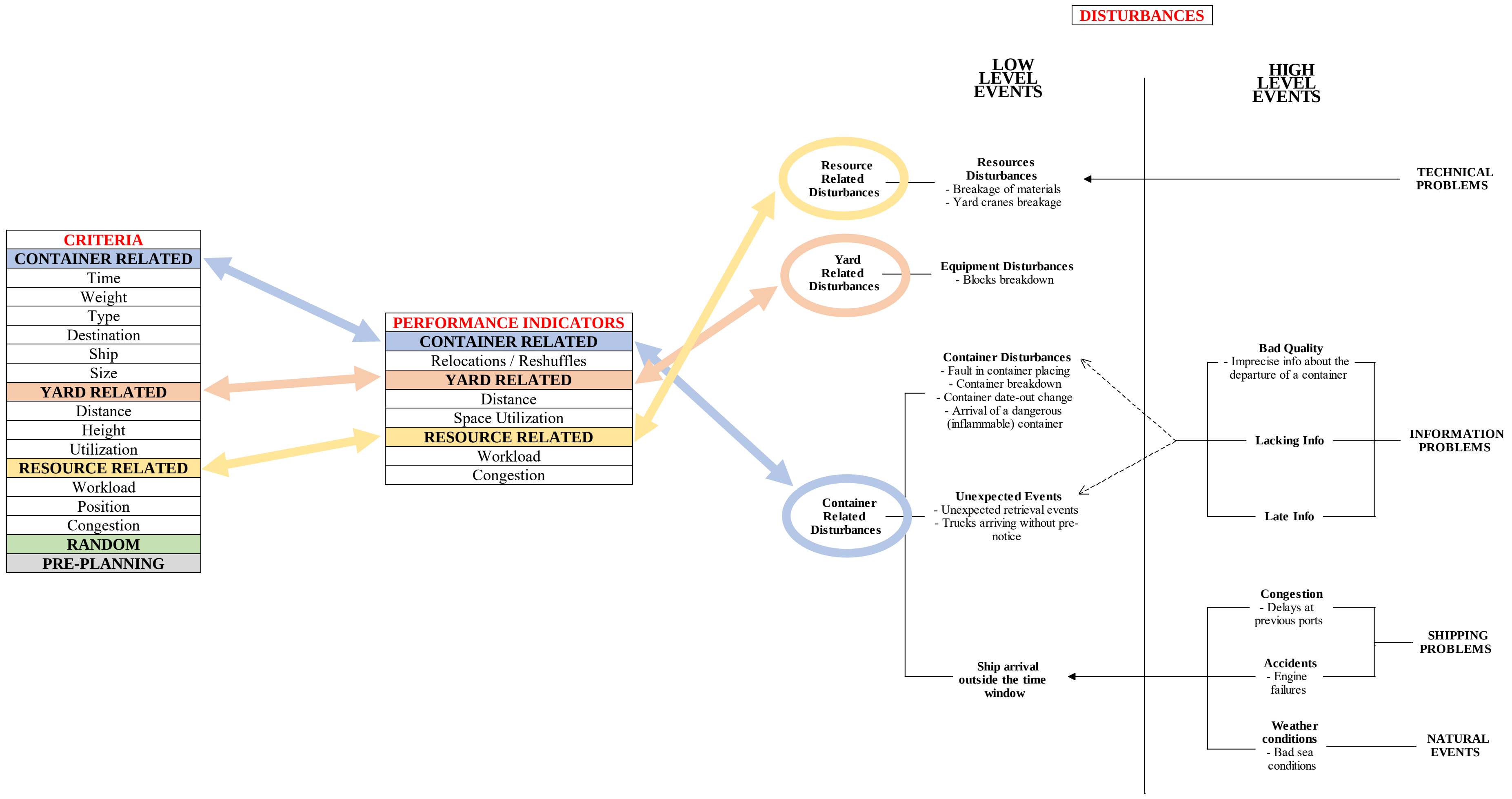


Figure 3.2 The link between Criteria, Performance Indicators and Disturbances

CHAPTER 4

The Case Study: The Port of Arica

This chapter includes a detailed description of the case study, the port of Arica, Chile. After a brief historical and geographical introduction, the chapter provides an insight into the layout and the main characteristics of the port.

4.1 A Historical Framing

Arica is a city located in the north of Chile, close to the border with Peru (see Figure 4.1). Up until 1880 Arica was a Peruvian town: it changed hands during the War of the Pacific (1879-1883), also known as the Saltpeter War. It was an armed confrontation between Chile and an alliance between Bolivia and Peru.

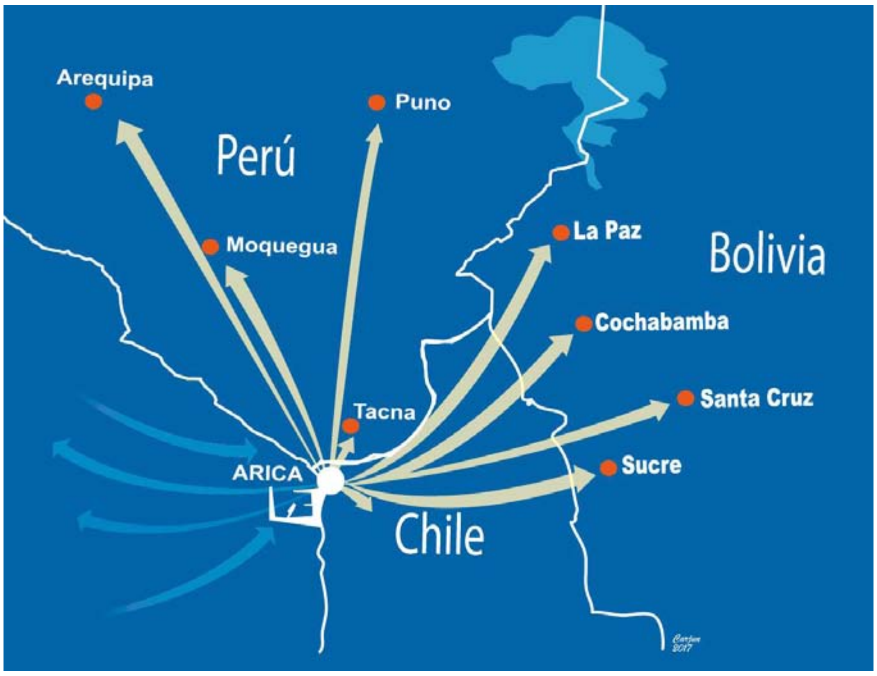


Figure 4.1 The geographical location of the port of Arica and the commercial links of the port (TPA website Arica map)

The result was a considerable territorial gain by Chile and the total loss to of an access to the sea for Bolivia. This was settled in 1904 with the Treaty of Lima, which granted full access to Chilean ports, Arica included, and no tariffs on Bolivian goods. This is the agreement that has been in place ever since.

4.2 The Port of Arica

Arica is the home of an important container terminal which is administered by a company called Terminal Puerto Arica (TPA) S.A. Employing 363 people, the terminal is, according to its 2018 Annual report, a commercial leader in the Macro Region Andina, a region that comprehends Northern Chile, Southern Peru, Bolivia and North-Western Argentina. During the year 2018, the terminal transferred 3.091.206 tons of goods, with a little decrease from 2017. With respect to the type of loads handled by the port, the 74% of them was represented by containers, 18% by bulk and 8% by break-bulk. Focusing on the containers, 2.296.427 of them were transferred through the port, which received the arrival of 204 Full Container Ships and 41 Multi-purpose vessels. An interesting consideration can be done regarding the breakdown of the transferred load in terms of origin or destination: 80% of it was bound to or came from Bolivia, 16% to or from Chile and 3% to or from Peru. This huge share of the load related to Bolivia is an important cause of uncertainty for the yard environment because it is a considerable quantity of goods that go through the port of a sovereign State but belong to a different Nation and it is supposed to be free from the normal constraints that are usually attached to cargo as a result of the aforementioned Treaty of Lima.

4.3 The Yard Layout

The layout of the TPA is shown in Figure 4.2.

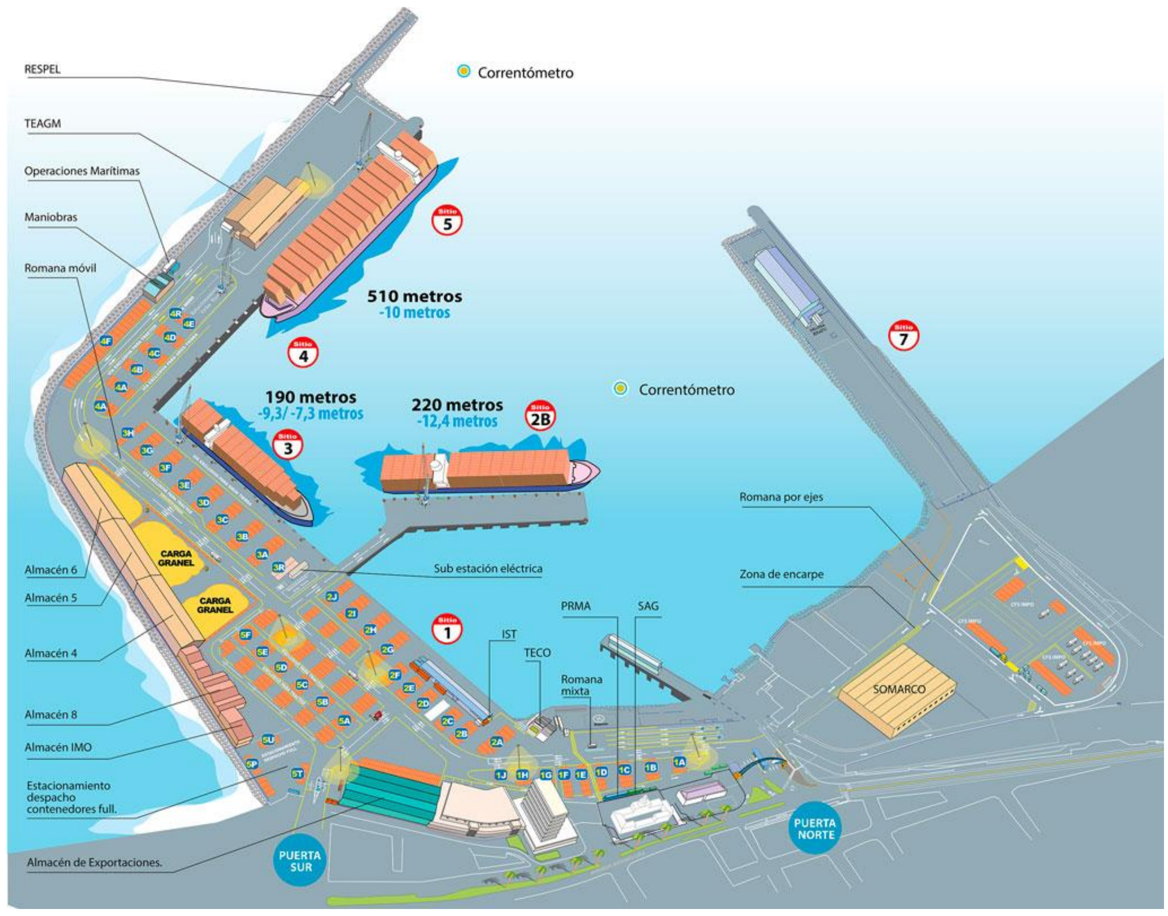


Figure 4.2 A visual representation of the container terminal in Arica (TPA website Yard Layout)

The yard of the port of Arica has 6 berthing sites for the arriving vessels. The containers that are unloaded or are scheduled to be loaded on the berthed vessels have 18 blocks to be stacked in. Those containers have entered the yard or will leave it passing through two gates, called respectively Gate 1 and Gate 2.

The Distance between the elements of the yard were calculated with the aid of satellite images and the relative instrument available on Google Earth. The map of the yard is represented in Figure 4.3.

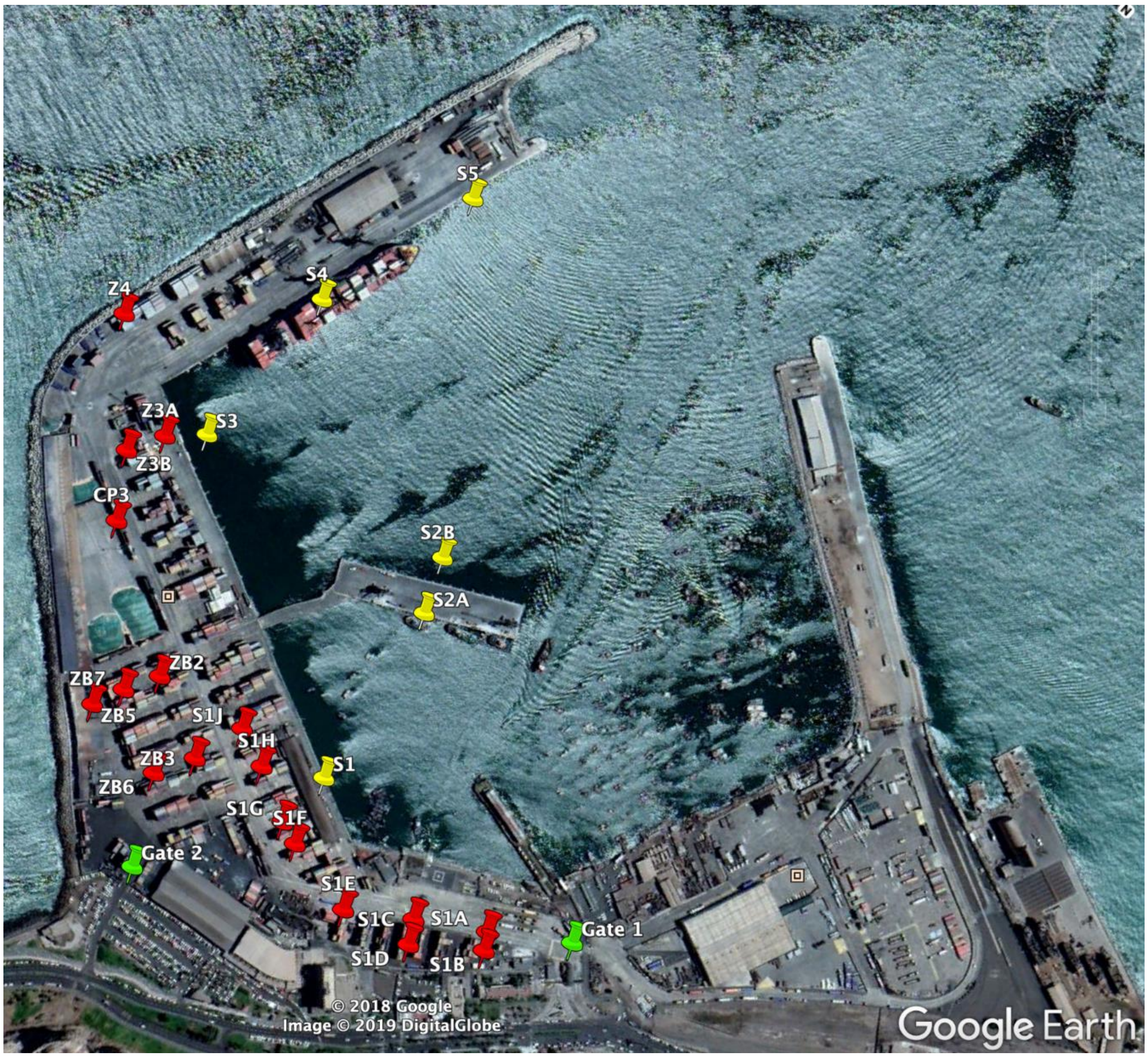


Figure 4.3 The Google Earth representation of the Port of Arica. The quays are pinpointed in yellow, the blocks in red and the gates in green

Distances were taken with the Manhattan approach which measures the space that separates two points following a path of straight lines and 90° turns: this way of assessing distance is considered the closest representation of the actual course followed by the vehicles in the yard. Quay-Block and Block-Gate distances, as a result of those measurements, are presented in Table 4.1 and 4.2 respectively. In the former, each block with its code is associated with its distance in metres from every quay (also called berthing site or more simply site), while in the latter the distance between each single block and every gate is reported.

	Quays/Berthing Sites					
Block	**S1**	**S2A**	**S2B**	**S3**	**S4**	**S5**
CP3	424.89	417.73	443.78	291.49	401.05	613.96
S1A	553.63	765.53	779.34	772.06	1043.64	1249.4
S1B	581.43	793.53	821.65	812.18	1097.28	1306.06
S1C	470.22	676.41	695.59	691.84	965.43	1169.92
S1D	513.73	741.01	749.78	734.77	1023.8	1222.28
S1E	406.57	633.21	654.46	642.57	928.1	1130.52
S1F	296.22	514.18	532.85	538.64	828.71	1015.49
S1G	276.12	493.28	508.9	498.88	774.66	987.51
S1H	197.13	434.71	440.74	430.52	714.97	924.94
S1J	142.95	386.86	406.24	396.62	668.16	876.57
Z4	558.71	555.1	573.31	158.06	259.27	470.36
Z3A	411.66	409.19	422.55	34.41	286.33	490.85
Z3B	474.87	483.47	499.81	219.2	336.87	538.3
ZB2	286.9	333.27	369.15	364.66	634.03	851.39
ZB3	211.57	422.15	450.24	448.55	727.52	930.16
ZB5	301.04	387.1	403.68	401.64	686.27	895.88
ZB6	261.74	463.31	485.14	477.86	788.64	990.45
ZB7	384.69	429.39	437.93	435.41	688.5	893.92

Table 4.1 Block-Quay Distances in metres

	Block	**CP3**	**S1A**	**S1B**	**S1C**	**S1D**	**S1E**	**S1F**	**S1G**	**S1H**
Gate	**Gate 1**	752.17	127.2	129.7	195.71	223.95	271.66	335.16	383.96	448.96
	Gate 2	491,04	503.01	517.35	426.8	446,66	358,77	257.82	231.1	188.96

	Block	S1J	Z4	Z3A	Z3B	ZB2	ZB3	ZB5	ZB6	ZB7
Gate	**Gate 1**	489.7	967.32	877.5	812.57	598.61	509.22	648.91	559.26	673.68
	Gate 2	227.99	717.65	615.83	554.67	256.85	164.4	217.27	122.1	197.12

Table 4.2 Block-Gate Distances in metres

Regarding the capacity of the blocks and the yard in it its entirety, it is possible to refer to table 4.3 which shows the code of the 18 blocks, the size of containers that each block can accommodate, its number of rows, bays and tiers and its total capacity in terms of number of containers (which is calculated as the number of bays times the number of rows times the number of tiers):

Block Name	Size	Bays	Rows	Tiers	Block Capacity
ZB7	40’	8	3	4	96
ZB5	40’	9	3	4	108
ZB6	40’	7	3	4	84
S1A	40’	7	6	4	168
S1C	40’	2	6	4	48
S1D	40’	4	6	4	96
S1E	40’ or 20’	7 or 13	4	4	112 or 208
S1J	40’	3	6	4	72
S1H	40’	3	6	4	72
S1G	40’	3	6	4	72
S1F	40’	3	6	4	72
Z4	40’	15	6	4	360
ZB2	20’	17	3	5	255
ZB3	20’	17	3	5	255
S1B	20’	5	6	5	150
CP3	40’ or 20’	12 or 24	3	4	144 or 288
Z3B	40’ or 20’	3 or 5	6	4	72 or 120
Z3A	40’ or 20’	3 or 5	6	4	72 or 120

Table 4.3 Blocks Layout

It is worth noting that a certain number of blocks can accommodate containers of two different sizes. In three of the four cases (S1E, Z3B, Z3A), the number of bays for 20’ containers is odd. The available data about the blocks had a conceptual error in the calculation of their capacity: the capacity for 40’ containers was calculated as the product of bays, rows and tiers using the same number of bays as for 20’ containers and then dividing the result by 2. This corresponds to dividing the number of 20’ bays by 2, which might seem correct since a 40’ container occupies twice the space of a 20’ container (hence occupying two 20’ bays at the same time) Whilst this is not a problem if the number of 20’ bays is even, results in a rational number in the case of an odd number of 20’ bays. Therefore, an assumption was made: the rational number was rounded up to the closest natural number.

4.4 Information about the Containers

The attributes and characteristics of the incoming container are derived from a database that served as the basis for the model proposed by Maldonado et al. (2019). In this work, which focused on the same Port of Arica, the allocation of a series of import containers was based on their predicted dwell time. The prediction was derived from the application of peculiar analytic techniques. The retrieval of the containers, on the other hand, was based on the actual dwell time, measured directly in the yard on the TPA. This approach proved to be really competitive in terms of reshuffles with regards to the current practices in use at the Chilean terminal.

The analytical prediction of the dwell time was performed for 1591 containers and it was stored in a dataset that contains all the available information on the containers that have travelled through the yard in reality, from the unloading from the ship to their retrieval from the stack. For each container, the available data are:

- The Code of the Container: an alphanumeric code which represents an identifier of the container itself
- The Real and the Predicted Dwell time in discrete terms: three intervals are defined for both cases (less than 7 days, between 7 and 14 days, more than 14 days).
- The Real and Predicted Dwell time in hours. The Real Dwell time comes from a real-life measurement of the time spent by each container in the yard.
- The month in which the container arrived in the yard (January, February etc.). However, nothing is known in relation to the exact day-in of each container.
- The error of estimation of the dwell time expressed in days and hours (The error is calculated as a simple difference between Real and Predicted Dwell time in hours).
- A column which states whether the prediction of the discretised dwell time is verified or not (indicated with a simple yes or no).

- The size of the incoming containers (20’ so a TEU or 40’ so a FEU)
- The state of the load of the containers which means whether they are full or empty.
- A code which is generated each time a container leaves the yard through the gate. It is used in relation to the external trucks.
- The port from which the incoming container arrived
- The name of the ship which carried the container
- The berthing site where the ship moored

An example of those data for one container is presented in Table 4.4.

Container ID	**Discrete Real Dwell time**	**Real Dwell time (in hours)**	**Discrete Predicted Dwell time**	**Predicted Dwell time (in hours)**	**Error (in hours)**	**Error (in days)**	**Correctness of the prediction**
FSCU414499-9	between 7 and 14 days	195.43	between 7 and 14 days	248.2770428	53	2	Yes

Size	**State**	**Despatch Type**	**Port**	**Month**	**Despatching code**	**Unloading Ship**	**Quay**
40GP	Full	Indirect	New York	January	1039853	MSC LEANNE \ UW602R	1

Table 4.4 An example of the information available in the database for each container

4.5 The Resources

At the moment of the development of the model, information about the resources of the yard (and vehicles in particular) can be found in the 2018 Annual Report of TPA. The company states that the terminal possesses:

- 4 Quay Cranes (QCs)
- 4 Reach-stackers and 2 forklifts
- 5 Yard trucks with a semi-trailer, 15 Yard trucks and 18 semi-trailers

In addition to this there are other infrastructures for the transfer of bulk.

For the sake of simplicity of the developed model, an assumption was made regarding the infrastructures that serve each block during the stacking or retrieval phase. First of all, the number of available blocks for the stacking of import containers is reduced from 18 to 9 (the other 9 blocks are supposed to be dedicated to export containers). The rationale behind this choice will be explained later. Focusing on the import containers, each of the 9 blocks is associated with a reach-stacker which is responsible for the stacking phase. Another reach-stacker is assumed to be dedicated to the retrieval phase: the two vehicles are supposed to operate independently of each other, with no clashing of the two operations which means that whether a retrieval is happening during the stacking of an incoming container or not, there is no difference in operational terms and vice-versa. The reach-stackers dedicated to the stacking operation are assumed to be three and coded as 101, 102 and 103. Each reach-stacker serves a group of blocks located close-by, which is shown in Table 4.5 (the association mentioned above). Moreover, when one of the three reach-stackers breaks down or is not working, the reach-stacker operating on the closest blocks to affected one comes to help, controlling the allocation of both groups. Hence, each reach-stacker is associated with two group of blocks: the one served during normal operations and the one that is served during an emergency situation due to a breakdown. This is highlighted in Table 4.6 and 4.7.

Reach-Stacker code	**Blocks Served**	**Additional Blocks served during a breakdown**
101	ZB7, ZB5, ZB2	ZB3, ZB6, S1J
102	ZB6, S1J, ZB3	ZB2, ZB5, ZB7, S1F, S1G, S1H
103	S1H, S1G, S1F	-

Table 4.5 Reach-stacker - Block association under normal conditions

Broken Reach-stacker	101	102	103
Helping Reach-stacker	102	101	102

Table 4.6 This table shows which reach-stacker substitutes a broken one. The first line presents the broken reach-stacker while the second shows which reach-stacker substitutes the broken one

	Normal Operations			**101 Down**			**102 Down**			**103 Down**		
	Reach-stacker code			**Reach-stacker code**			**Reach-stacker code**			**Reach-stacker code**		
Block	**101**	**102**	**103**	**101**	**102**	**103**	**101**	**102**	**103**	**101**	**102**	**103**
ZB2	●				●		●			●		
ZB5	●				●		●			●		
ZB7	●				●		●			●		
ZB3		●			●		●				●	
ZB6		●			●		●				●	
S1J		●			●		●				●	
S1F			●			●			●		●	
S1G			●			●			●		●	
S1H			●			●			●		●	

Table 4.7 This table is a comprehensive view on how the reach-stackers operate in relation to the blocks. The dots at the intersection between lines and columns show that the block that corresponds to the line is served by the reach-stacker of the respective column. The red dots represent an emergency situation (during a breakdown) and highlight that they are served by a substitute reach-stacker.

CHAPTER 5

Decision Support using Fuzzy Logic

This chapter starts with an introduction to Fuzzy Logic, followed by a detailed description of a Fuzzy Inference System and how it can be applied to the container stacking problem. Following this, a set of stacking criteria are developed and presented in detail. Finally, the criteria are combined together to form the so-called Fuzzy Systems

5.1 An Introduction to Fuzzy Logic

As defined by Ries et al. (2014), Fuzzy Logic is, in a nutshell, a rule-based approach which allows to associate crisp values of a certain variable to a variety of linguistic terms and to introduce rules in order to obtain the value of interest.

It was formalised for the first time by Zadeh (1965) and it arose from the consideration that real life classes of objects do not have a precise definition of the criteria of membership and they have rather unclear boundaries. An example of this, proposed again by Zadeh (1965), is the class of tall men: while a man who is 2.00 m tall is very likely to belong to that class with a high degree of membership, what can be said about a man who is 1.80 m tall? Or 1.75 m tall? Despite not being a proper set in the mathematical sense of the word, classes like the one of tall men still exert a fundamental role in human thinking and the communication of information. The same reasons are valid for a container terminal: what does long mean in terms of distance in the yard? How short is a short dwell time? The importance of having those definitions clear has been explained in the previous chapters, where the two variables have been presented as two of the main criteria for allocation that can be used in a port.

Zadeh (1965) tried to address this issue by giving a mathematical definition of a new kind of set with unsharp boundaries, the fuzzy set:

$$A = \{x, f_a(x) \mid x \in X\} \tag{5.1}$$

(let X be a space of points or objects where a generic element is denoted as x, so that X = {x}. X is called universe of discourse) a fuzzy set A in X is characterised by a membership function called $f_a(x)$ which associates every point in X with a real number in the interval [0, 1]. The value of $f_a(x)$ at x, which is that real number comprised between 0 and 1, represents the grade of membership of x in A. An example of five fuzzy sets for the variable Distance is represented in Figure 5.1, each one with a triangular membership function.

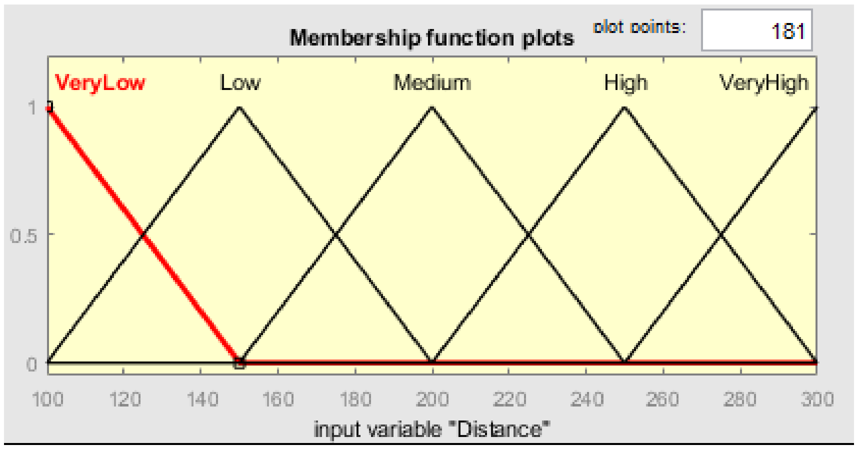


Figure 5.1 A representation of five triangular membership functions (Very Low, Low, Medium, High and Very High) for the variable Distance. On the y-axis is represented the value of the membership function

The author also defined three main operations between fuzzy sets: union, intersection and complement. Before presenting them, however, it is important to also give the definition of containment:

A is contained in B (or, equivalently, A is a subset of B, or A is smaller than or equal to B) if and only if $f_A \leq f_B$. In symbols:

$$A \subset B \Leftrightarrow f_A \leq f_B \tag{5.2}$$

The union of two fuzzy sets called A and B, with respective membership functions $f_A(x)$ and $f_B(x)$, is a fuzzy set C, written as $C = A \cup B$, whose membership function is related to those of A and B by

$$f_C(x) = \text{Max}\,[f_A(x), f_B(x)], \quad x \in X \tag{5.3}$$

A more direct and intuitive definition of union is the following: the union of sets A and B is the smallest fuzzy set that contains both A and B.

The intersection of two fuzzy sets A and B, with respective membership functions $f_A(x)$ and $f_B(x)$ is a fuzzy set C, written as $C = A \cap B$, whose membership function is related to those of A and B by

$$f_C(x) = \text{Min}\,[f_A(x), f_B(x)], \quad x \in X \tag{5.4}$$

In the same way as for the union, the intersection between the sets A and B can be defined also as the largest fuzzy set contained in both A and B.

The complement of a fuzzy set A is denoted by A' and is defined by

$$f_{A'}(x) = 1 - f_A \,, \quad x \in X \tag{5.5}$$

Union and intersection are represented in Figure 5.2: f_A and f_B are the membership functions of two different fuzzy sets, A and B. The membership function of the union is represented in blue by segments 1 and 2 while the intersection is represented in red by segments 3 and 4.

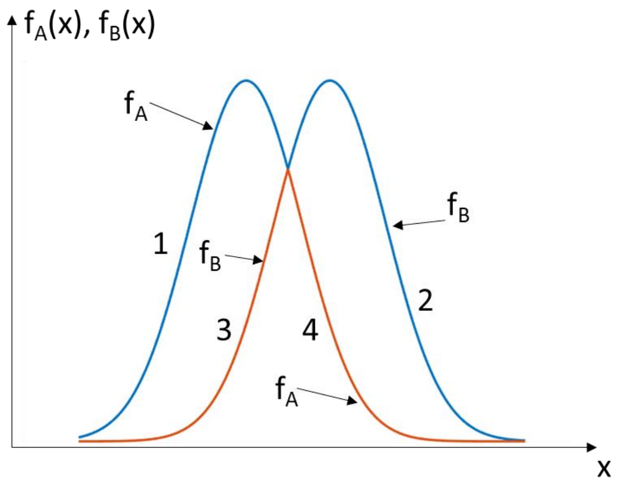


Figure 5.2 Illustration of union and intersection of two fuzzy sets with f_A and f_B as their respective membership functions

Complement is represented in Figure 5.3: f_A and $f_{A'}$ are the membership functions of fuzzy set A and its complement respectively. The membership function of fuzzy set A is represented in blue while the membership function of its complement is drawn in red.

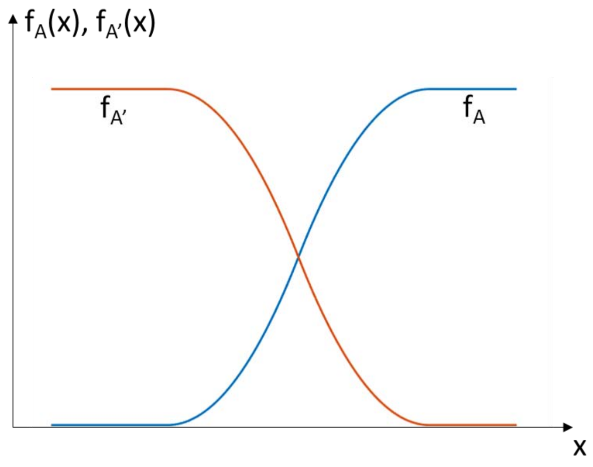


Figure 5.3 Illustration of the complement of fuzzy set A

According to Jang and Gulley (1995), if the values of the membership functions are kept at their extremes, 0 (which corresponds to completely false) and 1 (corresponding to completely true), the three operators that were introduced before are able to preserve the results of the standard Boolean logic operations. In particular, it is possible to draw the following correspondence between two-valued and multivalued logical operations:

- Fuzzy union, defined with the function *max*, and the Boolean operator OR
- Fuzzy intersection, defined with the function *min*, and the Boolean operator AND
- Fuzzy complement and the Boolean operator NOT

This can be highlighted by the tables of truth that are presented below, where false and true correspond to the extreme values 0 and 1 respectively.

Boolean OR

A	B	A OR B
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**1**

Table 5.1 Truth table of the Boolean operator OR

Fuzzy Union

A	B	max (A, B)
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**1**

Table 5.2 Truth table of Fuzzy Union

Boolean AND

A	B	A AND B
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**1**

Table 5.3 Truth table of the Boolean operator AND

Fuzzy Intersection

A	B	min (A, B)
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**1**

Table 5.4 Truth table of Fuzzy Intersection

Boolean NOT

A	NOT A
0	**1**
1	**0**

Table 5.5 Truth table of the Boolean operator NOT

Fuzzy Complement

A	1 - A
0	**1**
1	**0**

Table 5.6 Truth table of Fuzzy Complement

The truth tables show the same results for the fuzzy operators and their corresponding Boolean operations.

However, in fuzzy logic, the truth of a statement is a matter of degree since, as introduced before, the membership functions of the fuzzy sets can assume all the values in the interval [0 1] and not only its extremes. Thus, the truth tables can be substituted by the plots of the membership functions themselves and the logical operations between them are now entirely fuzzy (their input values can be any real number between 0 and 1): fuzzy OR (corresponding to the union), fuzzy AND (corresponding to the intersection) and fuzzy NOT (corresponding to the complement). Therefore, any logical construction can be resolved with those three operations and the fuzzy sets. This makes fuzzy logic a superset of the standard Boolean logic.

It is important to note that the correspondence existing between two-valued and fuzzy logical operations is not unique. As a matter of fact, fuzzy union (OR) and intersection (AND) can be defined with different functions other than *max* and *min* respectively. Despite the possibility of customising those functions, given by Matlab specific Fuzzy Logic Toolbox, in this work the traditional definitions and functions were used.

As stated by Jang and Gulley (1995), fuzzy sets and fuzzy operations are the subjects and verbs of fuzzy logic. In order to formulate a conditional statement, these elements are combined through if-then rules. The general structure of an if-then rule is:

$$\text{IF } x \text{ is } A \text{ AND/OR } y \text{ is } B\text{, THEN } z \text{ is } C$$

where A, B and C are linguistic values defined by fuzzy sets which belong to their respective universes of discourse, X, Y and Z, x and y are input variables and z is the output variable. Within the rule, the part that starts with the word IF and ends before the word THEN is called antecedent or premise while the part that follows the word THEN is called consequent or conclusion. Despite being fuzzy sets, in the structure of the rule, A and B are represented as numbers comprised between 0 and 1; they are then combined into a single number between 0 and 1, which is the outcome of the antecedent, making it an interpretation. On the other hand, C is an entire fuzzy set, which is assigned to the output variable: this makes the consequent an assignment. Consequently, the verb "*is*" assumes two different roles whether it appears in the antecedent or in the consequent.

On a more general level, interpreting an if-then rule can be seen as a two-part process:

1. Evaluation of the antecedent

a. Fuzzification of the input(s). All the statements in the antecedent are resolved into a number between 0 and 1, according to the value of the respective input variable.

b. Application of the fuzzy operators. If there are two or more fuzzy statements, all of them are fuzzified at the same time; then they are combined into one single number comprised between 0 and 1 using the fuzzy logical operators. This single number is the result the of the evaluation of the antecedent.

2. Application of the implication method to the consequent: it consists of assigning a fuzzy set to the output variable. That fuzzy set is then modified according to the implication function, which accounts for the impact of the antecedent on the consequent. In the case of multiple consequents, they are all affected equally by the antecedent. But what is meant by impact of the antecedent? In the standard Boolean logic, if the antecedent is true, so is the consequent; in fuzzy logic, as mentioned before, truth is a matter of degree so if the antecedent is true to a certain degree of membership, the consequent is true to the same degree. Therefore, the fuzzy set assigned to the output variable has to be modified accordingly. Two of the most common ways of doing this are truncation and scaling.

Usually, more than one if-then rules are usually put in place. Their outcomes, the fuzzy sets, are then aggregated into one single output fuzzy set. In order to obtain a single crisp number to assign to the output variable, this final fuzzy set is then defuzzified. A more detailed explanation of the whole process is presented in the next sub-section, which is focused on the Fuzzy Inference Process.

5.2 The Fuzzy Inference Process

Fuzzy Inference is the process which, given a certain input, provides the corresponding output using fuzzy logic. It ties together all the elements that were introduced in the previous section. According to Jang and Gulley (1995), it is five-phase process, where the first three correspond exactly to the interpretation of an if-then rule:

1. Fuzzification of the inputs
2. Application of the fuzzy operators
3. Application of the implication method
4. Aggregation of the outputs of each rule

5. Defuzzification

The process and each of its phases are explained through an example set in the port of Arica. Let's consider the block assignment problem: it consists of finding the best possible block in the yard to allocate an incoming import container. To solve this problem, a Decision Rule with two criteria is used: Distance and Block Utilisation (it corresponds to the so-called B12 block assignment policy which will be introduced later in the chapter). This Decision Rule is implemented through fuzzy logic and its two criteria coincide with the two input variables of the Fuzzy Inference Process: block-gate distance and utilisation of one block in the yard. The output variable is the Value of Goodness of the block (Block VoG), which is a way to assess the validity of a certain block assignment: the higher the VoG, the better it is to allocate the incoming container to the considered block.

For the first input variable, Distance, the universe of discourse is represented by all the possible block to gate distances in the range comprised between 100 m and 300 m. Within this universe, five fuzzy sets can be created and each of them is assigned a linguistic term: Very Low, Low, Medium, High, Very High. Each one of them is represented by a triangular membership function, defined in the following way:

$$f_{SET}(x, Min, Med, Max) = \begin{cases} 0 & x < Min \\ \dfrac{x - Min}{Med - Min} & Min \leq x \leq Med \\ \dfrac{Max - x}{Max - Med} & Med < x \leq Max \\ 0 & x > Max \end{cases} \tag{5.6}$$

where *x* is the input variable, *Med* is the value at which the membership function peaks, *Min* is the value that corresponds to the left apex of the triangle representing the membership function and *Max* is the value that corresponds to the right apex. Fuzzy sets and their membership functions are represented in Figure 5.4. The values of the parameters *Min*, *Med* and *Max* are reported in Table 5.7 for each fuzzy set.

Regarding the second input variable Block Utilisation, the universe of discourse is represented by the range [0 1]. It coincides with all the possible values that can be assumed by the share of container capacity of a block which is calculated as a ratio between the occupied containers slots and the total number of slots in the block. Within the universe, three sets are created, associated with the terms Low, Medium and High.

Fuzzy Set	**Min**	**Med**	**Max**
Very Low	100	100	150
Low	100	150	200
Medium	150	200	250
High	200	250	300
Very High	250	300	300

Table 5.7 Parameters of the membership functions of the fuzzy sets for Distance

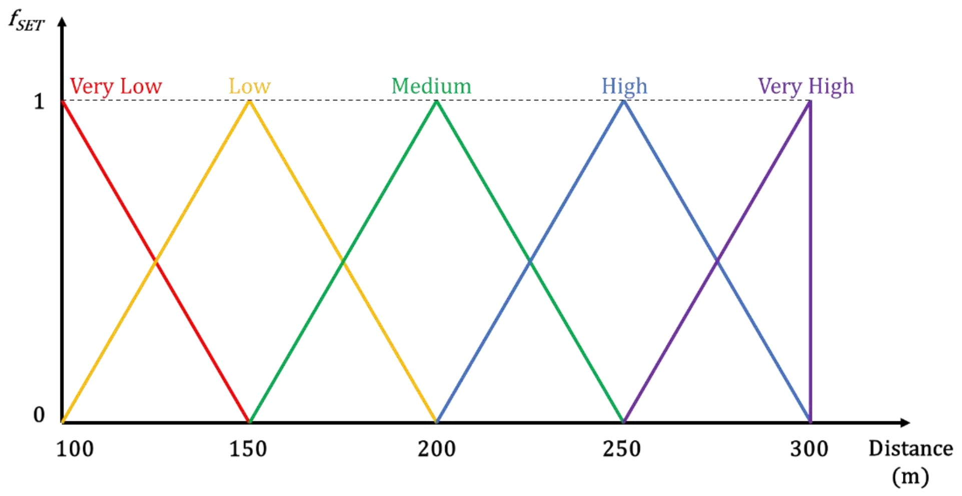


Figure 5.4 Representation of the membership functions of the fuzzy sets for Distance

Each fuzzy set is represented through a triangular membership function defined as in (5.6). The parameters of the membership functions are reported in Table 5.8 and the fuzzy sets are illustrated in Figure 5.5.

Finally, with regards to the output variable, Block VoG, the universe of discourse is represented by the interval [0 1]. It corresponds to all the possible values that can be assumed by Block VoG after defuzzification. Within the range, five fuzzy sets can be created and each one of them is described by a linguistic term: Very Low, Low, Medium, High and Very High. All of the sets are represented by a triangular membership function, defined as in (5.6). The membership functions are illustrated in Figure 5.6 and their parameters are reported in Table 5.9.

Fuzzy Set	**Min**	**Med**	**Max**
Low	0	0	0.4
Medium	0.1	0.5	0.9
High	0.6	1	1

Table 5.8 Parameters of the membership functions of the fuzzy sets for Block Utilisation

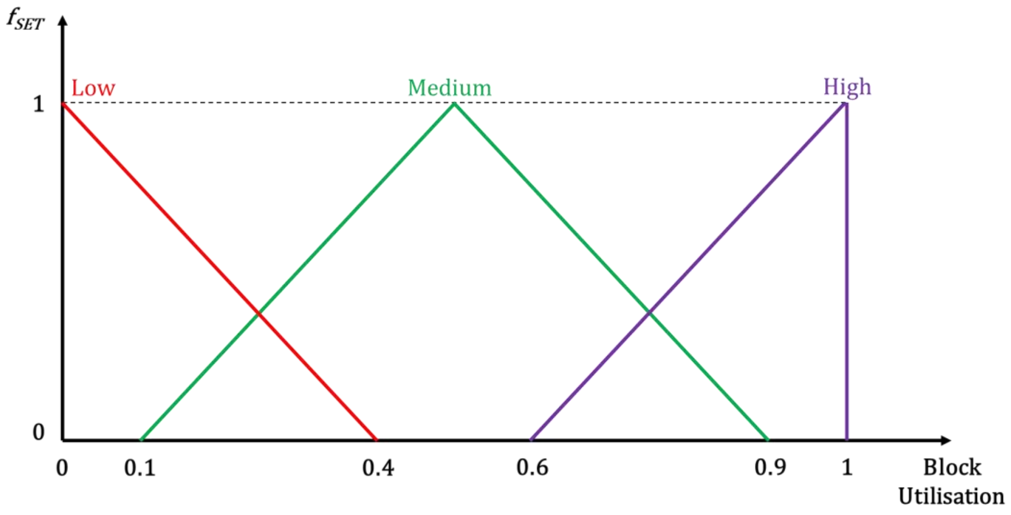


Figure 5.5 Representation of the membership functions of the fuzzy sets for Block Utilisation

Fuzzy Set	**Min**	**Med**	**Max**
Very Low	0	0	0.25
Low	0	0.25	0.5
Medium	0.25	0.5	0.75
High	0.5	0.75	1
Very High	0.75	1	1

Table 5.9 Parameters of the membership functions of the fuzzy sets for Block VoG

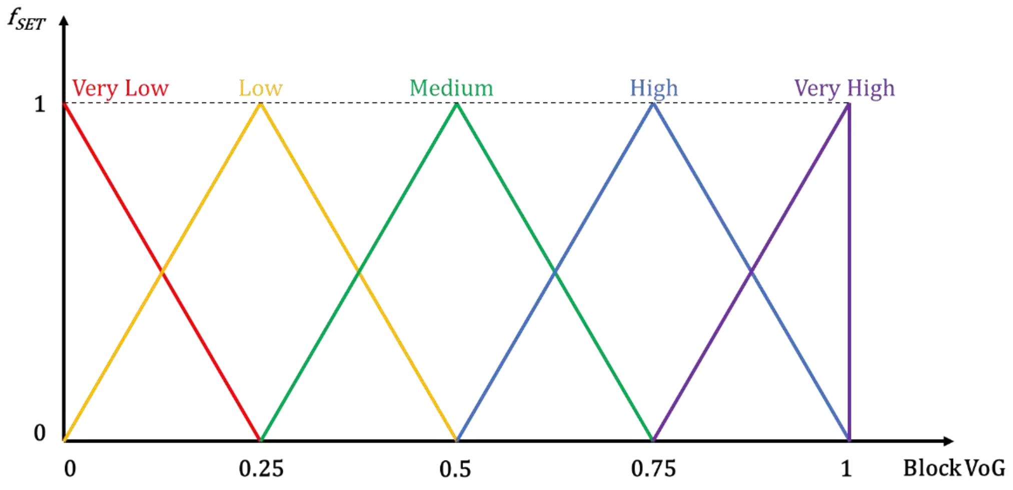


Figure 5.6 Representation of the membership functions of the fuzzy sets for Block VoG

Input and output are connected through a list of if-then rules, which are enumerated in Table 5.10. All the rules are in the shape of:

IF *Distance* is *A* AND *Block Utilisation* is *B*, THEN *Block VoG* is *C*

where the fuzzy operator is always AND. In Table 5.10 each line represents a rule while the columns report the linguistic values that are assumed by the input variables Distance and Block Utilisation and the output variable Block VoG for each rule.

Rule Number	Block-Gate Distance	Block Utilisation	Block VoG
1	Very Low	Low	Very High
2	Very Low	Medium	Very High
3	Very Low	High	High
4	Low	Low	Very High
5	Low	Medium	Very High
6	Low	High	High
7	Medium	Low	High
8	Medium	Medium	High
9	Medium	High	Medium
10	High	Low	Medium
11	High	Medium	Medium
12	High	High	Low
13	Very High	Low	Medium
14	Very High	Medium	Medium
15	Very High	High	Very Low

Table 5.10 Rule base for Block Assignment

Once the input and output variables and the fuzzy sets are defined in their entirety, it is possible to analyse the Inference Process step-by-step. In the example, the validity of the block assignment is evaluated for Block ZB3: its distance from Gate 2 is 164.4 m and its utilisation is 0.43 (which means that the block is full at 43% of its capacity).

5.2.1 Fuzzification of the inputs

The first step of the process consists of taking all the inputs, represented by the values of the input variables, and determining the degree to which they belong to the appropriate fuzzy sets through the membership functions. According to Jang and Gulley (1995), this amounts to a very simple function evaluation: given the crisp numerical value of the input variable, the degree of belonging to a certain fuzzy set is calculated using the relative equation that describes its membership function.

Regarding the example, the degree of membership for each set is calculated through eq. (5.6) using the respective parameters listed in Table 5.7. For instance, a block-gate distance of 164.4 m is located between (*Med* < 164.4 m < *Max*) *Med* (150 m) and *Max* (200 m) for the fuzzy set Low and belongs to such set with a degree of:

$$\frac{200 - 164.4}{200 - 150} = 0.712 \tag{5.7}$$

The degree of membership for all the sets are reported in Table 5.11.

Fuzzy Set	f_{SET}
Very Low	0
Low	0.712
Medium	0.288
High	0
Very High	0

Table 5.11 The degree of membership to each fuzzy set for Distance

Therefore, in linguistic terms it could be said that a block-gate distance of 164.4 m of Block ZB3 is Low to the degree 0.712 and Medium to the degree 0.288. This result show how the value of an input variable can belong to two or more different fuzzy sets contemporarily, with different degrees of membership. The same calculations are applied to the other input variable of the example, Block Utilisation. The degree of membership to the respective fuzzy sets are reported in Table 5.12. In this way, each input is fuzzified over all the membership functions required by the rules applied by the process.

Two examples of the process of fuzzification for the variable Distance are illustrated in Fig. 5.7 and Fig. 5.8.

Fuzzy Set	f_{SET}
Low	0
Medium	0.825
High	0

Table 5.12 The degree of membership to each fuzzy set for Block Utilisation

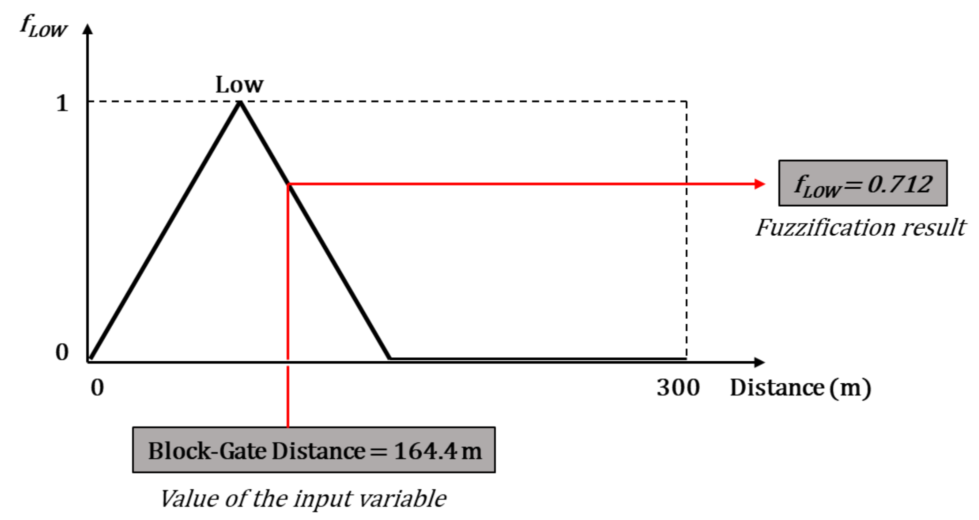


Figure 5.7 Illustration of the fuzzification process for the input variable Distance with regards to the set Low

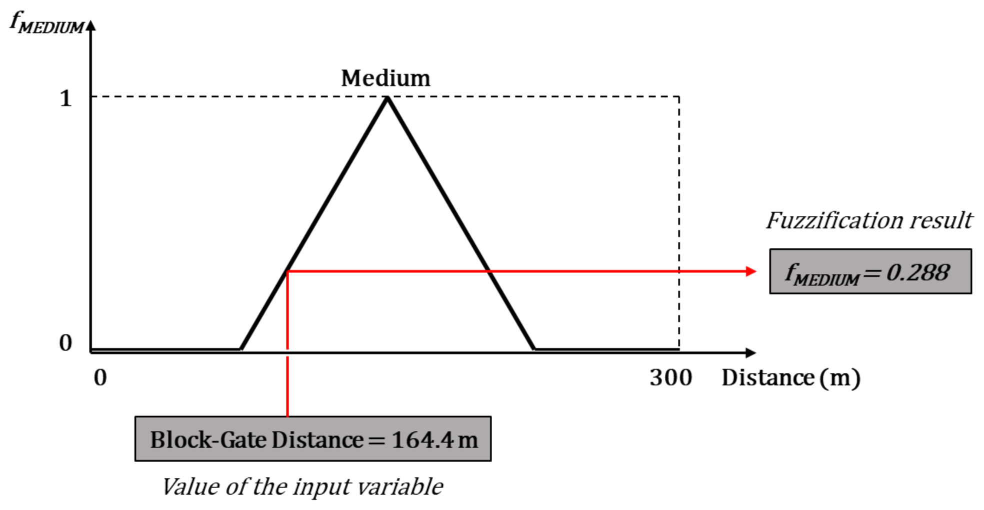


Figure 5.8 Illustration of the fuzzification process for the input variable Distance with regards to the set Medium

5.2.2 Application of the fuzzy operators

Once all the inputs have been fuzzified, it is time to start looking at the rules. However, when the antecedents of the if-then rules are constituted by more than one part, as in the example, it is necessary to resolve them to a single number between 0 and 1 in order to apply the implication method to the consequent. To do this, fuzzy operators are applied. As mentioned in section 4.1, many different functions can be used to fill in for those logical operations. The ones considered in this work are *min* (minimum) for AND and *max* (maximum) for OR.

With regards to the example, it is interesting to focus on the evaluation of the antecedent for rule number 5, where both fuzzified inputs are different from 0. In fact, the fuzzification process for Distance yielded a membership value for the set Low of 0.712 and for Block Utilisation gave a membership value of 0.825 for the set Medium. The fuzzy operator of the rule, fuzzy AND, selects the minimum between the two:

$$\min(0.712, 0.825) = 0.712 \tag{5.7}$$

The result of the application of the fuzzy operators on the antecedent is then 0.712. The application of the operators is illustrated below in Figure 5.9.

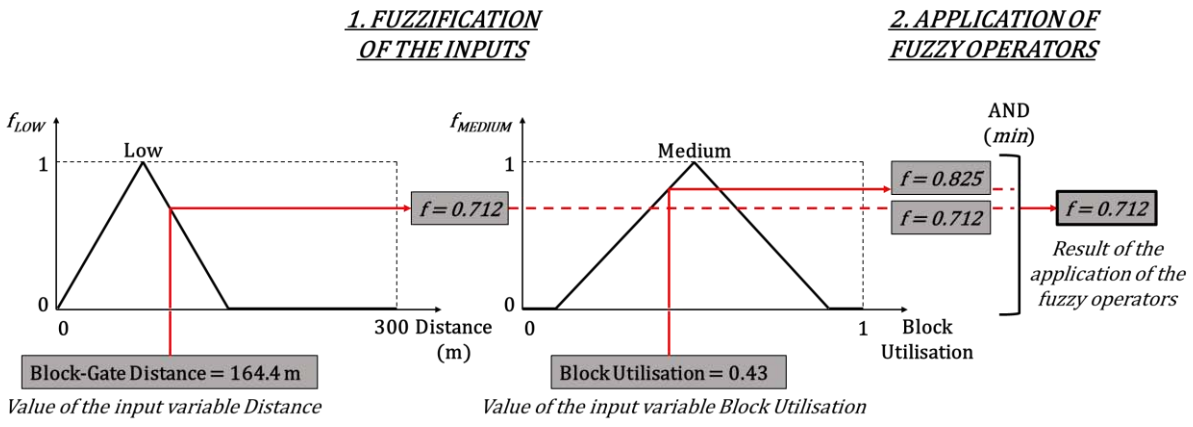

1. FUZZIFICATION OF THE INPUTS
2. APPLICATION OF FUZZY OPERATORS
f_{LOW}
Low
1
0
0
300 Distance (m)
f_{MEDIUM}
Medium
1
0
0
1
Block Utilisation
AND (min)
f = 0.712
f = 0.825
f = 0.712
f = 0.712
Result of the application of the fuzzy operators
Block-Gate Distance = 164.4 m
Value of the input variable Distance
Block Utilisation = 0.43
Value of the input variable Block Utilisation

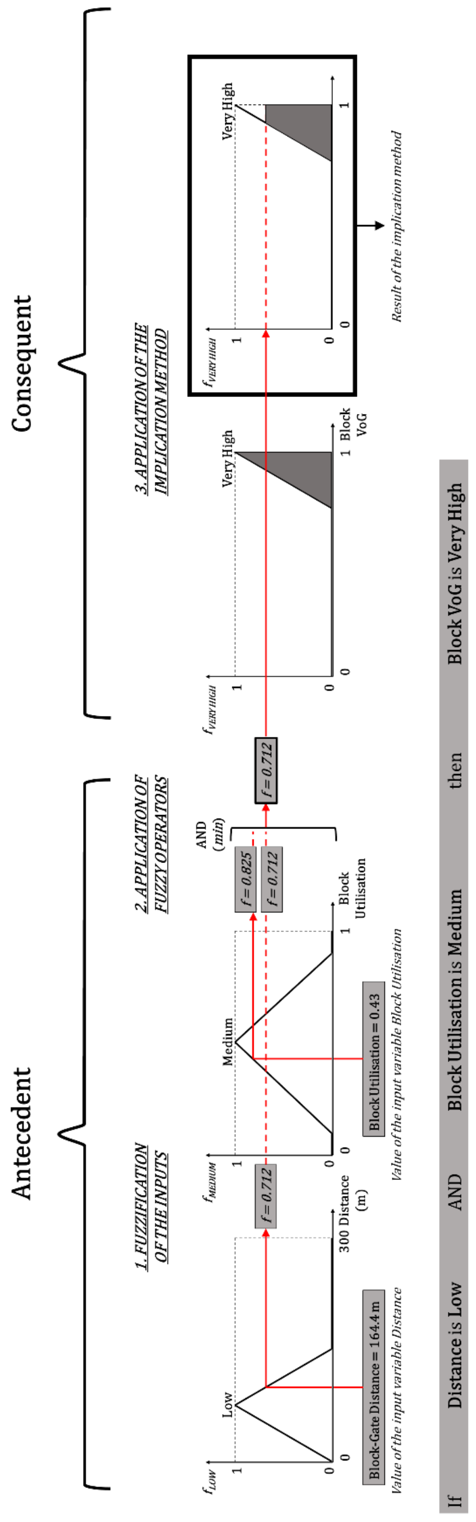
Figure 5.9 Illustration of the application of the fuzzy operators

5.2.3 Application of the implication method

Before applying the implication method to the consequent, it is necessary to address the weight of the rules. In fact, each rule is associated with a weight, which is used to tune the rule base according to specific needs, increasing the importance of certain rules and reducing the impact of others. The weight is a number between 0 and 1 and is applied to the result of the antecedent, which in the example is 0.712, by multiplying the two numbers. All the weights of the rules of the example are set to 1, so the result after the evaluation of the rule weight for rule number 5 is:

$$Result\ of\ antecedent\ \cdot Weight\ of\ the\ Rule = 0.712\ \cdot 1 = 0.712 \tag{5.8}$$

Once the result of the antecedent has been weighted, it then serves as the input for the implication method. The method consists of modifying the consequent of each rule according to the value of the antecedent. Therefore, the output variable is assigned the respective fuzzy set for each if-then rule, shaped according to the weighted antecedent result. There are two main ways of re-shaping the fuzzy set of the output variable: using the function *min* (minimum), which corresponds to the truncation of the output fuzzy set, or the function *prod* (product), which corresponds to a scaling of the output fuzzy set. The implication method is applied to each rule of the rule base and its application for rule number 5 is illustrated in Figure 5.10.



Antecedent
Consequent
1. FUZZIFICATION OF THE INPUTS
2. APPLICATION OF FUZZY OPERATORS
3. APPLICATION OF THE IMPLICATION METHOD
f_{LOW}
Low
1
0
0
300 Distance (m)
Block-Gate Distance = 164.4 m
Value of the input variable Distance
f = 0.712
f_{MEDIUM}
Medium
1
Block Utilisation
Block Utilisation = 0.43
Value of the input variable Block Utilisation
f = 0.825
f = 0.712
AND (min)
f = 0.712
$f_{VERY HIGH}$
Very High
1 Block VoG
Result of the implication method
If
Distance is Low
AND
Block Utilisation is Medium
then
Block VoG is Very High

Figure 5.10 Illustration of the application of the implication method for rule 5

It is interesting to see how the membership function of the set Very High for the output variable Block VoG is truncated: a triangular membership function is turned into a trapezoidal function whose height corresponds exactly to the result of the antecedent.

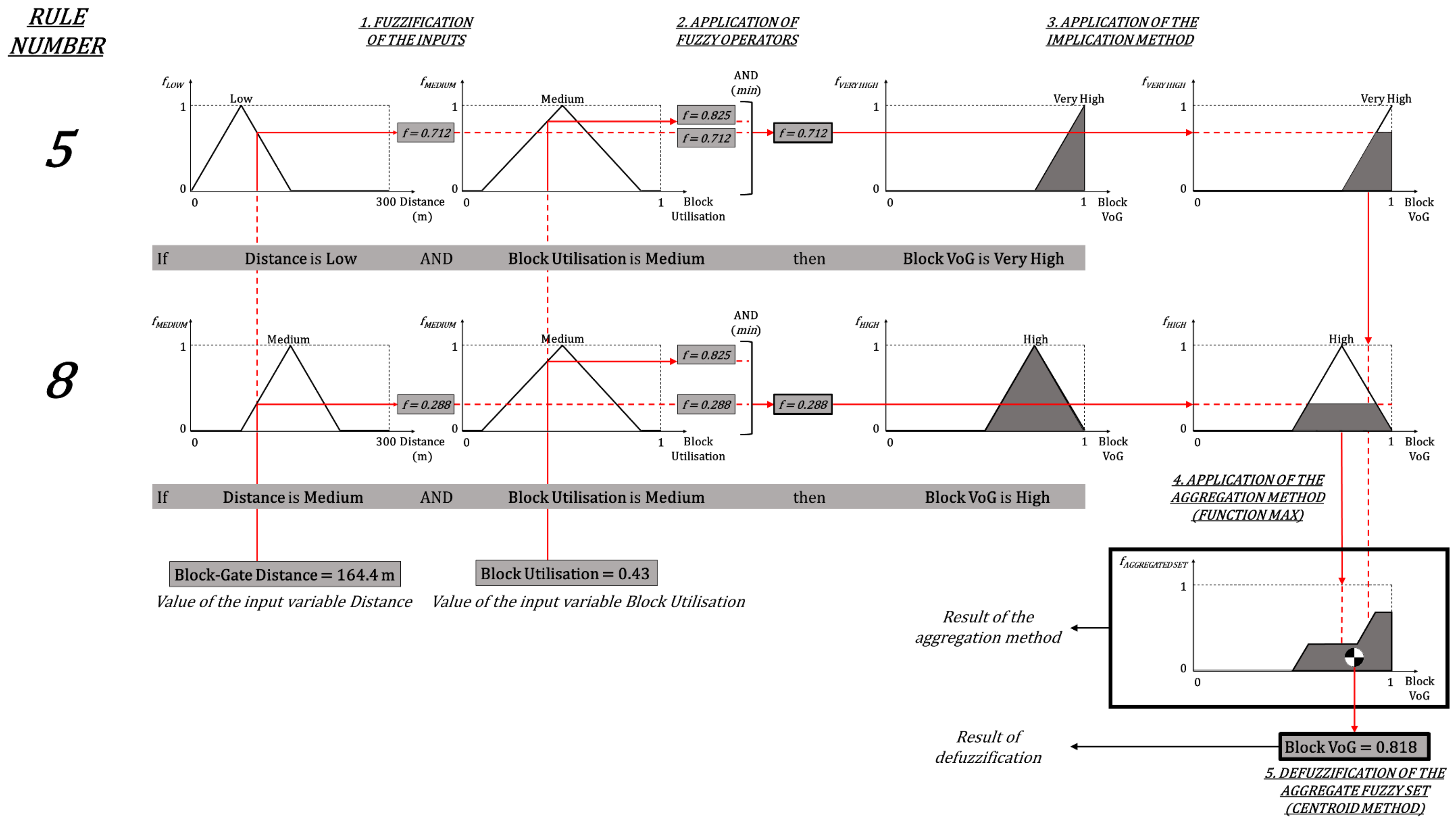
5.2.4 Aggregation of all the outputs

The fourth step is aggregation, which corresponds to the unification of the outputs of each rule. These outputs are the truncated fuzzy sets returned by the implication process and they are combined together into one single fuzzy set, in preparation for the final defuzzification. In this way, each output variable is assigned one single combined fuzzy set. There are three main functions that are commonly used to implement the aggregation process: *max* (maximum), *probor* (probabilistic or) and *sum* (summing all the membership functions of each output fuzzy set). It is fundamental to note that the aggregation method is commutative so the order in which the outputs of the rules, and so the order in which the rules are executed, is irrelevant.

Figure 5.11 illustrates the aggregation process in the proposed example. Only two rules and their relative outputs are shown: in fact, only rule number 5 and 8 yield a fuzzy set whose membership function is different from 0. In all the other cases, the resulting fuzzy set of each rule is irrelevant to the effect of the aggregation. The function that is used to implement the process is *max*: the two trapezoidal membership functions that result from the truncation of the fuzzy sets Very High (rule 5) and High (rule 8) are aggregated into one single set, assigned to the output variable Block VoG.

5.2.5 Defuzzification

While fuzziness helps while dealing with sets with unclear boundaries and during the intermediate steps of the Inference Process, for the purpose of decision making it is important to have one single crisp number assigned to the output variable. At the moment this is not possible, since the output variable is still associated with a combined fuzzy set that encompasses a range of values. Therefore, defuzzification is needed. There are many methods to perform defuzzification: centroid, bisector, middle of maximum, largest of maximum and smallest of maximum. The most commonly used of the five, also applied in the example, is centroid, which calculates the value that corresponds to the coordinate of the centre of the area under the curve of the membership function of the combined fuzzy set. The resulting number is then assigned to the output variable. The Fuzzy Inference Process, therefore, ends with a single crisp output.

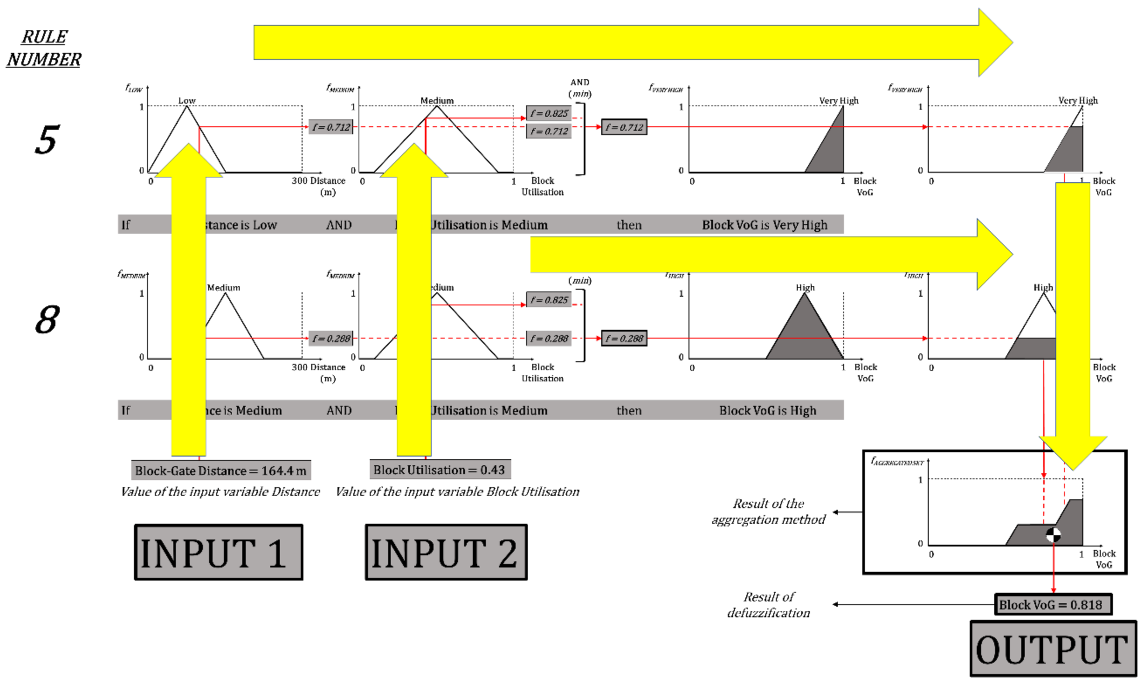


RULE NUMBER
1. FUZZIFICATION OF THE INPUTS
2. APPLICATION OF FUZZY OPERATORS
3. APPLICATION OF THE IMPLICATION METHOD
5
f_{LOW}
Low
1
0
0
300 Distance (m)
$f = 0.712$
f_{MEDIUM}
Medium
1
0
0
1 Block Utilisation
AND (min)
$f = 0.825$
$f = 0.712$
$f = 0.712$
$f_{VERY\,HIGH}$
Very High
1
0
0
1 Block VoG
$f_{VERY\,HIGH}$
Very High
1
0
0
1 Block VoG
If Distance is Low AND Block Utilisation is Medium then Block VoG is Very High
8
f_{MEDIUM}
Medium
1
0
0
300 Distance (m)
$f = 0.288$
f_{MEDIUM}
Medium
1
0
0
1 Block Utilisation
AND (min)
$f = 0.825$
$f = 0.288$
$f = 0.288$
f_{HIGH}
High
1
0
0
1 Block VoG
f_{HIGH}
High
1
0
0
1 Block VoG
If Distance is Medium AND Block Utilisation is Medium then Block VoG is High
4. APPLICATION OF THE AGGREGATION METHOD (FUNCTION MAX)
Block-Gate Distance = 164.4 m
Value of the input variable Distance
Block Utilisation = 0.43
Value of the input variable Block Utilisation
$f_{AGGREGATED\,SET}$
1
0
0
1 Block VoG
Result of the aggregation method
Result of defuzzification
Block VoG = 0.818
5. DEFUZZIFICATION OF THE AGGREGATE FUZZY SET (CENTROID METHOD)

Figure 5.11 Illustration of the Fuzzy Inference Process, including aggregation and defuzzification

In the example, defuzzification is implemented through the centroid method, resulting in a finale value of the Block VoG of 0.818, as also shown in Figure 5.11. The inference process should then be applied to all the other blocks of the yard, finding the Block VoG for each one of them. The incoming container is then assigned to the block that has the highest Block VoG, meaning that the chosen block has the best combination of Distance and Utilisation for the purpose of allocation.

Aside from including aggregation and defuzzification, Figure 5.11 represents the whole Fuzzy Inference Process. Jang and Gulley (1995) called this graph the Fuzzy Inference Diagram since it is able to show all the different steps of the Inference Process at once. It is also interesting to see how information flows within the diagram, highlighted by the yellow arrows in Figure 5.12.



RULE NUMBER
5
8
If Distance is Low AND Utilisation is Medium then Block VoG is Very High
If Distance is Medium AND Utilisation is Medium then Block VoG is High
f = 0.712
f = 0.825
f = 0.288
AND (min)
Block-Gate Distance = 164.4 m
Value of the input variable Distance
Block Utilisation = 0.43
Value of the input variable Block Utilisation
INPUT 1
INPUT 2
Result of the aggregation method
Result of defuzzification
Block VoG = 0.818
OUTPUT

Figure 5.12 Information flow in the Fuzzy Inference Diagram

Starting from two crisp input values, information flows upwards through the process of fuzzification. Then, it moves across each row, accounting for the evaluation of each rule by application of the fuzzy operators and the implication method. Finally, it flows downwards through the aggregation and defuzzification processes, resulting in a final crisp output. Hence, fuzziness is embedded within the Inference Process but does not appear outside of it, since input and outputs are, as mentioned, crisp values.

5.3 Fuzzy Logic and the Container Allocation Problem

After having introduced fuzzy logic and its Inference Process, it is time to apply it to the problem under examination: the container allocation problem. Fuzzy Logic has already been used to deal with the same problem by Ries et al. (2014). Addressing the allocation problem with fuzzy logic is a fitting choice because of a number of reasons. First of all, fuzziness is an element which is present in the container terminal and affects many of the elements that are you used to evaluate where to stack an incoming container: how "long" is a long block-gate distance? How "many" are many rehandles? What does it mean having a "high" block utilisation? And, above all, how "good" is a block or a stack as a possible destination for the incoming containers? Fuzzy logic allows to perform a multi-criteria evaluation, founded on a robust logical basis: the result of the contemporary comparison of multiple criteria is not based on an empirical trade-off assessment but on the theory of fuzzy sets. Furthermore, the prominent use of linguistic terms in the rule definition is close to human condition and it is particularly helpful for those port managers that frequently make decisions based on those terms, without having the quantitative and theoretical support provided by fuzzy logic. This is even more true in those container terminals, like the case study of the Port of Arica under examination, where the decision on the final allocation position of an incoming container is demanded entirely to the judgement of the operators of cranes and internal vehicles, who resolve the stacking problem either by randomness or using their real-life experience. Finally, if the aim is to create a reactive system that is able to allocate containers also considering the events and disturbances that might affect the operations in the yard, fuzzy logic seems an ideal choice because of its flexibility and ability to deal with imprecise and uncertain data.

Could the allocation problem have been solved without fuzzy logic? The answer is yes, but as reported by Jang and Gulley (1995), Lofti Zadeh, who is considered the father of fuzzy logic, once remarked: "In almost every case you can build the same product without fuzzy logic, but fuzzy is faster and cheaper".

Given the important contribution provided by fuzzy logic to decision-making in a container terminal, the available data about the incoming containers, the layout of the yard and its resources were then examined in order to find out which elements were suitable to build criteria for allocation that match the proposed classification. Those newly defined criteria, which will be explained in detail in the next section, would serve as the inputs for the proposed stacking strategy.

The proposed stacking strategy is divided in two phases, in the same way as for the policy proposed by Ries et al. (2014). The two phases are:

- Phase 1: Block Assignment. In this phase, the focus is on finding the best block in the yard considering the current values of the criteria.
- Phase 2: Stack Assignment. The aim of this phase is to find the best stack within the block that was found in Phase 1, using the current values of the criteria.

The two phases are supposed to happen one immediately after the other for each incoming container: once the container is unloaded from the ship, Block Assignment is performed first; once the best block for allocation is found, then Stack Assignment can be performed, so that the search for the best stack only happens amongst the stacks of the best block.

The core of the proposed system is to use fuzzy logic for both Block and Stack Assignment. In this way, each one of the two phases is implemented through the Fuzzy Inference Process. As mentioned above, the Inference Process is a way to map certain inputs to certain outputs. If the inputs of Block and Stack Assignment are the criteria, what are the outputs? The outputs are two variables, one for each phase, that are consistent with the aim of evaluating multiple criteria at the same time:

- Block Value of Goodness (Block VoG): it is the output variable for Block Assignment and it is a way to assess the overall validity of a block as a possible stacking destination for an incoming container. The higher the VoG for a certain block, the better it is to stack the incoming container in that block.
- Stack Value of Goodness (Stack VoG); it is the output variable for Stack Assignment and it is a way to measure the overall validity of a stack as a possible stacking destination for an incoming container. The higher the VoG for a given stack, the better it is to stack the incoming container in said stack.

Therefore, for Block Assignment, the value of each criteria is evaluated based on the real-time state of the yard, each time a new incoming container arrives; those inputs values are then fuzzified, their membership values are combined through the fuzzy operators of specifically-designed rules and the implication method results in multiple truncated fuzzy sets assigned to Block VoG; those fuzzy sets are aggregated into one single set (using the function *max*) which is finally defuzzified (with the centroid method), resulting in the output value of Block VoG. The Inference Process is applied to all blocks, selecting the block with the highest VoG. Once that is chosen, the Inference Process is then applied to all the stacks in that block, resulting in multiple values of Stack VoG. In the end, the incoming container is allocated in the stack with the highest VoG.

5.4 Block and Stack Assignment and their Rule Bases

This section details the implementation of the stacking policy. It was realised thanks to the Fuzzy Logic Toolbox provided by Matlab. Separating Block from Stack Assignment, each phase is presented by listing the input variables (criteria), the output variables and the rules that link them, with a little insight on the nomenclature.

5.4.1 Block Assignment

5.4.1.1 Input Variables/Criteria

The three input variables for Block Assignment are:

1. **Block-Gate Distance.** The distance between the considered block and Gate 2. An important assumption has been made about which gate to consider: all the external trucks that arrive at the terminal to retrieve containers at the end of their dwell time pass through Gate 2 while Gate 1 is dedicated to export containers and therefore is not of interest.

   Five fuzzy sets were created to describe Block-Gate Distance: Very Low, Low, Medium, High and Very High. They are represented in Figure 5.13. All of them are described by a triangular membership function defined by 5.6 and whose parameters are reported in Table 5.13. It is important to note that the universe of discourse has been modelled considering the distances of only 9 blocks: in fact, those nine blocks are assumed to be dedicated to import containers while the other 9, with their relative distances, are disregarded because they are assigned to export containers. This will be explained in detail in the next chapter.

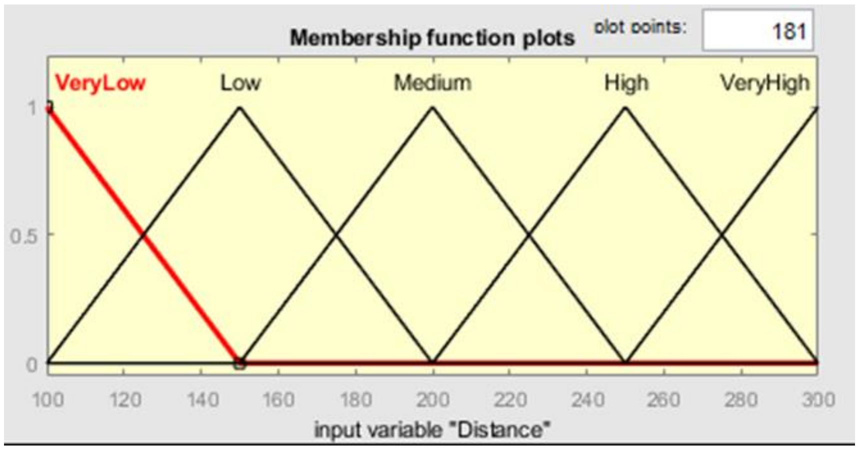


Figure 5.13 Illustration of the fuzzy sets and their membership functions for the variable Block-Gate Distance. Image taken from Matlab

Block-Gate Distance			
Fuzzy Set	**Min**	**Med**	**Max**
Very Low	100	100	150
Low	100	150	200
Medium	150	200	250
High	200	250	300
Very High	250	250	300

Table 5.13 Parameters of the membership functions for the variable Block-Gate Distance

2. **Block Utilisation.** Block Utilisation is the share of capacity of the block that is currently occupied. It is calculated in real time as the ratio between the number of occupied slots in the block and the total number of slots of such block where a slot is nothing other than room for one container:

$$Block\ Utilisation = \frac{Occupied\ Slots}{Total\ number\ of\ slots\ in\ the\ block} = \frac{Occupied\ slots}{Occupied\ slots + Available\ slots} \quad (5.9)$$

Three fuzzy sets describe Block Utilisation in linguistic terms: Low, Medium and High. All of the three are described through a triangular membership function. The sets are described in Figure 5.14 and the parameters are listed in Table 5.14.

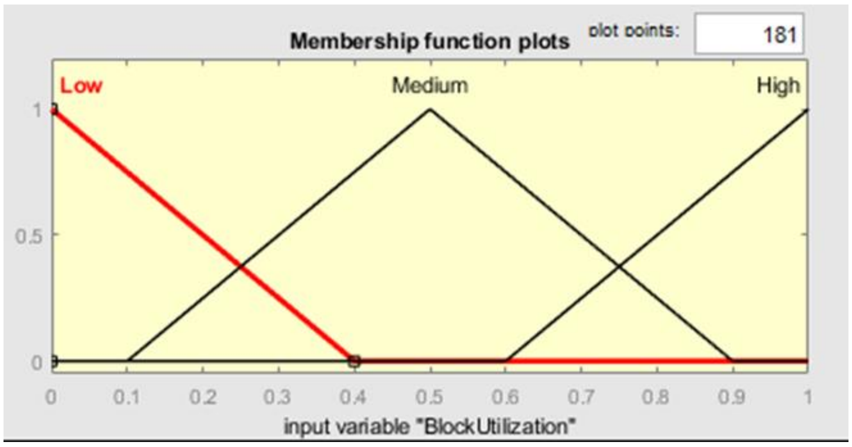


Figure 5.14 Illustration of the fuzzy sets and their membership functions for the variable Block Utilisation. Image taken from Matlab

Block Utilisation			
Fuzzy Set	**Min**	**Med**	**Max**
Low	0	0	0.4
Medium	0.1	0.5	0.9
High	0.6	1	1

Table 5.14 Parameters of the membership functions for the variable Block Utilisation

3. **Congestion.** It is defined as the queue of containers, transported by internal trucks, that are directed towards a given block at a certain point in time (when the incoming container is unloaded from the vessel). A container is considered part of the queue from the moment in which it is assigned to the considered block to the instant in which it effectively stacked in that block: this means that even during the unloading operations from the internal truck and the successive handling operations by the reach-stacker, a container is still part of the queue because it is effectively creating congestion at the block, obliging other containers bound to the same block to wait for its allocation.

   Three fuzzy sets describe Congestion the possible levels of congestion in linguistic terms: Low, Medium and High. Low is described by a triangular function while Medium and High by a trapezoidal membership function, defined by (5.10). The membership functions are illustrated in Figure 5.15 and the parameters are listed in Table 5.15. The dimension of the universe of discourse, that stretches from 0 to 20, was tuned after the first rounds of training, where the maximum levels of congestion reached peaks of 13 containers in queue.

$$f_{SET}(x, Min, Med, Max) = \begin{cases} 0 & x < Min \\ \dfrac{x - Min}{Med1 - Min} & Min \le x \le Med1 \\ 1 & Med1 \le x \le Med2 \\ \dfrac{Max - x}{Max - Med2} & Med2 < x \le Max \\ 0 & x > Max \end{cases} \qquad (5.10)$$

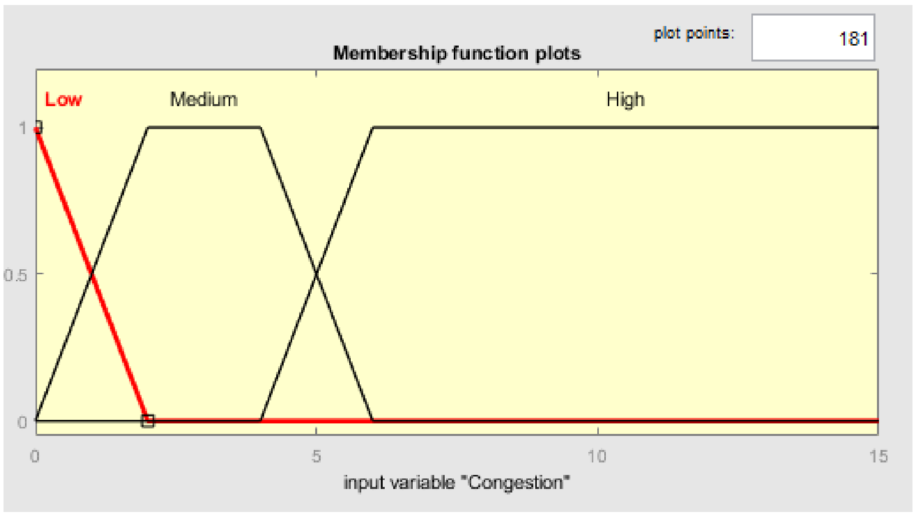


Figure 5.15 Illustration of the fuzzy sets and their membership functions for the variable Congestion. Image taken from Matlab

Congestion				
Fuzzy Set	**Min**	**Med**		**Max**
Low	0	0		2
Fuzzy Set	**Min**	**Med1**	**Med2**	**Max**
Medium	0	2	4	6
High	4	6	20	20

Table 5.15 Parameters of the membership functions for the variable Congestion

5.4.1.2 Output variables

As mentioned before, for Block Assignment there is only one output variable, represented by Block VoG.

1. **Value of Goodness of the Block (Block VoG).** Block VoG has already been described in Section 4.3. It can be described in linguistic terms with five different

fuzzy sets: Very Low, Low, Medium, High and Very High. The choice of creating five fuzzy sets came with the intention of allowing the Inference Process, and the creation of the rules, to be more flexible in comparison with an output variable with only three fuzzy sets. Those sets are represented in Figure 5.16 and their parameters are listed in Table 5.16.

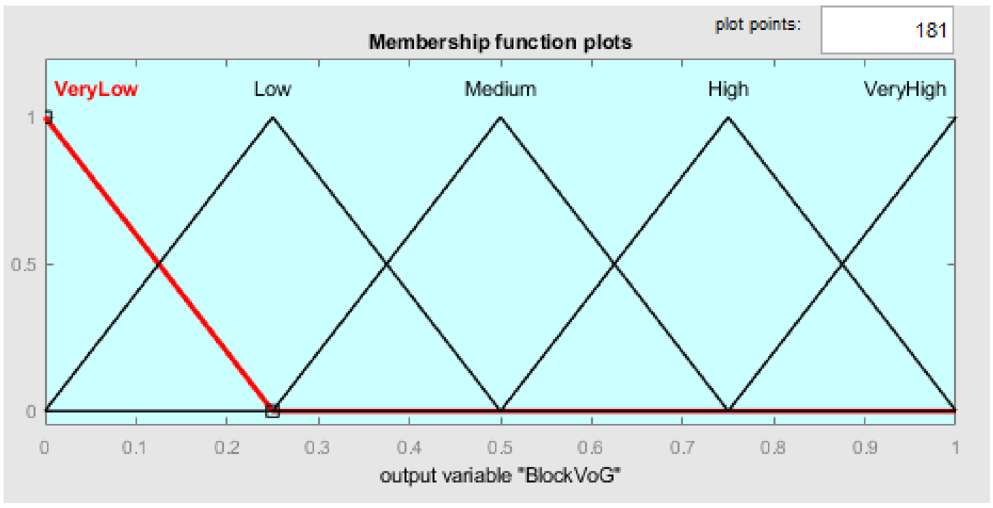


Figure 5.16 Illustration of the fuzzy sets and their membership functions for the output variable Block VoG. Image taken from Matlab

Block VoG			
Fuzzy Set	**Min**	**Med**	**Max**
Very Low	0	0	0.25
Low	0	0.25	0.5
Medium	0.25	0.5	0.75
High	0.5	0.75	1
Very High	0.75	1	1

Table 5.16 Parameters of the membership functions for the output variable Block VoG

5.4.1.3 Combinations of Criteria and Nomenclature

The criteria presented above do not necessarily need to be applied together. As a matter of fact, they were only listed together because they are all the Block Assignment-specific criteria that it was possible to build given the available data. This means that each possible combination of criteria is actually suitable to be an input for a Fuzzy Inference Process. In order to refer to these combinations more freely and directly, a very simple nomenclature or coding has been proposed. Each criterion is assigned a number:

- 1 for Block-Gate Distance
- 2 for Block Utilisation
- 3 for Congestion

Each combination is coded with the letter B, indicating that it is a combination of criteria for Block Assignment, and the numbers associated with the criteria that compose such combination. For example:

B13

Indicates a combination of criteria for Block Assignment (B), that employs Block-Gate Distance (1) and Congestion (3). The 7 possible combinations are then listed in Table 5.17.

Block-Gate Distance	**Block Utilisation**	**Congestion**	**Input Variables for the Inference Process**	**Code**
✗			Block-Gate Distance	B1
	✗		Block Utilisation	B2
		✗	Congestion	B3
✗	✗		Block-Gate Distance, Block Utilisation	B12
	✗	✗	Block Utilisation, Congestion	B23
✗		✗	Block-Gate Distance, Congestion	B13
✗	✗	✗	Block-Gate Distance, Block Utilisation, Congestion	B123

Table 5.17 A collection of all the combinations of criteria for Block Assignment with their codes listed in the last column

Therefore, each one of those combinations represents a different Block Assignment policy since it relies on different criteria. This means that 7 different Fuzzy Inference Processes have to be designed, one for each policy: the input variables of each Inference Process coincide with the criteria that constitute the respective policy, while the output variable is always the same, Block VoG. This is highlighted Figure 5.17 and 5.18. Having 7 different Inference Processes means having 7 different Rule bases, which are detailed in the next section.

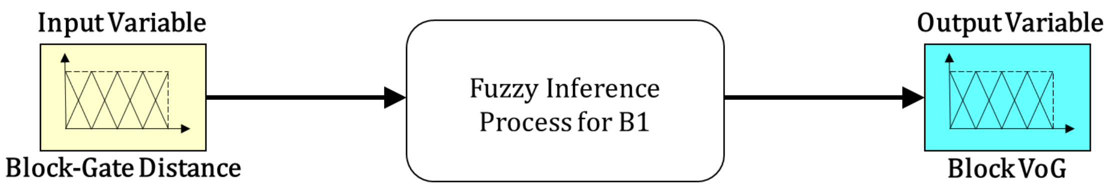


Figure 5.17 Illustration of the input and output variables for B1

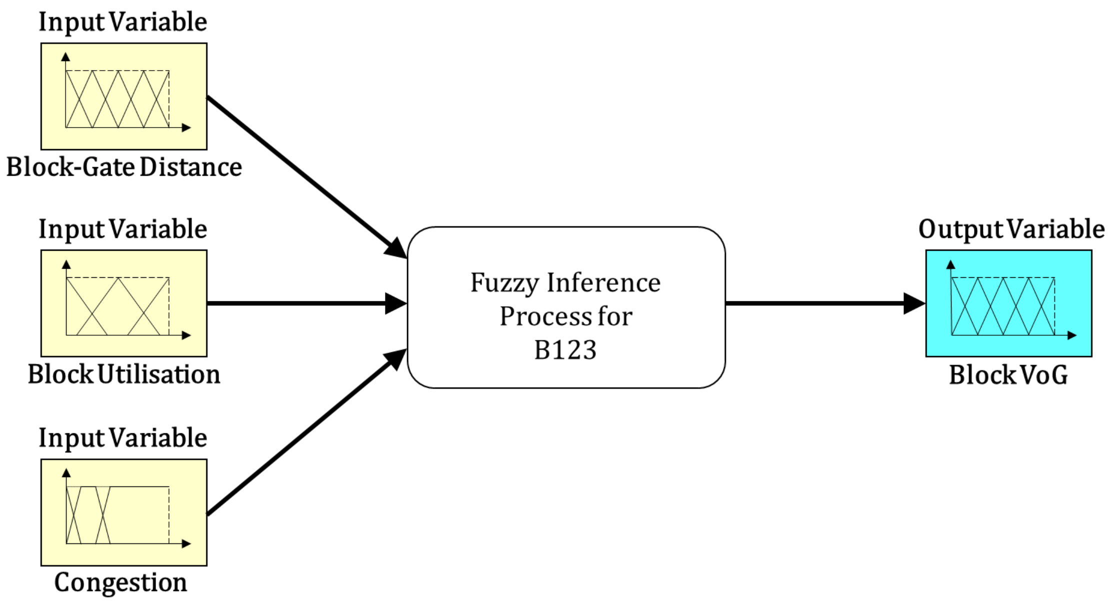


Figure 5.18 Illustration of the input and output variables for B123

5.4.1.4 The Rule Base

Each one of the 7 different policies has its own rule base, according to the specific criteria it uses. In order to maintain a certain consistency across the 7 rule bases, given also that the output variable, and the fuzzy sets that describe it, are always the same, they were compiled following this strategy:

1) The rule base for the most complete policy, B123, is defined first since it comprises all the possible combinations of the values of the fuzzy sets in the rule statements.
2) The rule bases for the policies that employ two of the three available criteria are derived from the complete rule base for B123. All the rules where the missing criterion has a value which is different from its lowest possible are eliminated. Within the remaining rules, the missing criterion is erased from the if-then statements. What is left after this procedure are several rules where the value of the output variable, Block VoG, is the same of the rules for B123 where the missing criterion is at its lowest. This corresponds to selecting the rules where that criterion has the lowest impact on the output variable. An example of this is provided by the definition of the rules for B13 policy. The missing criterion is Block Utilisation. Therefore, all the rules for B123 where Block Utilisation is not "Low" are eliminated. Then, Block Utilisation is erased from the fuzzy statements. It is now possible to compare the remaining rules to the original ones for B123; let's see, for example, Rule 2 for B123

and Rule 2 for B13: they both have the same value of Block VoG, "Very High" and generally, the fuzzy statement for Rule 2 for B13 is exactly the same that for Rule 2 for B123, except for the absence of Block Utilisation.

3) Starting from the rules obtained from the procedure described in the previous point, a more refined tuning is performed: in some rules the value of the output variable Block VoG is modified in order to fit better the combination of criteria that constitute the policy.
4) Finally, the rules for the policies where there is only input variable/criteria are created ex-novo, without any reference to the original complete rule base for B123.

The rule bases of the 7 Block Assignment policies are listed below:

1. Rule base for B123

B123 is the most differentiated Block Assignment policy since it comprises all the available criteria. Combining all the 5 fuzzy sets for Block-Gate Distance, the 3 sets for Block Utilisation and the 3 sets for Congestion means having 45 rules. Those rules are presented in a table format in Table 5.18. Each line represents a rule; the first column contains the number of the rules, the second, third and fourth columns contain the values of the input variables; the fifth column shows the values assigned by each rule to the output variable and the last column records the weight of each rule. In all the rules, the fuzzy operator is always AND, implemented through the function *min.*

	Antecedent			**Consequent**	
Rule Number	**Block-Gate Distance**	**Block Utilisation**	**Congestion**	**Block VoG**	**Rule Weight**
1	Very Low	Low	Low	Very High	1
2	Very Low	Low	Medium	Very High	1
3	Very Low	Low	High	High	1
4	Very Low	Medium	Low	Very High	1
5	Very Low	Medium	Medium	Very High	1
6	Very Low	Medium	High	High	1
7	Very Low	High	Low	High	1
8	Very Low	High	Medium	High	1
9	Very Low	High	High	Medium	1
10	Low	Low	Low	Very High	1
11	Low	Low	Medium	Very High	1
12	Low	Low	High	Medium	1
13	Low	Medium	Low	Very High	1
14	Low	Medium	Medium	High	1

Rule Number	Block-Gate Distance	Block Utilisation	Congestion	Block VoG	Rule Weight
15	Low	Medium	High	Medium	1
16	Low	High	Low	High	1
17	Low	High	Medium	Medium	1
18	Low	High	High	Low	1
19	Medium	Low	Low	High	1
20	Medium	Low	Medium	Medium	1
21	Medium	Low	High	Low	1
22	Medium	Medium	Low	High	1
23	Medium	Medium	Medium	Medium	1
24	Medium	Medium	High	Low	1
25	Medium	High	Low	Medium	1
26	Medium	High	Medium	Low	1
27	Medium	High	High	Very Low	1
28	High	Low	Low	Medium	1
29	High	Low	Medium	Medium	1
30	High	Low	High	Low	1
31	High	Medium	Low	Medium	1
32	High	Medium	Medium	Low	1
33	High	Medium	High	Very Low	1
34	High	High	Low	Low	1
35	High	High	Medium	Very Low	1
36	High	High	High	Very Low	1
37	Very High	Low	Low	Medium	1
38	Very High	Low	Medium	Low	1
39	Very High	Low	High	Very Low	1
40	Very High	Medium	Low	Medium	1
41	Very High	Medium	Medium	Low	1
42	Very High	Medium	High	Very Low	1
43	Very High	High	Low	Very Low	1
44	Very High	High	Medium	Very Low	1
45	Very High	High	High	Very Low	1

Table 5.18 Rule base for B123

2. Rule Base for B13

The rule base for B13 is presented in a table format in Table 5.19. In all the rules, the fuzzy operator is always AND, implemented through the function *min.*

	Antecedent		Consequent	
Rule Number	**Block-Gate Distance**	**Congestion**	**Block VoG**	**Rule Weight**
1	Very Low	Low	Very High	1
2	Very Low	Medium	Very High	1
3	Very Low	High	High	1
4	Low	Low	Very High	1
5	Low	Medium	High	1
6	Low	High	Medium	1
7	Medium	Low	High	1
8	Medium	Medium	Medium	1
9	Medium	High	Low	1
10	High	Low	Medium	1
11	High	Medium	Medium	1
12	High	High	Very Low	1
13	Very High	Low	Medium	1
14	Very High	Medium	Low	1
15	Very High	High	Very Low	1

Table 5.19 Rule base for B13

3. Rule Base for B23

The rule base for B23 is presented in a table format in Table 5.20. In all the rules, the fuzzy operator is always AND, implemented through the function *min*.

	Antecedent		Consequent	
Rule Number	**Block Utilisation**	**Congestion**	**Block VoG**	**Rule Weight**
1	Low	Low	Very High	1
2	Low	Medium	Very High	1
3	Low	High	Medium	1
4	Medium	Low	Very High	1
5	Medium	Medium	High	1
6	Medium	High	Medium	1
7	High	Low	High	1
8	High	Medium	Medium	1
9	High	High	Low	1

Table 5.20 Rule base for B23

4. Rule Base for B12

The rule base for B12 is presented in a table format in Table 5.21. In all the rules, the fuzzy operator is always AND, implemented through the function *min.*

	Antecedent		Consequent	
Rule Number	Block-Gate Distance	Block Utilisation	Block VoG	Rule Weight
1	Very Low	Low	Very High	1
2	Very Low	Medium	Very High	1
3	Very Low	High	High	1
4	Low	Low	Very High	1
5	Low	Medium	Very High	1
6	Low	High	High	1
7	Medium	Low	High	1
8	Medium	Medium	High	1
9	Medium	High	Medium	1
10	High	Low	Medium	1
11	High	Medium	Medium	1
12	High	High	Low	1
13	Very High	Low	Medium	1
14	Very High	Medium	Very Low	1
15	Very High	High	Very Low	1

Table 5.21 Rule base for B12

5. Rule Base for B1

The rule base for B1 is presented in a table format in Table 5.22. In this case, no fuzzy operators are required since there is only one input variable.

	Antecedent	Consequent	
Rule Number	Block-Gate Distance	Block VoG	Rule Weight
1	Very Low	Very High	1
2	Low	High	1
3	Medium	Medium	1
4	High	Low	1
5	Very High	Very Low	1

Table 5.22 Rule base for B1

These rules were defined independently from the complete rule base for B123 and the task was helped by the fact that both the input and the output variable are described by the same number of fuzzy sets. The general meaning of this combination of rules is that it is preferable to stack an incoming container close to the gate. In this way, the external trucks that will retrieve the containers from the block at the end of the dwell time, will travel less inside the yard, speeding up the retrieval operations and creating less bottlenecks.

6. Rule Base for B2

The rule base for B2 is presented in a table format in Table 5.23. In this case, no fuzzy operators are required since there is only one input variable.

	Antecedent	Consequent	
Rule Number	Block Utilisation	Block VoG	Rule Weight
1	Low	Very High	1
2	Low	High	1
3	Medium	Medium	1
4	High	Low	1
5	High	Very Low	1

Table 5.23 Rule base for B2

An observation is needed regarding the layout of these rules. There are two couples of rules (1 and 2, 4 and 5), where the value of the input variable is the same, but the corresponding value of the output variable is different. This might look like a contradiction but it was done for operational purposes, in order to give more flexibility to the Inference Process. In fact, adding a fuzzy set at the extremes of the universe of discourse (Very High or Very Low) allows to have a wider aggregated fuzzy set at the end of the aggregation process: in this way, the range of values that can be assumed by the centroid of the aggregated set is enlarged. The layout of the rules can also be supported on a logical level if High and Very High (in the same way as Low and Very Low) are considered a unique fuzzy set, assigned to the output variable when the input is Low (and, respectively, High).

7. Rule Base for B3

The rule base for B3 is presented in a table format in Table 5.24. In this case, no fuzzy operators are required since there is only one input variable.

	Antecedent	Consequent	
Rule Number	Congestion	Block VoG	Rule Weight
1	Low	Very High	1
2	Low	High	1
3	Medium	Medium	1
4	High	Low	1
5	High	Very Low	1

Table 5.24 Rule base for B3

The rationale behind the design of these rules is the same that was applied for the rule base of the policy B2.

5.4.1.5 The Summary Table for Block Assignment

A table has been created in order to collect all the information and elements of the various versions of the Fuzzy Inference Process through which the 7 different policies for Block Assignment are implemented. This table, Table 5.25, is presented below.

Code	Input Variables	Rule Base (Table)	Fuzzy Operators	Implication Method	Aggregation Method	Defuzzification	Output Variable
B1	Block-Gate Distance	5.22	-	Truncation	*max*	Centroid	Block VoG
B2	Block Utilisation	5.23	-	Truncation	*max*	Centroid	Block VoG
B3	Congestion	5.24	-	Truncation	*max*	Centroid	Block VoG
B12	Block-Gate Distance, Block Utilisation	5.21	AND (*min*)	Truncation	*max*	Centroid	Block VoG
B23	Block Utilisation, Congestion	5.20	AND (*min*)	Truncation	*max*	Centroid	Block VoG
B13	Block-gate Distance, Congestion	5.19	AND (*min*)	Truncation	*max*	Centroid	Block VoG
B123	Block-gate Distance, Block Utilisation, Congestion	5.18	AND (*min*)	Truncation	*max*	Centroid	Block VoG

Table 5.25 Summary table for Block Assignment

Each line represents a stacking policy. The first column contains the codes of the 7 policies; the second column reports the input variables; the third column includes a reference to the rule base by listing the table where it is listed in detail; the fourth column indicates which fuzzy operators are used to compile the fuzzy statements and to combine the different input variables; the fifth column lists which methods are used for the implication process; the sixth column indicates the functions that are used for the aggregation procedure; the seventh column reports the method of defuzzification and, finally, the eight column contains the output variable.

5.4.2 Stack Assignment

5.4.2.1 Input Variables/Criteria

There are two input variables for Stack Assignment, Rehandles and Stack Height:

1. **Rehandles.** Rehandles/Reshuffles are defined as the number of unproductive moves needed to retrieve a container from a given stack. But what are unproductive moves? When a container needs to be retrieved from a stack, an unproductive move is the movement required to move out of the way a container placed on top of the outgoing one. On the other hand, an efficient move is the "successful" retrieval movement performed on the outgoing container. Once the required container is retrieved, the containers that needed to be moved are relocated back in the stack in the same order as they were before the retrieval operation: as stated by Guerra-Olivares et al. (2017), this is a common assumption while dealing with reshuffles and the container allocation problem.

   Since rehandles happen during retrieval operations, in order to use them as a criterion it is fundamental to know in advance the dwell time of the containers. As mentioned in Chapter 4, Maldonado et al. (2019) developed a technique to predict the dwell time of import containers at the Port of Arica. Since this information is present in the available data, it is used to estimate the number of rehandles caused by each arriving container. In fact, a more proper name for this criterion should be "Potential Rehandles": each stack is evaluated as a potential allocation option by calculating the number of rehandles that an incoming container is likely to cause if positioned on top of the considered stack. The number of reshuffles is calculated following the retrieval order given by the predicted dwell times and considering to empty the stack, without any arrival of other containers.

   An example of how potential rehandles are calculated is shown in Figure 5.19.

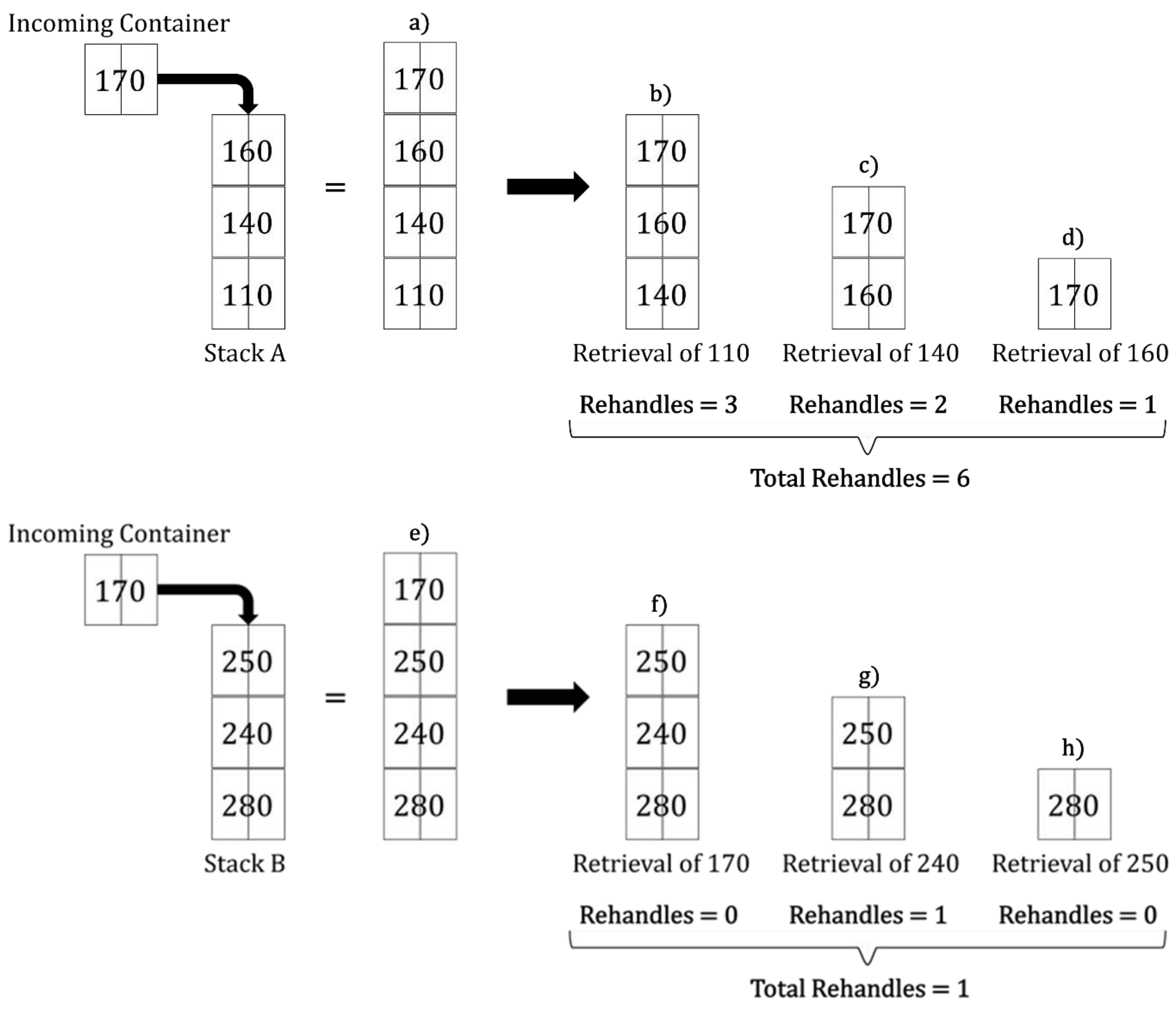


Figure 5.19 Illustration of how potential rehandles are calculated

The incoming container, with a predicted dwell time of 170, has two options for allocation: Stack A and Stack B. Stack A is evaluated first: it is supposed that the incoming container is stacked on top of it (Figure 5.19a). The numbers on the containers are their dwell times. Therefore, the retrieval order is 110, 140, 160, 170. The first container to leave is the one with a dwell time of 110, which is below other 3 containers: three rehandles are needed to retrieve it. The stack after the first retrieval is shown in Figure 5.19b. The second container to leave is 140 (for the sake of simplicity, in this example the container is named after its dwell time). Two containers are on top of it, so two rehandles are required. The stack after this retrieval is pictured in Figure 5.19c. Container 160 is the next one to be retrieved and it is located below Container 170, so another rehandle is needed. Finally, only Container 170 is left (Figure 5.19d) and, since there are no containers blocking its way, is retrieved with no additional rehandles. Once the stack is empty, the total number of rehandles caused by the allocation of container 170 on top of Stack A is calculated as 3 + 2 + 1 = 6. This represents the worst possible condition for a four-tier high stack, since the retrieving order is the exact opposite of the stacking order of the containers. Stack B is evaluated in the same way: Container 170 is virtually allocated on top of it (Figure 5.19e). The retrieval order is 170, 240, 250, 280. The first container to leave the stack is the incoming one and, since it is not blocked by other containers, does not generate rehandles. The resulting stack is shown in Figure 5.19f. Container 240 is the second in line to be retrieved, causing 1 rehandle. After the retrieval of Container 240, the configuration of the stack is represented by Figure 5.19g. The last two remaining containers, 250 and 280, leave the stack in their stacking order, without additional rehandles. Now that the stack is empty, the total number of rehandles caused by the allocation of container 170 on top of Stack B is calculated as 0 + 1 + 0 = 1. Therefore, Stack B appears to be a better choice for stacking the incoming container 170, since the number of potential rehandles is lower. A higher number of rehandles, in fact, results in higher costs and longer service times for the container terminal.

This approach of using the dwell times to evaluate the number of potential rehandles seems like a novelty in the existing literature or, at least, amongst the examined works. In the vast majority of them, in fact, the focus was only on the dwell time (or the relative category) of the incoming container and the container at the top of the stack, without considering the remaining containers at the base of the stack. Only Borgman et al. (2010) used dwell times, segregated into categories, to estimate the probability of rehandling by comparing the earlier category in the whole stack with the category of the incoming container.

Out of the 9 blocks dedicated to import containers, 7 of them have a maximum stack height of 4 tiers and the remaining 2 have a maximum height of 5 tiers. The maximum stack height influences the maximum number of potential relocations, which is one of the extremes of the universe of discourse for the input variable Rehandles. For this reason, two different group of fuzzy sets were created: one is dedicated to the blocks with stacks of maximum 4 tiers and the other one to the blocks with stacks of 5 tiers.

For 4-tier stacks, three fuzzy sets describe Rehandles in linguistic terms: Low, Medium, High. They are represented in Figure 5.20. All of them are described by a triangular membership function defined by 5.6 and whose parameters are reported in Table 5.26.

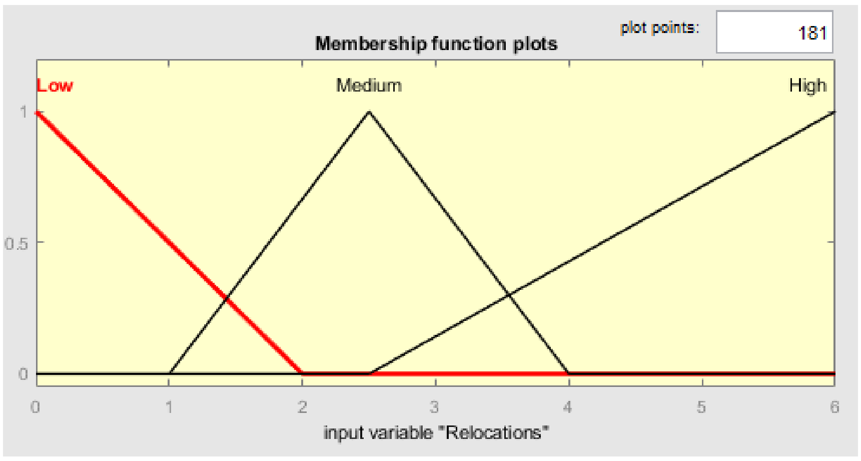


Figure 5.20 Illustration of the fuzzy sets and their membership functions for the variable Rehandles for a 4-tier stack. Image taken from Matlab

Rehandles			
Fuzzy Set	**Min**	**Med**	**Max**
Low	0	0	2
Medium	1	2.5	4
High	2.5	6	6

Table 5.26 Parameters of the membership functions for the variable Rehandles for a 4-tier stack.

For a 5-tier stack, four fuzzy sets are associated with Rehandles: Low, Medium, High and Very High. They are represented in Figure 5.21. The first three of them are described by a triangular membership function defined by 5.6, while Very High is described by a trapezoidal function such as 5.10. The parameters of the functions are listed in Table 5.27.

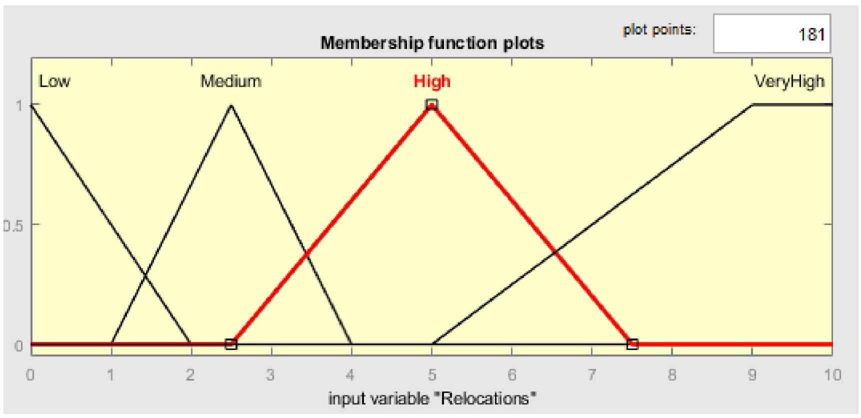


Figure 5.21 Illustration of the fuzzy sets and their membership functions for the variable Rehandles for a 5-tier stack. Image taken from Matlab

Rehandles				
Fuzzy Set	**Min**	**Med**		**Max**
Low	0	0		2
Medium	1	2.5		4
High	2.5	5		7.5
Fuzzy Set	**Min**	**Med1**	**Med2**	**Max**
Very High	5	9	10	10

Figure 5.27 Parameters of the membership functions for the variable Rehandles for a 5-tier stack

2. **Stack Height.** Stack Height is defined as the current height of a stack, where the height is measured in the number of containers that make up the pile.

   Again, as there are blocks with a different maximum number of tiers, which affects the extension of the universe of discourse, two different group of fuzzy sets for the variable Stack Height were created: one is dedicated to the blocks with stacks of maximum 4 tiers and the other one to the blocks with stacks of 5 tiers.

   For 4-tier stacks, three fuzzy sets describe Stack Height in linguistic terms: Low, Medium, High. They are represented in Figure 5.22. All of them are described by a triangular membership function defined by 5.6 and whose parameters are reported in Table 5.28.

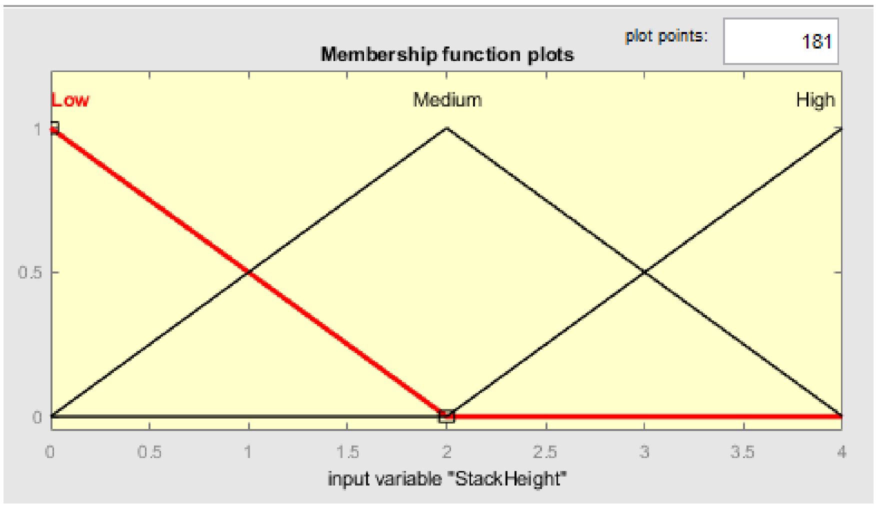


Figure 5.22 Illustration of the fuzzy sets and their membership functions for the variable Stack Height for a 4-tier stack. Image taken from Matlab

Stack Height			
Fuzzy Set	**Min**	**Med**	**Max**
Low	0	0	2
Medium	0	2	4
High	2	4	4

Table 5.28 Parameters of the membership functions for the variable Stack Height for a 4-tier stack.

For a 5-tier stack, three fuzzy sets are associated with Stack Height: Low, Medium, High. They are represented in Figure 5.23. The first two of them are described by a triangular membership function defined by 5.6, while High is described by a trapezoidal function such as 5.10. The parameters of the functions are listed in Table 5.29.

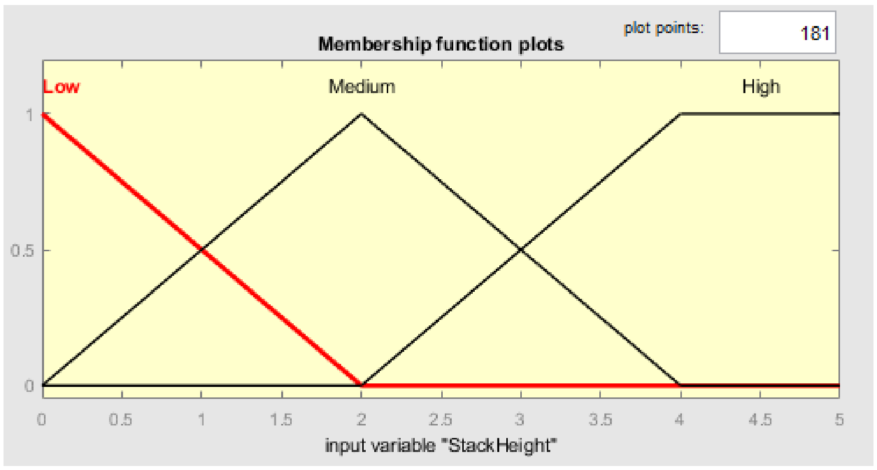


Figure 5.23 Illustration of the fuzzy sets and their membership functions for the variable Stack Height for a 5-tier stack. Image taken from Matlab

Stack Height				
Fuzzy Set	**Min**	**Med**		**Max**
Low	0	0		2
Medium	0	2		4
Fuzzy Set	**Min**	**Med1**	**Med2**	**Max**
High	2	4	5	5

Table 5.28 Parameters of the membership functions for the variable Stack Height for a 5-tier stack.

5.4.2.2 Output variables

With regards to Stack Assignment, there is only one output variable: Stack VoG.

1. **Value of Goodness of the Stack (Stack VoG).** Stack VoG has already been described in Section 4.3.

   For a stack with 4 tiers, the output variable Stack VoG is described by five fuzzy sets: Very Low, Low, Medium, High, Very High. The shape of these sets corresponds exactly to the sets of the same name that were created for Block VoG. All of them are described by a triangular membership function, whose equation is 5.6. They are represented in Figure 5.24 and the parameters are collected in Table 5.29.

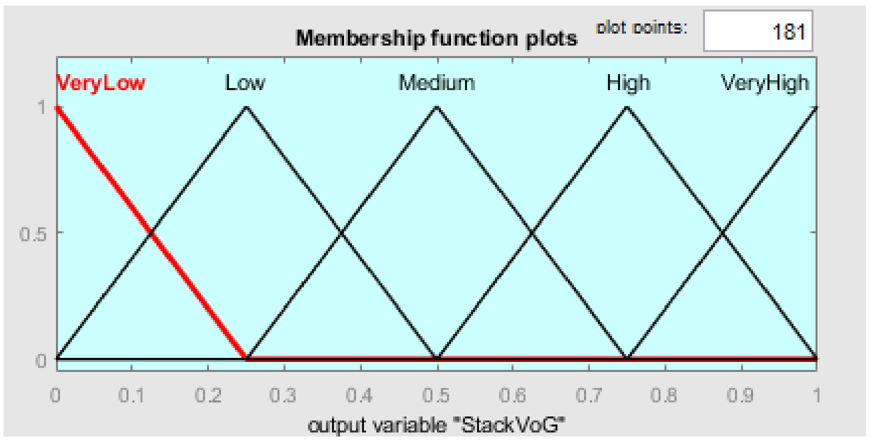


Figure 5.24 Illustration of the fuzzy sets and their membership functions for the output variable Stack VoG for a 4-tier stack. Image taken from Matlab

Stack VoG			
Fuzzy Set	**Min**	**Med**	**Max**
Very Low	0	0	0.25
Low	0	0.25	0.5
Medium	0.25	0.5	0.75
High	0.5	0.75	1
Very High	0.75	1	1

Table 5.29 Parameters of the membership functions for the output variable Stack VoG for a 4-tier stack.

For a 5-tier stack, an additional fuzzy set has been added: Very Very Low. This was done in order to increase the sensitivity to a large number of rehandles (up to 10), which cannot be reached in a 4-tier stack. The additional set is again described by a triangular membership function. The sets for a 5-tier stack are shown in Figure 5.25 and their parameters are reported in Table 5.30.

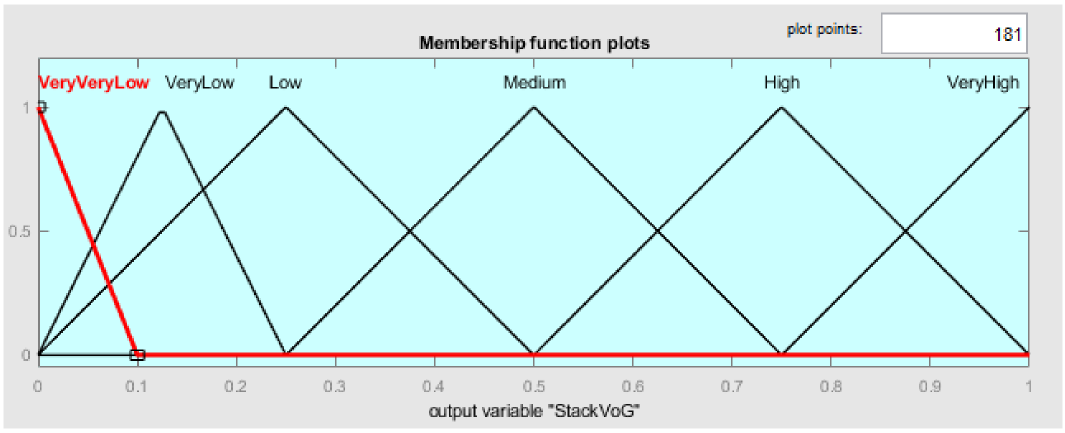


Figure 5.25 Illustration of the fuzzy sets and their membership functions for the output variable Stack VoG for a 5-tier stack. Image taken from Matlab

Stack VoG			
Fuzzy Set	**Min**	**Med**	**Max**
Very Very Low	0	0	0.1
Very Low	0	0.125	0.25
Low	0	0.25	0.5
Medium	0.25	0.5	0.75
High	0.5	0.75	1
Very High	0.75	1	1

Table 5.30 Parameters of the membership functions for the output variable Stack VoG for a 5-tier stack.

5.4.2.3 Combinations of Criteria and Nomenclature

As stated for Block Assignment, the two criteria presented above do not need to be applied together but can also be used alone as inputs for a Fuzzy Inference Process. In this case too, criteria and their combination are coded so to address them more directly. Each criterion is assigned a number:

- 1 for Block-Gate Distance
- 2 for Block Utilisation

The coding starts with the letter S, indicating Stack Assignment, and is followed by the numbers associated with the criteria. There are three possible combinations of criteria and are listed in Table 5.31.

Rehandles	Stack Height	Input Variables for the Inference Process	Code
✗		Rehandles	S1
	✗	Stack Height	S2
✗	✗	Rehandles, Stack Height	S12

Table 5.31 A collection of all the combinations of criteria for Stack Assignment with their codes listed in the last column

Therefore, there are 3 different Stack Assignment policies, which offer three possible different inputs for the Fuzzy Inference Process.

5.4.2.4 The Rule Base

The 3 combinations of input variables are mapped to the output variable Stack VoG by defining a rule base. Each one of the 3 different policies has its own rule base, according to the specific criteria it uses. Moreover, a distinction is made between the rule base for a 4-tier stack and a 5-tier stack. In this case, given that 2 out of the 3 policies apply only one criterion as an input variable, the rule bases were compiled without a particular strategy.

1a. Rule Base for S1 (4-tier stack)

The rule base for S1 for a 4-tier stack is presented in a table format in Table 5.32. In this case, no fuzzy operators are required since there is only one input variable.

	Antecedent	Consequent	
Rule Number	Rehandles	Stack VoG	Rule Weight
1	Low	Very High	1
2	Low	High	1
3	Medium	Medium	1
4	High	Low	1
5	High	Very Low	1

Table 5.32 Rule base for S1 (4-tier stack)

The apparent contradiction of having two rules with the same value of the input variable and two different outcomes for the output variable is explained in the same way as for B2 and B3.

1b. Rule Base for S1 (5-tier stack)

The rule base for S1 for a 5-tier stack is presented in a table format in Table 5.33. In this case, no fuzzy operators are required since there is only one input variable.

	Antecedent	Consequent	
Rule Number	Rehandles	Stack VoG	Rule Weight
1	Low	Very High	1
2	Low	High	1
3	Medium	Medium	1
4	High	Low	1
5	High	Very Low	1
6	Very High	Very Very Low	1

Table 5.33 Rules base for S1 (5-tier stack)

2a. Rule Base for S2 (4-tier stack)

The rule base for S2 for a 4-tier stack is presented in a table format in Table 5.34. In this case, no fuzzy operators are required since there is only one input variable.

	Antecedent	Consequent	
Rule Number	Stack Height	Stack VoG	Rule Weight
1	Low	Very High	1
2	Low	High	1
3	Medium	Medium	1
4	High	Low	1
5	High	Very Low	1

Table 5.34 Rule base for S2 (4-tier stack)

2b. Rule Base for S2 (5-tier stack)

The rule base for S2 for a 5-tier stack is presented in a table format in Table 5.35. In this case, no fuzzy operators are required since there is only one input variable.

	Antecedent	Consequent	
Rule Number	Stack Height	Stack VoG	Rule Weight
1	Low	Very High	1
2	Low	High	1
3	Medium	Medium	1
4	High	Low	1
5	High	Very Low	1
6	High	Very Very Low	1

Table 5.35 Rules base for S2 (5-tier stack)

3a. Rule Base for S12 (4-tier stack)

The rule base for S12 for a 4-tier stack is presented in a table format in Table 5.36. In all the rules, the fuzzy operator is always AND, implemented through the function *min.*

	Antecedent		**Consequent**	
Rule Number	**Rehandles**	**Stack Height**	**Stack VoG**	**Rule Weight**
1	Low	Low	Very High	1
2	Low	Medium	Very High	1
3	Low	High	Very High	1
4	Medium	Low	Medium	1
5	Medium	Medium	Medium	1
6	Medium	High	High	1
7	High	Low	Low	1
8	High	Medium	Very Low	1
9	High	High	Very Low	1

Table 5.36 Rule base for S12 (4-tier stack)

There are two Rules that might look counterintuitive at a first glance: Rule 3 and 6. The reason why those rules were implemented as such is the following: it is exactly when the stack is high that having few Rehandles becomes important because a higher stack means a higher number of potential relocations. Therefore, those Rules prioritise the stacks that, albeit being high, allow only a low number of Rehandles to happen.

Moreover, some of the rules were compiled even if they will not be triggered in any situation: having a "High" number of Rehandles with a Stack Height that is "Low" is practically impossible because there are not enough containers to generate so many relocations.

3b. Rule Base for S12 (5-tier stack)

The rule base for S12 for a 5-tier stack is presented in a table format in Table 5.37. In all the rules, the fuzzy operator is always AND, implemented through the function *min.* This rule base makes use of the additional fuzzy sets "Very High" for the input variable Rehandles and "Very Very Low" for the output variable Stack VoG.

	Antecedent		Consequent	
Rule Number	**Rehandles**	**Stack Height**	**Stack VoG**	**Rule Weight**
1	Low	Low	Very High	1
2	Low	Medium	Very High	1
3	Low	High	Very High	1
4	Medium	Low	Medium	1
5	Medium	Medium	Medium	1
6	Medium	High	High	1
7	High	Low	Low	1
8	High	Medium	Very Low	1
9	High	High	Very Low	1
10	Very High	Low	Very Low	1
11	Very High	Medium	Very Very Low	1
12	Very High	High	Very Very Low	1

Table 5.37 Rule base for S12 (5-tier stack)

5.4.2.5 The Summary Table for Stack Assignment

A table has been created in order to collect all the information and elements of the various versions of the Fuzzy Inference Process through which the 3 different policies for Stack Assignment are implemented. This table, Table 5.38, is presented below. The structure is identical to the twin table 5.25 for Stack Assignment.

Code	**Input Variables**	**Rule Base (Table) 4-tier / 5-tier**	**Fuzzy Operators**	**Implication Method**	**Aggregation Method**	**Defuzzification**	**Output Variable**
S1	Rehandles	5.32 / 5.33	-	Truncation	*max*	Centroid	Stack VoG
S2	Stack Height	5.34 / 5.35	-	Truncation	*max*	Centroid	Stack VoG
S12	Rehandles, Stack Height	5.36 / 5.37	AND (*min*)	Truncation	*max*	Centroid	Stack VoG

Table 5.38 Summary table for Stack Assignment

5.5 The Fuzzy Systems

As shown by Tables 5.25 and 5.38 there are 7 different policies for Block Assignment and 3 policies for Stack Assignment, respectively. However, as mentioned in Section 4.3, Block and Stack Assignment are two phases of a single process: the allocation of an incoming container. In fact, Block Assignment aims at finding the best block in the yard at a given moment and Stack Assignment looks for the best stack within the best block. Therefore, the policies for Block and Stack Assignment need to be put together in order to control the allocation of an incoming container. The combination of a Block Assignment and a Stack Assignment policy is called Fuzzy System since it is a system that governs the allocation of incoming containers through fuzzy logic. There are 21 combinations of Block and Stack Assignment policies, which generate an equal number of Fuzzy Systems, as presented in table 5.39.

		Stack Assignment		
		S1	**S2**	**S12**
Block Assignment	**B1**	B1 S1	B1 S2	B1 S12
	B2	B2 S1	B2 S2	B2 S12
	B3	B3 S1	B3 S2	B3 S12
	B12	B12 S1	B12 S2	B12 S12
	B23	B23 S1	B23 S2	B23 S12
	B13	B13 S1	B13 S2	B13 S12
	B123	B123 S1	B123 S2	B123 S12

Table 5.39 A collection of all the possible Fuzzy Systems given the available criteria

Each Fuzzy System is coded with the combination of the codes of its Block and Stack Assignment policies. For example:

B23 S1

Indicates a Fuzzy System that employs Block Utilisation (2) and Congestion (3) as criteria for Block Assignment (B) and Rehandles (1) as the criterion for Stack Assignment (S).

These Fuzzy Systems are going to be tested under different circumstances, in order to understand how they react to events and disturbances. Finally, according to their performances, they are going to be used as the bricks that build up a dynamic allocation strategy.

CHAPTER 6

Modelling the Yard and the Events with Matlab

This chapter presents how the yard and the operations related to stacking and retrieval have been modelled. Matlab is the programming language where the model has been developed. After an overview of how the model works, a detailed description of the main input variables is provided. Then, the focus is shifted towards the functions employed by the model and the modelling of the events, with the aid of pseudocode and a flow chart.

6.1 Introduction

The core of the model is represented by a time loop which works like a clock, simulating the passing of time. Prior to the loop, a set of support variables, vectors and matrices are initialised. Within the loop itself, a series of functions reproduce various operations that happen in the yard: the unloading of a container from a vessel, its transportation from the berth to the assigned block, the allocation procedure on the chosen stack, the retrieval of a container at the end of its dwell time, the queue of internal trucks that are directed towards the same block and so on, including the Block and Stack Assignment decision-making process, implemented through the Fuzzy Systems that were introduced in the previous chapter. In addition to the traditional yard management procedure, a brief list of events and disturbances are also modelled with peculiar focus on their impact. All these occurrences are reproduced time unit by time unit, in order to reproduce their real-time development. At the end of the time loop, the data are then collected and presented in tables that can be read with ease, highlighting the main Performance Indicators.

6.2 The outline of the Matlab model

A representation of how the model works is shown in Figure 6.1 through a flowchart. The model starts with the definition of some variables that contain information about the layout of the container terminal at the Port of Arica: *Yard Matrix*, a matrix where the number of stacks and tiers for each block are stored and two matrices that report quay-block and block-gate distances respectively, *Block Quay Distance* and *Block Gate Distances*.

Then, a set of support variables are introduced and initialised. The most important of them is *Container Database*, which is a matrix where every possible information about a container

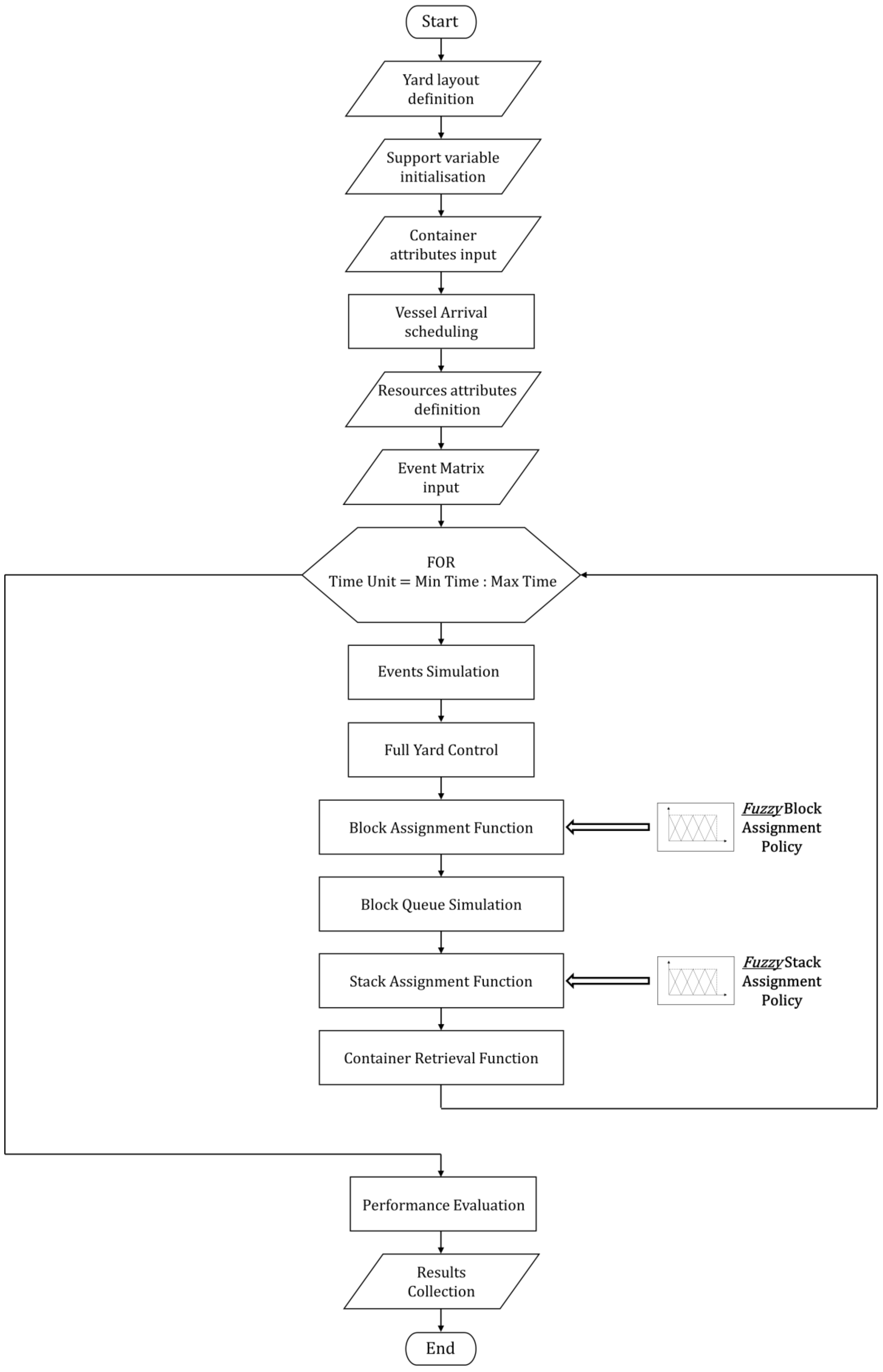

Start
Yard layout definition
Support variable initialisation
Container attributes input
Vessel Arrival scheduling
Resources attributes definition
Event Matrix input
FOR Time Unit = Min Time : Max Time
Events Simulation
Full Yard Control
Block Assignment Function
Fuzzy Block Assignment Policy
Block Queue Simulation
Stack Assignment Function
Fuzzy Stack Assignment Policy
Container Retrieval Function
Performance Evaluation
Results Collection
End

Figure 6.1 An illustration of the flowchart that shows the main steps of the model

is recorded: dwell time, position in the yard, quay of arrival, travelled quay-block distance, travelled or to be travelled block-gate distance, congestion at the block on arrival and so on.

Other important support variables are *Real Yard* and *Predicted Yard*, matrices that represent a visual representation of the real-time condition of the blocks, showing each stack for every block and which containers it stores. Fundamental are also the variables *Congestion*, *Congestion Matrix*, *Block Feasibility* and *Stack Feasibility*: *Congestion* is a vector which contains the queue of containers for each block, *Congestion Matrix* is a matrix that helps governing the travel of the containers on the internal vehicles, *Block Feasibility* is another vector that indicates which blocks are available for the stacking of an incoming container and *Stack Feasibility* is a vector defined for each block, showing the candidate stacks available for allocation.

The dwell time of the incoming containers are retrieved from an Excel file derived from the original database and are listed in a dedicated vector.

Since there is no information available about the arrival times of the containers of the database, the arrival of the vessels is decided with a very brief calculation. The resulting arrivals times, alongside the quays of arrival for the vessels, are put together in the same new matrix, called simply *Times*, with the predicted dwell times.

The needed attributes related to the internal trucks and reach stackers are defined through dedicated variables: *Normal Stacking Time* indicates how much time it takes to pick up a container from an internal truck and place it on the chosen stack by a reach stacker or another type of crane, *Vehicle Speed* is the speed at which the internal vehicles are allowed to travel in the yard, *Crane Block Association* associates the block to the crane that serves it and *Crane Relation* is a matrix which reproduces the content of Table 4.6 (it shows which resource substitutes a broken one in the case of a breakdown).

Finally, *Event Matrix* is retrieved from the Excel file where it has been previously stored. *Event Matrix* is simply a matrix that indicates when an event happens, how long does it last, and the typology of the events. After that, a series of other matrices related to the single types of the events are generated in order to simulate the occurrences.

The time loop is then ready to start: it has been modelled as a for-loop and it goes from a starting moment (*Min Time*) to a final moment (*Max Time*), time unit by time unit. Each run corresponds to a time unit and during that run all the possible occurrences (events or operations) that might happen in that instant are evaluated through specific functions.

The first thing that is done inside the loop are the calling of the functions related to single type of events: each one of those functions reproduce the impact of those events before any of the other operations. In this way, before assigning the incoming container to a block or a stack, the dwell time of the containers is modified, a block is barred from the allocation possibilities etc.

After the evaluation of the events, a control mechanism is put in place to verify whether, at a certain time unit, the yard is full and there are no more available places to stack an incoming container. If this is the case for one or more containers, they have to wait until a new slot becomes available in one of the blocks.

When the arrival time of one of the containers that belong to the matrix *Times* equals the current value of the *Time Unit*, the Block Assignment Function is called. Given the data of the container under examination and the current state of the yard, this function implements the selection process of the blocks with the Block Assignment part (B) of one the Fuzzy Systems that were introduced at the end of Chapter 5. Once one of the blocks has been selected, *Congestion Matrix* is updated with the data of the incoming container, signalling that the container is now travelling towards its block of destination.

If one of the travelling containers arrives in front of his block of destination at the current *Time Unit*, the function Block Queue Simulation is triggered. This function simulates what happens in front of a block: the incoming container might be allocated immediately or might have to wait because the reach stacker that serves the block might be busy allocating another container arrived earlier. The most important output of the function is represented by the time unit when the incoming container becomes the first of the queue and is ready to be stacked.

If the current *Time Unit* equals the instant when one of the containers in *Congestion Matrix* becomes available for the final stacking, the Stack Assignment Function is called. Considering the real time state of the stacks at the assigned block and the attributes of the incoming container, the best possible stack is chosen with the Stack Assignment phase of one of the Fuzzy Systems. Once the container is stacked, the matrix *Container Database* increases by one line containing all the data about the incoming container and *Real* and *Predicted Yard* are updated by placing said container on the top of the assigned stack.

When the current *Time Unit* corresponds to the time of departure of one of the containers in *Container Database*, the function Container Retrieval comes into action: it simulates the retrieval of the container from the stack, counting the number of required Rehandles and Efficient Moves, and updates all the concerned variables by eliminating the retrieved container.

As for the current layout of the model, a container is assigned to a block during one run of the loop (so one time unit), is assigned to a stack after some runs (due to the travelling time and the possible queues) and is effectively allocated in the chosen slot after other runs (depending on the stacking speed of the reach stackers). Finally, the container leaves the yard after as many runs of the loop as its dwell time (expressed in time units).

After the end of the time loop, the main performances are calculated and collected in tables and matrices. The main source for this performance evaluation is *Container Database*, where the main of data about the containers are stored.

6.3 Important input and support variables

This section provides a more detailed explanation of the most important support variables employed in the model.

6.3.1 Yard Matrix

Yard Matrix associates each block with a number and displays how many stacks and tiers it has. The structure of the matrix is the following:

$$Yard\ Matrix = \begin{bmatrix} 1 & 24 & 4 \\ 2 & 27 & 4 \\ \dots & \dots & \dots \end{bmatrix} \tag{6.1}$$

The first line shows that block number 1 has 24 stacks and 4 tiers, block number 2 has 27 stacks and 4 tiers and so on.

6.3.2 Distances and Quay-Block Distances

Distances is an input variable which reports the distance of each block from the gate. It is implemented with a vertical vector where the position of each element corresponds to the number associated to each block in *Yard Matrix*:

$$Block - Gate\ Distances = \begin{bmatrix} 197.12 \\ 217.27 \\ 122.10 \\ 227.99 \\ \cdots \end{bmatrix} \tag{6.2}$$

The third element of the vector reports the distance from the gate to block number 3 (122.10 m).

Block Quay Distances is a matrix which reports the distance of every block from each one of the quays of the terminal. The number of lines corresponds to the number of blocks and the number of columns to the number of quays, so the element in position (m, n) is the distance of block m from quay n. An example of the matrix is presented below:

$$Block - Quay\ Distances = \begin{bmatrix} 384.69 & 429.39 & 437.93 & 435.41 & 688.50 & 893.20 \\ 301.04 & 387.10 & 403.68 & 401.64 & 686.27 & 895.88 \\ 261.74 & 463.31 & 485.14 & 477.86 & 788.64 & 990.45 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix} \tag{6.3}$$

6.3.3. Real and Predicted Yard

Real and *Predicted Yard* are two matrices that show the real time state of every stack in each block. A *Real* and *Predicted Yard* matrices are defined for each one of the blocks in the yard. The size of both matrices, for a given block b, is $i_b \times j_b$, where i_b is the number of tiers in block b and j_b is the number of stacks in block b. Both matrices can be initialised in different ways, one of which is to simulate an empty block: each position of the matrix is filled with a big number (it has to be larger than the latest possible time of delivery) such as 1000000, representing an empty slot. As the model enters the time loop, the two matrices are filled with the time of departures of the incoming containers in the position where they are allocated: the time of departure is defined as the sum of the dwell time and the allocation time of the container, where by allocation time it is meant the time unit when the incoming container is effectively stacked in the block. *Predicted Yard* uses predicted dwell times to calculate the estimated time of departure (ETD) while *Real Yard* represents containers with their real time of departure (RTD, estimated time of departure modified as an effect of some disturbances). An example of these two matrices:

$$Real\ Yard\ b = \begin{bmatrix} 1000000 & 1000000 & 1000000 & 1000000 & \cdots \\ 1000000 & 1000000 & 645.50 & 1000000 & \cdots \\ 750.00 & 1000000 & 871.00 & 1000000 & \cdots \\ 814.00 & 1000000 & 1010.0 & 450.75 & \cdots \end{bmatrix} \tag{6.4}$$

$$Predicted\ Yard\ b = \begin{bmatrix} 1000000 & 1000000 & 1000000 & 1000000 & \cdots \\ 1000000 & 1000000 & 645.50 & 1000000 & \cdots \\ 830.00 & 1000000 & 871.00 & 1000000 & \cdots \\ 814.00 & 1000000 & 1010.0 & 500.05 & \cdots \end{bmatrix} \quad (6.5)$$

(6.4) and (6.5) represent an example of a *Predicted Yard* and *Real Yard* matrix, respectively. The first column in (6.4) represents a half-filled stack: the fourth and third tiers are empty (as signalled by the reference number 1000000) while the second and first are occupied by two containers that are expected to leave the yard at time unit 750 and 814 respectively. It is worth noting that in (6.5), where real time of departures are used, the time of departure of the container in the second tier is different from the estimated time of departure as the result of events that happened during its dwell time. The second column is entirely empty since all its elements are all 1000000, the third column stores three containers and so on.

6.3.4 Block Feasibility, Stack Feasibility and Congestion

Block Feasibility is a vector that has the same length of the number of blocks in the yard. The position of the elements of the vector corresponds to the identifying number of the blocks and their value can only be 0 or 1: 1 if the block is open and available for allocation, 0 if the block is full or excluded from the possible stacking destinations as a consequence of a block-related event.

$$Block\ Feasibility = [0\ 1\ 1\ 1\ 1\ 0\ 0\ 1\ 1] \quad (6.6)$$

The vector shows that blocks 2, 3, 4, 5, 8 and 9 are feasible as block allocation destination while blocks 1, 6 and 7 are temporarily shut down.

Stack Feasibility is another horizontal vector. If there are n blocks in the yard, n Stack Feasibility vectors are defined. The length of each one of those vectors is equal to the number of stacks of the corresponding block: each stack is therefore associated with a number. In the same way as for Block Feasibility, the elements can assume only the values 0 and 1, signalling whether the stack is full or available for allocation.

Congestion is a vector whose size is again equal to the number of blocks in the yard. Each element represents the current congestion at the block that corresponds to its position.

$$Congestion = [0\ 0\ 3\ 0\ 0\ 2\ 1\ 0\ 0] \quad (6.7)$$

The vector is showing that there is a queue of 3 containers that are directed toward block 3, 2 containers are heading to block 6 and 1 container is currently moving to block 7. All the other blocks are not affected by congestion.

6.3.5 Vessel Arrival Scheduling and the matrix Times

The predicted dwell times of the incoming containers are retrieved from an Excel file that comes from the original database developed by Maldonado et al. (2019). Those data, however, do not include the arrival times of the containers. In order to assign an arrival time to the containers, so to properly simulate the variation of the load in the yard, a very simple algorithm is used. Its pseudocode is reported below:

```
define Number of Vessel Arrivals
define Unloading Time
calculate ContainerXArrivals = round down to the nearest integer (Number of
Containers/Number of Vessel Arrivals)
define ArrivalWeights as a vector
define first element of vector IDArrival as 1
for i=2:Number of Vessel Arrivals
    Assign to element i of IDArrival the value IDArrival (i-1) + round up to the
    nearest integer (ContainersXArrival * ArrivalWeights (i-1))
end
define Time Interval between two consecutive Arrivals (Time Interval)
define IntervalWeights as a vector
define first element of vector IDArrivalTime as 0
for j=2:Number of Vessel Arrivals
    Assign to element j of IDArrivalTime the value IDArrivalTime (j-1) + round
    up (Time Interval * Interval Weights (j-1))
end
Initialise Times(2,1) as a time unit value
for f=2:Number of Containers
    for g=1:Number of Vessel Arrivals
        if f equals IDArrival(g)
            Assign to Times(2,f) the value in IDArrivalTime(g)
            Leave the for second for-loop
        else
            Assign to Times(2,f) the arrival time of the previous container +
            Unloading Time
        end
    end
end
```

This algorithm splits the number of incoming containers equally among the arriving vessels. Through the *ArrivalWeights* vector it is possible to adjust or modify the number of containers for each arrival. The first for-loop populates the *IDArrival* vector: its size corresponds to the number of vessel arrivals and each element indicates the progressive number of the first container to be unloaded after a new vessel arrival. The vessels are supposed to arrive only one at a time and separated by the same time interval, which is defined by the *Time Interval* variable. This time difference can be adjusted through the vector *IntervalWeights*, a vector that works in the same way as *ArrivalWeights.* The second for-loop populates the vector *IDArrivalTime*: it works in the same way as for *IDArrival* and each element represents the arrival time of the first container to be unloaded from an incoming vessel. Finally, the last part of the algorithm assigns an arrival time to every container that comes from the database. All the containers coming from the same ship are supposed to be unloaded in a sequential way at a constant pace, indicated by the variable *Unloading Time*. Those arrival times are listed in the second line of the matrix *Times*.

The results of this process are grouped into one matrix called *Times.* In this matrix each column represents a container while the first line indicates its predicted dwell time, the second line its arrival time, the third line the quay where it has been unloaded from the corresponding ship and the fourth line includes a variable that refers to the source (which part of the database) of the dwell time prediction. An example is shown below:

$$Times = \begin{bmatrix} 190.75 & 520.15 & 816.20 & 340.00 & \cdots \\ 3 & 5 & 7 & 9 & \cdots \\ 6 & 6 & 6 & 6 & \cdots \\ 2000 & 2000 & 2000 & 2000 & \cdots \end{bmatrix} \tag{6.8}$$

6.3.6 Attributes of the yard resources

The attributes of the yard resources are defined by four main variables. *Vehicle Speed* and *Normal Stacking Time* have already been introduced in section 6.2. *Crane Block Association* is a horizontal vector that indicates which reach-stacker serves a certain block. The size of the vector corresponds to the number of blocks in the yard and its elements are the codes of the cranes serving their respective blocks:

$$Crane\ Block\ Association = [101\ 101\ 102\ 102\ 103\ 103\ 103\ 101\ 102] \tag{6.9}$$

This vector shows that crane 101 serves blocks 1, 2 and 8, crane 102 serves blocks 2, 4 and 9 and crane 103 serve blocks 5, 6 and 7.

Crane Relations is a matrix that reproduces Table 4.7, which shows which reach-stacker substitutes are broken one. The first line indicates the codes of the cranes while the second one shows the helping crane:

$$Crane\ Relations = \begin{bmatrix} 101 & 102 & 103 \\ 102 & 101 & 102 \end{bmatrix} \tag{6.10}$$

The first column indicates that when crane 101 breaks down, crane 102 substitutes it, increasing its workload. The same meaning applies to the other columns.

Another important vector, not mentioned before but important for the function Block Queue Simulation, is *Stacking Time Array*. It is a horizontal vector, whose size coincides with the number of blocks, that reports the stacking time at each block, depending on which crane is serving it. It is initialised with the same *Normal Stacking Time* for every block.

6.4 Functions

This section provides an insight into the principal functions of the model.

6.4.1 Block Assignment Function

The Block Assignment procedure is demanded to a specific function that, given the data of an incoming container and the current state of the yard, finds the best available block in the yard. As mentioned above, this function is called whenever the variable *Time of Arrival* of an incoming container, contained in *Times* matrix, equals the time unit of the loop. Before presenting the pseudocode, a little introductory description of one of the input variables of the function is needed: *Block Utilisation*. It is a horizontal vector that reports the number of containers currently stored in each block. For example:

$$Block\ Utilisation = [35\ 72\ 15\ 20\ 66\ 89\ 44\ 7\ 11] \qquad (6.11)$$

The variable states that there are 35 containers in block 1, 72 containers in block 2, 15 containers in block 3 and so on. The elements of the vector are divided by the total amount of available slots in each block

With regards to the input variables, it is important to specify that *Predicted Dwell Time, ID, Time of Arrival and Quay* refer to the incoming container. The pseudocode for the function is the following:

```
Input = Predicted Dwell Time, Block Utilisation, Block Feasibility, Number of
Blocks, Distances, Congestion, ID, Time of Arrival, Block-Quay Distances, Quay,
Congestion Matrix, Vehicle Speed, Yard Matrix, Available slots in the block
Output = Block Utilisation, Block Feasibility, Congestion Matrix, Congestion
initialise Max Block VoG as 0
for j=1:Number of Blocks
    if Block Feasibility of block j is 1
       calculate Block VoG of block j using the Block Assignment policy of a Fuzzy
       System, picking its input variables from Block-Quay Distances, (Block
       Utilisation / Available slots in the block), Congestion of block j
       if Block VoG > Max Block VoG
```

```
            assign the value of Block VoG to Max Block VoG
            assign the value of j to the variable Chosen Block
         end
      end
end
update Block Utilisation of Chosen Block by 1
if Block Utilisation of Chosen Block >= Available slots in the block
   set Block Feasibility of Chosen Block at 0
end
calculate Standard Deviation of the elements of Block Utilisation and assign it
to the variable Standard Deviation of the Blocks
calculate Travel Time = Block Quay Distance (Chosen Block, Quay) / Vehicle Speed
calculate Time of Arrival at the Block = Time of Arrival + Travel Time
update Congestion Matrix by adding a line with the data about the incoming
container
for k=1:Number of Blocks
    for i=1:number of lines of Congestion Matrix
        count the containers in the queue at block k
    end
update vector Congestion at position k with the queue at block k
end
```

The main outputs of this function are the allocation of the incoming container to one of the blocks in the yard and the updating of the variable *Congestion Matrix*, which records the details related to the travel and stacking phase of the containers. Each line corresponds to a container while each column contains one of its attributes. The columns are:

- *ID*. It is the progressive number of the containers as they arrive in the yard and it is used as an ID since two containers cannot be unloaded from a vessel at the same time.
- *Predicted Dwell Time*
- *Time of Arrival*
- *Quay*
- *Travel Time*
- *Time of Arrival at the Block*
- *Queue*. This variable has to be interpreted as the actual waiting time of a container before the stacking phase (set at 0 in Block Assignment Function and modified within Block Queue Simulation Function)

- The first time unit when the incoming container is ready to be placed in a stack since the queue has cleared (set at 0 in Block Assignment Function and modified within Block Queue Simulation Function). The variable is called *First Time Available*
- The *Allocation Time* of the incoming container: the time unit when the incoming container is effectively placed in its stack of destination, leaving the queue (set at 0 in Block Assignment Function and modified within Block Queue Simulation Function)
- *Chosen Block*. It is the reference number of the block with the highest VoG, chosen as the destination for the incoming container
- A Boolean variable which indicates whether the incoming containers is still part of a queue: it is 1 when the container is travelling towards Chosen Block, is waiting in front of it or is being transferred from the internal truck to the stack of destination while it goes to 0 once the container has been stacked. The name of this variable is *Travelling*
- Another Boolean variable is used to show whether the incoming container has arrived at the block or not: it assumes value 1 when the container is waiting in front of the block or is being allocated on the top of a stack; it is 0 during the travelling phase, since the container has not physically arrived at the block, and after the allocation is completed. The name of this variable is *Waiting at the Block*
- *Max Block VoG* (which is the VoG of chosen Block)
- The value of *Congestion at Chosen Block* prior to Block Assignment, which corresponds to the element corresponding to *Chosen Block* of the vector *Congestion*
- The value of *Block Utilisation of Chosen Block* prior to Block Assignment, which is the value of the element of the vector *Block Utilisation* corresponding to *Chosen Block*
- *Standard Deviation of the Blocks* after Block Assignment

6.4.2 Block Queue Simulation

This function controls the stacking and the end of the travelling phases of the incoming containers. It is called whenever a container arrives in front of its block of destination (when the variable *Time of Arrival at the Block*, recorded in *Congestion Matrix*, equals the time unit of the time loop). Among the input values of the function, ID, Chosen Block and Time of Arrival at the Block refer to the incoming container, just arrived at the block. Its pseudocode is shown below:

```
Input = Congestion Matrix, ID, Chosen Block, Time of Arrival at the Block, Stacking
Time Array
Output = Congestion Matrix
Retrieve Stacking Time of Chosen Block from Stacking Time Array
Initialise Block Matrix as an empty matrix
for s=1:number of lines of Congestion Matrix
    if block in line s of Congestion Matrix coincides with Chosen Block AND
    Waiting at the Block in lines s = 1
        add line s of Congestion Matrix to Block Matrix
    end
end
if Block Matrix is not empty
   find the row in Block Matrix with the latest Time of Arrival at the Block and
   call the corresponding container Last Container
   if Time of Arrival at the Block >= Allocation Time of Last Container
      assign the value of Time of Arrival at the Block to the variable First Time
      Available for Stacking
      Allocation Time = First Time Available for Stacking + Stacking Time
      Queue at Block = 0
   else
      assign the value of Allocation Time for Last Container to First Time
      Available for Stacking
      Allocation Time = First Time Available for Stacking + Stacking Time
      Queue at Block= First Time Available for Stacking – Time of Arrival at the
      Block
   end
else
    assign the value of Time of Arrival at the Block to the variable First Time
    Available for Stacking
    Allocation Time = First Time Available for Stacking + Stacking Time
    Queue at Block = 0
end
update Congestion Matrix with the values of Queue at Block, First Time Available
for Stacking, Allocation Time and Waiting at the Block = 1 for the incoming
container
```

In a few words, this function reproduces the different situations that might happen when an incoming container arrives in front of the block: there might be no containers at the block, so the incoming one is immediately ready to be stacked, or there might be a queue of containers, so that the incoming one is forced to wait before the Stack Assignment phase can begin. Information about the current condition of the arriving container (travelling, waiting in front of the block, being stacked etc.) are conveyed into specific variables that are part of *Congestion Matrix*.

6.4.3 Stack Assignment Function

The Stack Assignment phase of the allocation process is governed by a specifically designed function. It is triggered when the variable *First Available Time* of one of the containers in *Congestion Matrix* is equal to the current time unit of the time loop. This means that that container is ready to be stacked since there are no other containers in front of him in the queue. It is in this moment that the choice of the best possible stack in the block of destination is made. Before detailing the pseudocode, a brief description of another support variable is needed: *Stack Height*. *Stack Height* is a horizontal vector that is defined for each block. The length of the vector corresponds to the number of stacks in the referring block. Each element of the vector indicates the current height of the corresponding stack. An example is shown below:

$$Stack\ Height\ 3 = [1\ 4\ 4\ 4\ 3\ 3\ 0\ 2\ 3\ \cdots] \tag{6.12}$$

The variable shows that, in block 3, the first stack is 1-tier high, stack 2, 3 and 4 are full (the block is supposed to have a maximum height of 4 tiers), stack 5 is 3 tiers high and so on.

Regarding the input variables of the function, *Estimated Time of Departure, Chosen Block, ID, Time of Arrival, Allocation Time, Queue, Max Block VoG, Congestion at Chosen Block, Block Utilisation of Chosen Block* and *Standard Deviation of the Blocks* are attributes associated with the incoming container and derived from *Congestion Matrix* while *Stack Feasibility, Number of Stacks in a block* and *Maximum Stack Height* are related to *Chosen Block*. It is important to note that *Estimated Time of Departure* is defined by the sum of the *Allocation Time* and *Predicted Dwell Time* of the incoming container: it is an estimation of its date of delivery. It is now possible to introduce the pseudocode:

```
Input = Estimated Time of Departure, Chosen Block, Stack Height, Container Database, Stack Feasibility, Number of Stacks in Chosen Block, Maximum Stack Height, ID, Predicted Yard, Real Yard, Congestion Matrix, Time of Arrival, Allocation Time, Block Quay Distances, Block Gate Distances, Queue, Max Block VoG, Congestion at Chosen Block, Block Utilisation of Chosen Block, Standard Deviation of the Blocks
Output = Container Database, Stack Height, Stack Feasibility, Predicted Yard, Real Yard, Congestion Matrix
initialise Max Stack VoG as 0
assign Predicted Yard to the support variable Help Yard
for j=1:Number of Stacks in Chosen Block
    if Stack Feasibility of stack j is 1
       assign Predicted Dwell Time to the slot on top of stack j
       assign to the variable Possible New Stack Height the value Stack Height of stack j + 1
```

```
        initialise Rehandles at 0
        initialise Efficient Moves at 0
        while stack j in Help Yard is not empty
            find earliest Estimated Time of Departure (ETD) in stack j of Help
            Yard and assign it to the variable Earliest ETD
            find the position of Earliest ETD in stack j of Help Yard (which line
            of Help Yard contains Earliest ETD) and assign it to the variable
            Position of Earliest ETD
            for k=(Maximum Stack Height - Possible New Stack Height + 1):Position
            of Earliest ETD
                if position k in the current stack is equal to Position of Earliest
                ETD
                    increase Efficient Moves by 1
                else
                    increase Rehandles by 1
                end
            end
            if Position of Earliest ETD coincides with the top tier of a full
            stack j OR Position of Earliest ETD is at the top of a non-full stack
            j
                substitute container in Position of Earliest ETD of stack j in
                Help Yard with an empty slot
                reduce Possible New Stack Height by 1
            else
                for y=Position of Earliest ETD:-1:(Maximum Stack Height –
                Possible New Stack Height + 1)
                    if y is a top tier position (first line of Help Yard)
                        substitute container in position y and stack j in Help
                        Yard with an empty slot
                    else
                        substitute container in position y and stack j in Help
                        Yard with the container in tier y-1 above it (it
                        corresponds to move the relocated containers down)
                    end
                end
                reduce Possible New Stack Height by 1
            end
        end
        calculate Stack VoG of stack j using the Stack Assignment policy of a Fuzzy
        System, picking its input variables from Rehandles and Stack Height at
        stack j
        if Stack VoG > Max Stack VoG
            assign the value of Stack VoG to Max Stack VoG
            assign the value of j to the variable Chosen Stack
        end
    end
end
update Stack Height of Chosen Stack by 1
```

```
if Stack Height of Chosen Stack equals Maximum Stack Height
   set Stack Feasibility of Chosen Stack at 0
end
update Container Database by adding a line with the data about the incoming container
inset incoming container with its Estimated Time of Departure on the top of Chosen Stack in Predicted Yard
inset incoming container with its Estimated Time of Departure on the top of Chosen Stack in Real Yard
```

With the first for-loop, the function simulates stacking the incoming container in each one of the stacks that compose the block of destination. The successive while-loop counts the number of potential rehandles caused by stacking the incoming container on top of each stack; to evaluate the amount of relocations, the retrieval process of the containers is simulated in each stack, according to their *Estimated Time of Departure*. It is important to note that retrieval simulation is based solely on the predicted dwell times and not on the real ones, affect by events and disturbances, since they are obviously not available upon arrival of the containers. Moreover, the simulation is needed only for the Stack Assignment policies that employ Rehandles as an input variable.

There are two main outputs of the Stack Assignment function. The first one is the updating of *Predicted Yard* and *Real Yard* matrices: in both cases, the incoming container is placed on the top of the stack of destination and is represented by its *Estimated Time of Departure*. If a series of disturbances ends up modifying the time of departure of the stacked containers, the change is only reflected in *Real Yard* and not in *Predicted Yard.* The other important output is the updating of another matrix: *Container Database*. It represents the principal database that records all the key data about the containers that have passed through the container terminal and a principal source to evaluate the performances of the stacking policy. In a similar way as for *Congestion Matrix*, each line of *Container Database* corresponds to a container while each column contains one of its attributes. Some of these attributes are shared with *Congestion* Matrix. The columns are:

- *ID*. It is the same variable defined in *Congestion Matrix*.
- *Estimated Time of Departure*
- *Real Time of Departure*. It is obtained by modifying the *Estimated Time of Departure* according to the disturbances that affect the terminal.
- *Time of Arrival*
- *Allocation Time*

- *Predicted Dwell Time*
- *Real Dwell Time*. It is calculated as the difference between the *Real Time of Departure* and *Allocation Time*
- *Block, Stack* and *Tier of Arrival*. They are three variables that indicate the block, the stack and tier where the incoming container is allocated for the first time, immediately after Stack Assignment. Needless to say, *Block of Arrival* coincides with *Chosen Block* and *Stack of Arrival* with *Chosen Stack*.
- *Current Block, Stack* and *Tier*. They are three variables that indicate the block, the stack and the tier where the container is allocated at the current time unit. They are initialised with the same values of *Block, Stack* and *Tier of Arrival*. Once the container has left the yard, the three variables assume a null value.
- *Final Block, Stack* and *Tier*. The three variables indicate the last position of the container before leaving the yard at the end of its dwell time. They are initialised as three 1s and updated after the retrieval of the container.
- *How Many Times a Container Has Been Moved*. It is a variable that counts the number of times a container has been relocated during retrieval operations to get another container stacked underneath. It is worth reminding that, under the current assumption, detailed in the previous chapter, relocated containers need to be repositioned in the same stack and with the same order they had prior to the retrieval procedure.
- *Max Block VoG*
- *Max Stack VoG.* It is the VoG of *Chosen Stack*
- *Quay*
- *Block-Quay Distance*. The distance from the *Quay* where the container has been unloaded to *Chosen Block*
- *Block-Gate Distance.* The distance from *Chosen Block* to the *Gate*.
- *Rehandles*. As defined in the pseudocode, this variable contains the number of potential rehandles caused by the container according to its *Estimated Time of Departure*
- *Congestion at Chosen Block*
- *Queue*
- *Block Utilisation of Chosen Block*

- *Stack Height of Chosen Stack*. This variable reports the height of *Chosen Stack* at the time unit when the Stack Assignment decision has been taken, so shortly before the allocation of the container.
- *Standard Deviation of the Blocks*

6.4.4 Container Retrieval Function

This function controls the retrieval process of the containers at the end of their dwell time. It is called whenever the current time unit equals the *Real Time of Departure* of one of the containers in the yard. Another brief introductory description of a specific variable is needed: the vector called *Rehandles*. A vector of this kind is defined for every block and its size corresponds to the number of stacks of each specific block. The elements of the vector represent the number of rehandles that occurred in each stack. An example is shown below:

$$Rehandles\ 4 = [11\ 5\ 7\ 4\ 9 \cdots] \qquad (6.13)$$

The example shows that, in block 4, during a certain time period, 11 rehandles occurred in stack 1, 5 rehandles in stack 2, 7 rehandles in stack 3 and so on. Summing the number of rehandles of each stack and for each block results in the total number of rehandles over a given period. A complementary set of vectors called *Efficient Moves* are also defined for each block: each one of their elements represents the number of efficient moves that occurred in a given stack located in a certain block. The pseudocode can now be presented:

```
Input = Predicted Yard, Real Yard, Rehandles, Efficient Moves, Real Time of
Departure, Block, Number of Stacks in a Block, Stack Height, Maximum Stack Height,
Block Utilisation, Block Feasibility, Stack Feasibility, Container Database, Yard
Matrix, Time of Arrival
Output = Predicted Yard, Real Yard, Rehandles, Efficient Moves, Stack Height,
Block Utilisation, Block Feasibility, Stack Feasibility, Container Database
for j=1:Number of Stacks in a Block
    find the line in Real Yard where the Real Time of Departure of the container
    is stored and assign it to the variable Position of RTD
    if the container in position (Position of RTD, j) in Real Yard has a departure
    time that coincides with the Real Time of Departure
       assign the value of j to the variable Last Stack
       find the line representing the container that is leaving the yard in
       Container Database
       update Final Block in Container Database with the value of the variable
       Block
       update Final Stack in Container Database with the value of the variable
       Last Stack
       update Final Tier in Container Database with the value of Maximum Stack
       Height – Position + 1
    update Current Block in Container Database with the value 0
    update Current Stack in Container Database with the value 0
    update Current Tier in Container Database with the value 0
    end
    for k=(Maximum Stack Height – Stack Height (Last Stack) + 1):Position of RTD
        if position k in Last Stack is equal to Position of RTD
          increase Efficient Moves in Last Stack by 1
        else
          increase Rehandles in Last Stack by 1
        end
    end
    if Position of RTD coincides with the top tier of a full Last Stack OR
      Position of RTD is at the top of a non-full Last Stack
      substitute container in Position of RTD and stack Last Stack in Real Yard
      with an empty slot
      substitute container in Position of RTD and stack Last Stack in Predicted
      Yard with an empty slot
      reduce Stack Height of Last Stack by 1
    else
      assign the value of Stack Height (Last Stack) to the support variable
      Height
      for x=Position of RTD:-1:(Maximum Stack Height – Height + 1)
          if x is a top tier position (first line of Real or Predicted Yard)
            substitute container in position x and stack Last Stack in Predicted
            Yard with an empty slot
            substitute container in position x and stack Last Stack in Real Yard
            with an empty slot
          else
            substitute container in position x and stack Last Stack in Predicted
            Yard with the container in tier x-1 above it (it corresponds to
            moving the relocated containers down)
            substitute container in position x and stack Last Stack in Real Yard
            with the container in tier x-1 above it (it corresponds to moving
            the relocated containers down)
            for y=1:size(Container Database, 1)
                find the line corresponding to the container moved to position
                x in Container Database
                update the value of Current Tier in Container Database with
                Maximum Stack Height (Last Stack) – x + 1
                update the value of variable How Many Times a Container Has Been
                Moved in Container Database by 1
            end
          end
      end
end
reduce Block utilisation of Block by 1
if Block Utilisation of Block < Available slots in the block
   set Block Feasibility of Block at 1
```

```
end
if Stack Height (Last Stack) < Maximum Stack Height
   Stack Feasibility (Last Stack) = 1
end
```

The retrieval process has been modelled exactly in the same way as for the Stack Assignment function. In this case, however, relocations and efficient moves are not simply simulated but are effectively implemented. The main output of the function consists of updating the matrices *Container Database* and *Real* and *Predicted Yard* and counting the number of Rehandles and Efficient Moves caused by the procedure. This function shows that the retrieval of the containers is based on their Real Time of Departure. Their allocation, on the other hand, when Rehandles is an input variable, employs the Predicted Time of Departure. In this time difference lies the impact of the disturbances that affect primarily containers and their attributes. These events, and their implementation, are detailed in the next subsection.

6.5 Events and Disturbances

Since the aim of the work is to define a dynamic allocation strategy that is able to react in real time to disturbances that affect the container terminal, it is fundamental to have a set of events implemented and integrated with the main Matlab model. Given the available data, five type of events have been modelled and each one has been given a code number:

- Event Type 1: Blocking a block (Yard-related event). One or more blocks are excluded for a certain period of time from the available allocation sites for an incoming container, even if they are not full. The closure of the block is valid only for allocation, not retrieval. The reasons for this exclusion are varied: a customs control that requires containers in the block to be opened, a stack falling down etc. This kind of event affects primarily the yard and its layout especially: closing one or more blocks influences the distances and block utilisations, two yard-related attributes, of the available options for an incoming container.
- Event Type 2: Traffic Jam (Container-related event). This event captures what happens when a group of external trucks is caught in a traffic jam in the proximity of the container terminal. Each traffic jam is characterised by its duration, which also defines the delay that affects the containers that were expected to leave the yard during that timeframe: their Real Time of Departure, a container-related attribute, is postponed by a quantity that equals the duration of the traffic jam.

- Event Type 3: Drivers' Strike (Container-related event). The drivers of the external trucks may go on strike, thus preventing the retrieval of the containers that are expected to leave the yard during the duration of the protest. A drivers' strike is expected to happen less frequently than a Traffic Jam and to have a much longer duration. The main impact is again on the Real Time of Departure: all the affected containers have their retrieval postponed at the end of the occurring strike.
- Event Type 4: Late Arrival of an External Truck (Container-related event). It refers to an external truck that arrives at the yard and retrieves a container much later than its Predicted Time of Departure. The reasons for this late arrival are the most varied (the driver got lost, a mechanical failure which required assistance etc.) and exclude a traffic jam and a drivers' strike, since they are already covered by the previous two event types and impact multiple containers at the same time. As for the other two events above, is a container related event since it impacts the Real Time of Departure which is a container related attribute
- Event Type 5: Crane Breakdown (Resource-related event). When one of the cranes that serve the blocks to allocate incoming container is out of service, it must be substituted by another crane, as shown in Table 4.7. The result of this substitution is represented by an increase in stacking time, which is the time it takes for every container to be taken from an internal truck, to be moved and placed on the top of its stack of destination. This happens because the replacement crane is also serving its original group of blocks, so it has doubled its workload. The causes of this breakdown could be a mechanical problem or a regular maintenance control that prevents the crane from being used. This is a resource-related event since it impacts primarily the performances of the cranes, which are listed among the resources that work in the yard.

It is important to highlight that the implemented events have also been chosen with an eye for the proposed classification. In fact, there is at least one event for each one of the three main categories proposed for Low Level Events: Event 1 belongs to the category of Yard-related events, Events 2, 3 and 4 to Container-related events and Event 5 to Resource-related events.

6.5.1 The Implementation of the Events in Matlab

To develop the events, a specific Matlab script has been developed. This script has the duty to create a matrix called *Event Matrix* and a series of related event-specific matrices.

First of all, the time period and the pace for the event creation have to be defined. The time period should generally coincide with the interval of the time loop of the main model (*Max Time – Min Time* of Figure 6.1), in order to distribute the events over the whole simulation span. With regards to the pace, the script is able to generate events every 60 minutes but nothing forbids to select a different rate.

The concept behind the creation of the events is probability. Once every 60 minutes, or according to the desired pace, an array with a size of 5 is created: the position in the array corresponds to the number that describes the type of event. Each element of the array is a randomly generated number, between 0 and 1, which represents the probability of the corresponding disturbance to happen at a given point in time. For example:

$$Events\ Probability = [0.712\ 0.441\ 0.129\ 0.423\ 0.823] \qquad (6.14)$$

The vector shows that, at a certain time unit, Event 1 has a probability of happening of 71.2%, Event 2 has a probability of 44.1% and so on. The model is constructed to consider one single event at time, so one of the five events and its probability have to be chosen: the idea is to select the event that has the highest probability of happening within the array. Therefore, in the example, Event 5 is the one considered to be happening at the specific time unit since it has the highest probability of the five, 82.3%.

In reality, however, events might also not happen and the operations in the terminal can flow without obstacles. To model this, a vector of thresholds is introduced. Each element represents a threshold valid for the corresponding event. As an example:

$$Threshold\ Vector = [0.5\ 0.6\ 0.7\ 0.7\ 0.8] \qquad (6.15)$$

Each probability is then compared with the respective threshold, resulting in a difference.

$$Probability\ Difference = [0.212 \quad -0.159 \quad -0.571 \quad -0.277 \quad 0.023] \qquad (6.16)$$

As mentioned above, only one disturbance can happen at a single time, so a selection is needed: the event that actually happens is the one with the largest positive difference from the threshold. If all the differences are negative, it means that no event has overcome its threshold, resulting in no events happening in the yard at that given moment. As shown in

6.14, the thresholds do not need to be the same for all the events but they can be customised according to specific needs.

The result of this procedure is a list of events, which can be seen as a first draft of an *Event Matrix*, that happen every 60 minutes. In real life, however, the events that have been modelled have a duration that can exceed 60 minutes. The length of the single events is recorded in a variable called *Event Duration*, and this information is used to modify the previous list of events. The result is the final version of *Event Matrix*. In this matrix, the lines show different generation procedures of random probabilities of the events; the first column represents the pace at which the event creation is evaluated ; the columns from the second to the sixth indicate the probability of happening for each event type, starting from Event 1 and ending with Event 5 in ascending order; the columns from the seventh to the eleventh record the difference between the probability of happening of each event with its respective threshold; the twelfth column shows the largest of these differences, the thirteenth the event type to which this difference belongs and the fourteenth states whether an event of that type happens or not (if the difference is positive, the event happens, if it is negative, no events happen); finally, the last column represents the time unit when the event is supposed to end. An example of Event Matrix and a visual representation of it are shown in Table 6.1 and in Figure 6.2 respectively.

Time Unit	Prob Event 1	Prob Event 2	Prob Event 3	Prob Event 4	Prob Event 5	Diff from thre. Ev 1	Diff from thre. Ev 2	Diff from thre. Ev 3	Diff from thre. Ev 4	Diff from thre. Ev 5	Largest Difference	Event Type	Is the Event Happening?	Event Ending
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
300	0.512	0.430	0.652	0.721	0.945	-0.188	-0.270	-0.048	0.021	0.245	0.245	5	1	535
360	0.512	0.430	0.652	0.721	0.945	-0.188	-0.270	-0.048	0.021	0.245	0.245	5	1.1	535
420	0.512	0.430	0.652	0.721	0.945	-0.188	-0.270	-0.048	0.021	0.245	0.245	5	1.1	535
480	0.512	0.430	0.652	0.721	0.945	-0.188	-0.270	-0.048	0.021	0.245	0.245	5	1.1	535
540	0.881	0.125	0.751	0.810	0.555	0.181	-0.575	0.051	0.110	-0.145	0.181	1	1	775
600	0.881	0.125	0.751	0.810	0.555	0.181	-0.575	0.051	0.110	-0.145	0.181	1	1.1	775
660	0.881	0.125	0.751	0.810	0.555	0.181	-0.575	0.051	0.110	-0.145	0.181	1	1.1	775
720	0.881	0.125	0.751	0.810	0.555	0.181	-0.575	0.051	0.110	-0.145	0.181	1	1.1	775
780	0.331	0.478	0.652	0.071	0.212	-0.369	-0.222	-0.048	-0.629	-0.488	-0.048	3	0	840
840	0.723	0.841	0.777	0.111	0.565	0.023	0.141	0.077	-0.589	-0.135	0.141	2	1	1075
900	0.723	0.841	0.777	0.111	0.565	0.023	0.141	0.077	-0.589	-0.135	0.141	2	1.1	1075
960	0.723	0.841	0.777	0.111	0.565	0.023	0.141	0.077	-0.589	-0.135	0.141	2	1.1	1075
1020	0.723	0.841	0.777	0.111	0.565	0.023	0.141	0.077	-0.589	-0.135	0.141	2	1.1	1075
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Table 6.1 Event Matrix example. In the second-to-last column, 1 and 1.1 mean that the Event is happening while 0 means that no Events happen

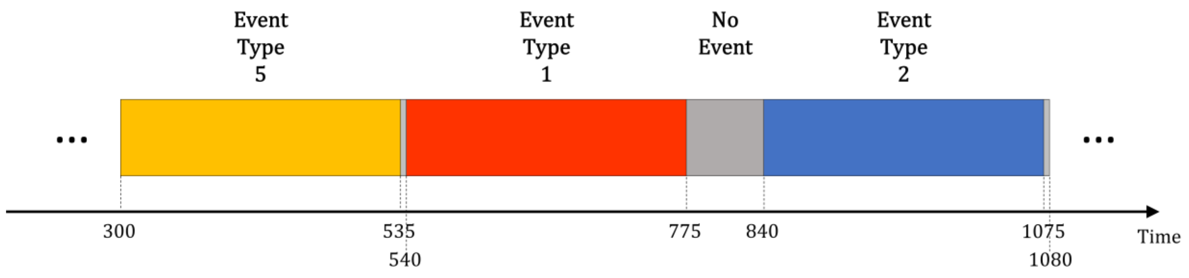


Figure 6.2 A visual representation of the events generated through Event Matrix. The rectangles in grey represent a No Event scenario.

Among the columns of *Event Matrix*, the four ones highlighted in yellow in Table 6.1 are definitely the most important for the purpose of simulating an event: in fact, they show whether an event is supposed to happen or not, to which type it belongs to, when it starts and when it ends. It is also interesting to note that some lines repeat themselves: this is the result of the modifications of the original draft of the matrix, after the introduction of the *Event Duration* variables, and they indicate that a certain event is continuing beyond a length of 1 hour.

Once *Event Matrix* has been defined, it is used to develop five Event-specific matrices. They are matrices that report the starting time of each event of a given type plus other parameters that are specific to that kind of disturbance. They are list below:

1. Event 1: *Blocked Blocks Matrix*. It is a matrix that indicates the starting time of each Event 1 in *Congestion Matrix* and a list of the blocks that are shut down during that event. The number of blocked blocks and which blocks are excluded from the possible destinations for an incoming container, are determined at random.

$$Blocked\ Blocks\ Matrix = \begin{bmatrix} 180 & 1 & 7 & 0 & 0 \\ 480 & 2 & 0 & 0 & 0 \\ 1020 & 4 & 5 & 6 & 9 \\ \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix} \quad (6.17)$$

   The matrix of the example shows that during the Event of type 1 that happens at time unit 180, blocks 1 and 7 are closed. For the Event 1 that happens at time unit 480, only block 2 is closed. For the Event 1 that happens at time unit 1020, four blocks are closed: 4, 5, 6 and 9; and so on.

2. Event 2: *Jam Delay Matrix*. It is a matrix that indicates the starting time of each Event 2 in *Congestion Matrix* and the time delay that each external truck suffers because of that traffic jam, incorporated in the variable *Jam Delay*. Again, the extent

of the delay can be generated randomly, choosing one number between two extreme values.

$$Jam\ Delay\ Matrix = \begin{bmatrix} 300 & 235 \\ 1260 & 118 \\ 2700 & 180 \\ \cdots & \cdots \end{bmatrix} \tag{6.18}$$

The matrix reports that the traffic jam that happens at time unit 300, causes a delay of 235 time units, the jam that happens at time unit 1260 is the cause of a time delay of 118 units and so on.

3. Event 3: *Strike Length Matrix*. It is a matrix that indicates the starting time of each Event 3 in *Congestion Matrix*, its ending time and the duration of the strike. This last parameter is generated randomly

$$Strike\ Length\ Matrix = \begin{bmatrix} 60 & 886 & 826 \\ 13200 & 13960 & 760 \\ 34200 & 34819 & 619 \\ \cdots & \cdots & \cdots \end{bmatrix} \tag{6.19}$$

The matrix shows that the strike that starts at time unit 60, lasts for 826 time units and ends at time unit 886, the strike that starts at time unit 13200 lasts for 760 time units and finishes at time unit 13960, and so on.

4. Event 4: *Late Arrivals Matrix*. It is a matrix indicates the time when Event 4 starts from *Congestion Matrix* and the time delay that affects one of the containers that are supposed to leave the yard between the start and the end of that disturbance. The length of this time delay is chosen randomly between two extreme values.

$$Late\ Arrivals\ Matrix = \begin{bmatrix} 360 & 6279 \\ 2040 & 9392 \\ 5580 & 4816 \\ \cdots & \cdots \end{bmatrix} \tag{6.20}$$

5. Event 5: *Crane Breakdown Matrix*. It is a matrix that indicates the starting time of each Event 5 in *Congestion Matrix* and the ID of the crane that breaks down during that event. The IDs are selected randomly for each event.

$$Crane\ Breakdown\ Matrix = \begin{bmatrix} 5700 & 103 \\ 8700 & 102 \\ 10560 & 101 \\ \cdots & \cdots \end{bmatrix} \tag{6.20}$$

The matrix of the example shows that at time unit 5700 the reach-stacker with code 103 breaks down, at time unit 8700 the reach-stacker with code 102 is not available and so on.

Therefore, the results of one run of the Matlab script dedicated to event generation are one *Event Matrix* and five Event-specific matrices of the type described above. This set of matrices are grouped together into one file that serves as in input to the main Matlab model that simulates the behaviour of the terminal. In this way it is possible to test how different allocation strategies behave with the same event combination and the same effects on the yard and on the container, eliminating any element of randomness. A comprehensive view of the event generation process is shown in Figure 6.3.

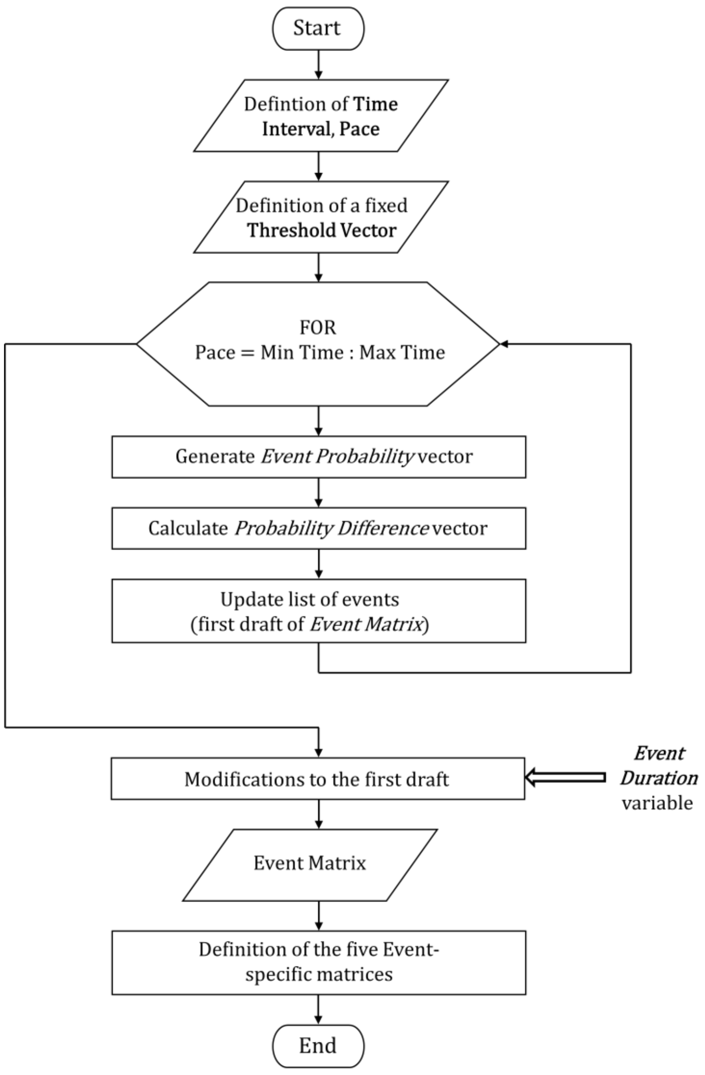

Start
Defintion of Time Interval, Pace
Definition of a fixed Threshold Vector
FOR
Pace = Min Time : Max Time
Generate Event Probability vector
Calculate Probability Difference vector
Update list of events
(first draft of Event Matrix)
Modifications to the first draft
Event Duration variable
Event Matrix
Definition of the five Event-specific matrices
End

Figure 6.3 Flowchart that shows the main steps of the event generation process

6.5.2 Integration of the Events with the Main Matlab model

Event Matrix and the other five event-specific matrices are used as input variables for the main Matlab model, where they are initialised prior to the time-loop. The simulation of the events is then demanded to an interaction between this set of matrices and five functions, located within the time-loop. Those functions are:

- Yard Event Function. This function recognizes what time it is in the simulation and whether an Event of type 1 is happening or not, according to *Event Matrix*. In the first case, it retrieves the list of corresponding blocked blocks from *Blocked Blocks Matrix*. Those blocks are then excluded from the possible destinations for an incoming container for the duration of the event by setting at 0 (when it is not already null) their corresponding value in *Block Feasibility*.
- Traffic Jam Function. This function recognizes what time it is in the simulation and whether an Event of type 2 is happening or not, according to *Event Matrix*. When this is true, all the containers that are expected to leave the terminal during the duration of the event have their Real Time of Departure postponed by the same quantity, defined for the specific event in *Jam Delay Matrix*.
- Strike Function. This function recognizes what time it is in the simulation and whether an Event of type 3 is happening or not, according to *Event Matrix*. When a strike is actually happening, all the containers that are supposed to leave the terminal during the duration of the event have their Real Time of Departure moved immediately after the end of the strike, which is indicated in *Strike Length Matrix*.
- Single Late Arrival Function. This function recognizes what time it is in the simulation and whether an Event of type 4 is happening or not, according to *Event Matrix*. If so, one single container among the ones that are expected to leave the yard during the event has its Real Time of Departure delayed by time interval described in *Late Arrivals Matrix*. In this case, the duration of the event is not a proper time extent but is just used as a time slot where to find a container to be delayed.
- Crane Breakdown Function. This function recognizes what time it is in the simulation and whether an Event of type 5 is happening or not, according to *Event Matrix.* In the first case, the reach-stacker that is out of work during the corresponding event is indicated in *Crane Breakdown Matrix*. The impact on the yard is modelled by doubling the stacking time in the blocks served by replacement reach-stacker.

The combination of the use of the matrices and the event-related functions constitutes what, in Figure 6.1, has been called Events Simulation.

6.6 The Results of the Model

After the end of each simulation, an Excel file is produced, containing fundamental data about the operations in the terminal. This file includes *Congestion Matrix*, *Container Database*, two tables showing the number of Rehandles and Efficient Moves in each block, a table reporting the Congestion in each block over the time period under examination, a list that shows the Rehandles in every stack for every block and two tables that detail Block Utilisation and the overall Yard Utilisation over time.

Thanks to the data, and especially to *Container Database*, which records information about every container that has passed through the yard during the simulation, it is possible to evaluate the performances of a stacking strategy: this performance evaluation effort represents the core of the next chapter.

CHAPTER 7

Computational Findings

This chapter presents a detailed description of the training and testing phases and their results. A list of the main assumptions that have been made prior to running the model is provided first. Then, the KPIs used to measure the performances of the Fuzzy Systems are introduced, with a focus on a multi-criteria evaluation. The testing phase is detailed in all its parts, especially considering the different approaches to define the best system. Finally, the testing phase and its results are listed and discussed.

7.1 The Assumptions of the Model

The model has been tested under a series of assumptions and hypothesis, which have to be clearly stated in order to understand the boundary conditions of the tests and their results. Those assumptions are valid throughout all the training and testing phases. The main suppositions are presented in the next subsections.

7.1.1 Assumptions Regarding the Yard Layout

As mentioned in section 4.2, the container terminal at the Port of Arica is composed of 18 blocks. They are supposed to be split in two groups: 9 blocks are dedicated to import containers and the other 9 to export containers. Since the available data only include import containers, the focus should then be only on the 9 import-dedicated blocks, neglecting the presence of the other 9. This choice brings two advantages:

- It is easier to model the behaviour of the terminal since it is possible to disregard the flows of the export containers
- The available capacity to stack incoming containers is reduced, since there are less usable slots. This allows to reach higher levels of overall yard utilisation with the same number of incoming containers and in the same time interval: a high yard utilisation is a preferable scenario for testing since it allows to evaluate the performances of a stacking strategy under stressful conditions for the port

The 9 blocks that have been dedicated to import containers are presented in Table 7.1. The second column in the table indicates the code that is given to each block in the Matlab model. Table 7.1 is derived from Table 4.3, which had a column that specified the size of the containers that were allowed to be placed in each block. In this work, however, all the incoming containers are assumed to be 40’ to simplify the model. Therefore, the total capacity of the yard in terms of container slots is 930.

Regarding distances, another important supposition has been made: as anticipated in Chapter 4, the external trucks that retrieve import containers at the end of their dwell time, enter and leave the yard from Gate 2; Gate 1 is dedicated to export containers only, so the related distances are disregarded. Block-Gate Distances are reported in Table 7.2

No assumptions have been made with regards to the quays: arriving vessels can berth in any of the 6 available sites at the TPA without restrictions; Quay-Block Distances of the 9 blocks are shown in Table 7.3.

Block Name	Block Code	Bays	Rows	Tiers	Block Capacity
ZB7	**1**	8	3	4	96
ZB5	**2**	9	3	4	108
ZB6	**3**	7	3	4	84
S1J	**4**	3	6	4	72
S1H	**5**	3	6	4	72
S1G	**6**	3	6	4	72
S1F	**7**	3	6	4	72
ZB2	**8**	9	3	5	135
ZB3	**9**	9	3	5	135

Table 7.1 Characteristics of the blocks

	Block	**ZB7**	**ZB5**	**ZB6**	**S1J**	**S1H**	**S1G**	**S1F**	**ZB2**	**ZB3**
	Block Code	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Gate	**Gate 2**	197.12	217.27	122.1	227.99	188.96	231.1	257.82	256.85	164.4

Table 7.2 Block-Gate Distances in metres of the 9 blocks for import containers

		Quays/Berthing Sites					
Block	**Block Code**	**S1**	**S2A**	**S2B**	**S3**	**S4**	**S5**
ZB7	**1**	384.69	429.39	437.93	435.41	688.5	893.92
ZB5	**2**	301.04	387.1	403.68	401.64	686.27	895.88
ZB6	**3**	261.74	463.31	485.14	477.86	788.64	990.45
S1J	**4**	142.95	386.86	406.24	396.62	668.16	876.57
S1H	**5**	197.13	434.71	440.74	430.52	714.97	924.94
S1G	**6**	276.12	493.28	508.9	498.88	774.66	987.51
S1F	**7**	296.22	514.18	532.85	538.64	828.71	1015.49
ZB2	**8**	286.9	333.27	369.15	364.66	634.03	851.39
ZB3	**9**	211.57	422.15	450.24	448.55	727.52	930.16

Table 7.3 Quay-Block Distances in metres of the 9 blocks for import containers

These three tables are turned into inputs for the Matlab model by the three variables *Yard Matrix*, *Block-Gate Distances* and *Block-Quay Distances* respectively.

7.1.2 Assumptions Regarding the Resources

Important assumptions regarding the resources of the yard have already been made in Chapter 4:

- The 9 blocks for import containers are served by 3 reach-stackers
- Each reach-stacker is dedicated to a group of blocks that are located close to each other
- A reach-stacker might be required to serve a second group of blocks when another one of the three reach-stackers is out of work
- Those three reach-stackers are dedicated to the allocation phase of incoming containers exclusively. The retrieval phase is controlled by other reach-stackers, which have not been modelled with the same level of detail of the previous three. The two phases are supposed to not overlap so that there is no clashing between operations.

Other important considerations regard the performances of the yard resources:

- *Vehicle Speed* = 5 km/h. It is the speed at which internal trucks are allowed to travel in the yard.
- *Normal Stacking Time* = 2 minutes. It is the time needed to stack a container by the reach-stacker, picking it up from an internal truck and placing it on a stack.

- *Stacking Time during Event 5* = 4 minutes. It is the time needed to stack a container by a reach-stacker that is serving two groups of blocks since one of the other two is out of work. It is the double of *Normal Stacking Time.*
- *Unloading Time* = 2 minutes. It is the time needed by a Quay Crane to unload a container from a vessel and place it on an internal truck.

7.1.3 Assumptions Regarding the Time Interval and the Number of Containers

The *Time Unit* of the time-loop of the model is the minute: therefore, each run of the loop corresponds to one minute. The extent of the simulation period is 80,000 minutes, which roughly corresponds to 55 days.

The total number of containers that arrive in the yard and require to be allocated using fuzzy logic principles is 3591. The first 2000 are taken arbitrarily from a database that contains unprocessed and raw data of all the import containers that have passed through the TPA over the course of 3 years: they are not provided with a dwell time prediction and their real dwell time is calculated from their arrival and exit dates. They are used to initialise (or “warm up”) the yard, which means filling the initially empty yard up until the desired overall utilisation value, and thus are not included in the performance evaluation. As mentioned in section 4.4, the remaining 1591 containers are derived from the database developed by Maldonado et al. (2019), which contains all the data shown in Table 4.4, including dwell time predictions. The members of this second group can be called “evaluation period” containers.

7.1.4 Vessel Arrival Scheduling

Vessel Arrival Scheduling is important because it does not only generate a time of arrival for all the 3591 incoming containers, splitting them between the arriving ships, but it also plays a pivotal role in controlling the overall utilisation of the yard. As a matter of fact, calibrating the number of vessel arrivals and the time interval between them, it is possible to maintain yard utilisation above a required level for a certain number of days. In this case, the aim is to keep the overall utilisation above 70% for the longest possible time, so that the largest possible number of containers can be allocated under stressful conditions, when performances are more meaningful. After a few trials, the final configuration of the parameters for the Vessel Arrival Scheduling procedure of the model has been determined as following:

$$Number\ of\ Vessel\ Arrivals = 16$$

$$Arrival\ Weights = [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0.93\ 1\ 1\ 1\ 1\ 1\ 1\ 1]$$

$$Time\ Interval\ between\ two\ consecutive\ Arrivals = 4000\ minutes$$

$$Interval\ Weights = [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0.8\ 1.25\ 1.25\ 1.25\ 1.25\ 1.70\ 1]$$

The results of the scheduling process are summarised in Table 7.4:

Number of Arrival	Containers loaded by each vessel	Time of Arrival of the Vessel (minute)
1	224	0
2	224	4,000
3	224	8,000
4	224	12,000
5	224	16,000
6	224	20,000
7	224	24,000
8	224	28,000
9	208	32,000
10	**224**	**35,200**
11	**224**	**40,200**
12	**224**	**45,200**
13	**224**	**50,200**
14	**224**	**55,200**
15	**224**	**62,000**
16	**247**	**66,000**
	Total Number of containers = 3591	

Table 7.4 A summary of the results of Vessel Arrival Scheduling. The arrivals of vessels loading "evaluation period" containers are highlighted in yellow

The 3591 containers have been split almost equally amongst the 16 ship arrivals: the first nine vessels are carrying warming up containers exclusively while from the 10th arrival onwards, the unloaded containers belong to the dwell time prediction database. Given an *Unloading Time* = 2 minutes, the time needed to unload an entire vessel is 448 minutes (except for vessel 9, 416 minutes, and vessel 16, 494 minutes). The interval between two consecutive arrivals is 4000 minutes. However, it has been increased from the 10th arrivals onwards, in order to avoid reaching the full capacity of the yard, given the concurrent high utilisation.

The resulting profile of the overall yard utilisation over time is shown in Figure 7.1. The period of time when overall yard utilisation is above 70% goes from day 16 to day 50, for a total of 34 days. In this way, all the "evaluation period" containers, which are unloaded at the terminal between day 24 and day 46 (over a period of around 22 days), arrive in a yard that is under stressful conditions.

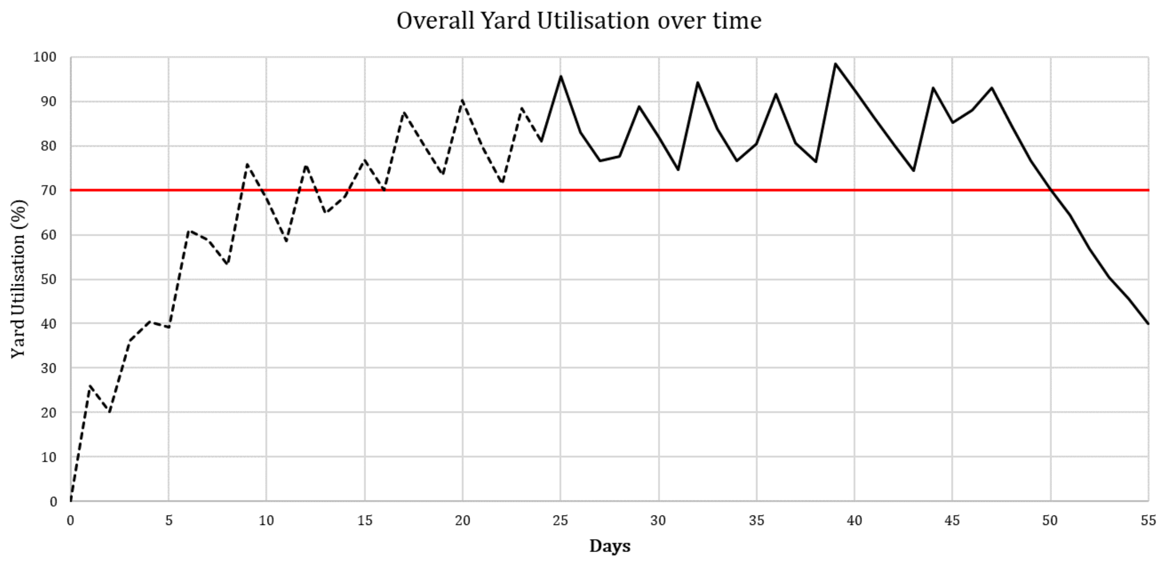


Figure 7.1 A graph that shows the evolution of the overall yard utilisation over time with the proposed parameters. The red line shows the required level of utilisation, 70%. The dashed line represents the warming up period and it turns into a solid black line after the arrival of the first vessel loading "evaluation period" containers

7.2 Training Phase

After the definition of the principal assumptions, it is now possible to address the main objective of this work: the creation of a decision support system, based on Fuzzy Logic, that is able to react dynamically to events that have an impact on the operations in the yard. The traditional stacking strategies maintain the same policy constantly, without considering disturbances. The basic idea is to create a dynamic stacking system by assembling together the best performing Fuzzy Systems for each kind of Event. The training phase therefore corresponds to all the testing rounds devoted to find the best Fuzzy System for every Event type.

7.2.1 Selection of the Fuzzy Systems

Not all the 21 Fuzzy Systems, defined in Chapter 5 as a result of the combination of the proposed Criteria, have actually been tested during the training phase. As a matter of fact, all the Fuzzy Systems that employ S2 as a Stack Assignment policy, using only the current Stack Height and avoiding the use of the potential number of Rehandles, have been discarded: Borgman et al. (2010) proved that even using partial or imprecise information on the dwell time to allocate containers is beneficial in terms of performances; this was confirmed by Maldonado et al. (2019), with the added virtue of doing it using the same dwell time prediction and the same container terminal of this work. Moreover, to reduce the potential training combinations and therefore the required computational time, all the Fuzzy Systems with S12 as a Stack Assignment policy, using both Rehandles and Stack Height, have not been considered. The remaining 7 Fuzzy Systems, selected to be evaluated in the training phase, are the ones that use S1 to allocate incoming containers on a stack: B1 S1, B2 S1, B12 S1, B23 S1, B13 S1 and B123 S1. They are shown in Table 7.5.

		Stack Assignment		
		S1	**S2**	**S12**
Block Assignment	**B1**	***B1 S1***	B1 S2	B1 S12
	B2	***B2 S1***	B2 S2	B2 S12
	B3	***B3 S1***	B3 S2	B3 S12
	B12	***B12 S1***	B12 S2	B12 S12
	B23	***B23 S1***	B23 S2	B23 S12
	B13	***B13 S1***	B13 S2	B13 S12
	B123	***B123 S1***	B123 S2	B123 S12

Table 7.5 Summary of the selected Fuzzy Systems for the training phase (highlighted in yellow)

7.2.2 Selection of the Events

To avoid having an excessively time-consuming number of combinations, a selection of the events to be included in the training phase has also been conducted. The rationale behind the choice is to have one Event type for each one of the three classes of Low Level Disturbances. The elected events are: Event 1 (Blocking a Block – Yard-related Event), Event 2 (Traffic Jam – Container-related Event) and Event 5 (Crane Breakdown – Resource-related Event). The two remaining, discarded, types of Events are: Event 3 (Drivers' Strike) and Event 4 (Late Arrival of an External Truck), both belonging to the class of Container-related events. The selection is reported in Table 7.6.

	Class of Low-Level Disturbances		
	Yard-related Events	**Container-related Events**	**Resource-related Events**
Event Type	***Event 1 – Blocking a Block***	***Event 2 – Traffic Jam***	***Event 5 – Crane Breakdown***
		Event 3 – Drivers' Strike	
		Event 4 – Late Arrival of an External Truck	

Table 7.6 Summary of the selected Events for the training phase (highlighted in yellow)

7.2.2.1 Characteristics of the Events

In order to understand the impact of the single events on the yard and to determine which one of the selected Fuzzy Systems is the best for every Event type, it is necessary to modify the construction of the *Event Matrix*: instead of creating a list of events belonging to different Event types, the event generation process shall now generate an *Event Matrix* where only events of one single Event type are allowed to happen at specific times. The characteristics of the *Event Matrices* generated for the single Event types are listed below:

- Event 1 (Blocking a block). *Event Duration*: 235 minutes. Every time the event happens, 1 out of the 9 blocks is randomly selected to be shut down. Not more than one block is allowed to be closed at a time in order to avoid reaching the full capacity of the yard, given the high overall utilisation. Two consecutive events of Type 1 are allowed to happen exclusively during each one of the 7 vessel arrivals that unload "evaluation period" containers, for a total of 14 events (e.g. during vessel arrival number 10, the first container is unloaded at minute 35200 and the last one at minute 35646. Two Events of Type 1 happen in the meantime: one starting at minute 35160 and ending at minute 35395 and the other one starting at 35400 and ending at 35635. In this way, the unloading process is affected by an Event 1 during almost all of its length). This is done in order to maximise the impact of the event: having an Event of Type 1 happening outside the time window of a vessel arrival has no effect on the yard, since no containers need to be allocated and the closure of the block do not interfere with any decision-making process. The time moments when the events are simulated are fixed and shown in Table 7.7.
- Event 2 (Traffic Jam). *Event Duration* = 240 minutes. *Jam Delay* = 240 minutes. In this case there is no need to simulate the events only during the unloading period of the vessels: the delaying of the external trucks caused by a traffic jam impacts retrieval operations, altering the dwell times of the containers, not the allocation process. Therefore, events of Type 2 can be distributed throughout all the "evaluation period": two traffic jams are simulated every day (one happening between 7:00am and 11:00am and the other one between 3:00pm and 7:00pm) for 25 days, starting from day 26, for a total of 50 events; the starting time of each traffic jam is generated randomly within the respective proposed time window.
- Event 5 (Crane Breakdown). *Event Duration* = 235 minutes. Every time the event happens, 1 out of the 3 reach-stackers that serve the allocation phase at the blocks is

selected to be considered out of work. Not more than one reach-stacker is assumed to break down at any given time. The same considerations stated for Event 1 are valid for Event 5: the breakdown of a reach-stacker only has a measurable impact on the yard if it happens during the berthing time of a vessel, when containers are being unloaded and need to be allocated in the blocks. Therefore, the same event generation strategy of Event 1 is applied, with Table 7.7 being valid also for Event 5.

Number of Vessel Arrival	**Duration of unloading period** ***(start-end minute)***	**Events Duration** ***(start-end minute)***
10	35,200 – 35,646	35,160 – 35,395
		35,400 – 35,635
11	40,200 – 40,646	40,200 – 40,435
		40,440 – 40,675
12	45,200 – 45,646	45,120 – 45,415
		45,420 – 45,655
13	50,200 – 50,646	50,160 – 50,395
		50,400 – 50,635
14	55,200 – 55,646	55,200 – 55,435
		55,440 – 55,675
15	62,000 – 62,446	61,980 – 62,215
		62,220 – 62,455
16	66,000 – 66,492	66,000 – 66,235
		66,240 – 66,475

Table 7.7 A table that shows the distribution of Events 1 and 5 over time and their correspondence with the unloading time of the vessels

7.2.3 Final Training Outline

In the final training outline, the set of 7 selected Fuzzy Systems are tested:

- 30 times for Event 1. In each one of these times, also called Runs in the training phase, all the 7 Fuzzy Systems are tested using the same *Event Matrix*, which is indicated by one or more capital letters (Run A, Run B, Run C etc.). Each *Event Matrix* is different from the other thanks to the randomness in the selection of the blocked blocks.
- 30 times for Event 2. In each one of these Runs, all the 7 Fuzzy Systems are tested using the same *Event Matrix*, indicated by one or more capital letters. Each *Event*

Matrix is different from the other thanks to the randomness in the generation of the starting time of the traffic jams within their time windows.

- 30 times for Event 5. In each one of these Runs, all the 7 Fuzzy Systems are tested using the same *Event Matrix*, indicated by one or more capital letters. Each *Event Matrix* is different from the other thanks to the randomness in the selection of the out-of-work crane.
- 1 time in a No Events scenario. In this case, the *Event Matrix* has been modified so that no events happen during the entire simulation period. One run is sufficient because, without the randomness of the event generation, the Matlab model works in a deterministic way.

The number of Runs for each event, 30, has been designed in order to have statistically significant results. Testing all the 7 Fuzzy Systems for 30 times for all the 3 events and 1 time for the No Events scenario results in running the 55-days simulation for 631 times. At the end of this process, the best Fuzzy System for each Event is determined through different strategies. The best Fuzzy Systems for each Event are then assembled to construct the Dynamic Fuzzy Systems, which are able to react dynamically to the disturbances that have an impact on the container terminal.

7.2.4 Performance Evaluation

How is it possible to find a "best" Fuzzy System? And what does "best" Fuzzy System mean? To answer these questions, a set of specific metrics that evaluate the performances of the Fuzzy Systems has to be introduced, bearing in mind the results of the Performance Indicators classification shown in section 3.3.

7.2.4.1 Key Performance Indicators (KPIs)

Four Key Performance Indicators have been defined. With the aim of measuring performances when the terminal is under stress, the KPIs are applied only to the containers that have their *Arrival Times* and *Real Times of Departure* comprised between minute 35200 (the arrival time of vessel number 10) and minute 72000 (the end of day 50): in this way, performances are assessed with an overall yard utilisation above 70%. From now on, the terms “evaluation period” and “evaluation period containers” are referred to the time interval that goes from 35200 to 72000. The four defined KPIs are listed below:

- Total Congestion: it is defined as the total sum of the congestion level experienced by each evaluation period container after it has been assigned to a specific block. Congestion for container i is defined in the same way as for the corresponding Criterion: the queue of containers heading towards a given block at the moment in which container i is unloaded from a vessel; the required data are listed in the specific column in *Container Database* matrix. In order to compare results with a different number of evaluation period containers, Average Congestion is defined as:

$$AVG\ Congestion = \frac{\sum_i Congestion\ level\ of\ container\ i\ during\ allocation}{Total\ number\ of\ evaluation\ period\ containers} \qquad (7.1)$$

  It corresponds to the average queue of containers that each evaluation period container finds while being allocated to its block of destination. A small value of AVG Congestion indicates that the yard is not congested and the stacking operations flow smoothly.

- Total Rehandles: it is defined as the total sum of the number of rehandles caused by each one of the evaluation period containers during retrieval operations. Rehandles are defined in the same way as for the corresponding Criterion: in this case, however the inefficient moves needed to reclaim a leaving container are effective and not predicted. In order to compare results with a different number of evaluation period containers, Average Rehandles is defined as:

$$AVG\ Rehandles = \frac{\sum_i Rehandles\ needed\ during\ retrieval\ of\ container\ i}{Total\ number\ of\ evaluation\ period\ containers} \qquad (7.2)$$

  It corresponds to the average number of rehandles caused by each evaluation period container while leaving the yard. Reducing the number of rehandles is beneficial for a container terminal since it allows to speed up retrieval operations and causes less wear of the resources devoted to the pick-up of the containers.
- Total Distance: it is defined as the sum of all the Block-Gate distances travelled by the external trucks while leaving the yard after having retrieved an evaluation period container at the end of its dwell time. The required data are listed in the specific column in *Container Database* matrix. Block-Gate Distance is defined in the same

way as for the Criterion of the same name. In order to compare results with a different number of evaluation period containers, Average Distance is defined as:

$$AVG\ Distance = \frac{\sum_i Block - Gate\ Distance\ travelled\ after\ the\ retrieval\ of\ container\ i}{Total\ number\ of\ evaluation\ period\ containers} \quad (7.3)$$

As stated by Ries et al. (2014), Total Distance is an indication for a potential risk of congestion and inefficiency of the stacking policy: the lower it is, the most beneficial it is for the terminal.

- Average of the Standard Deviation of Block Utilisation Per Day: it is a measure of how the containers are evenly distributed across the 9 blocks. At the end of each day (every 1440 minutes), the Utilisation of every one of the 9 blocks is recorded as a number that goes from 0 to 100 (a Utilisation of Block 2 of 87.15 means that Block 2 is full at 87.15% of its capacity). The standard deviation of the 9 levels of Utilisation is then calculated. This procedure is repeated each day from day 25 to day 50 included: the result is a list of 26 standard deviations of Block Utilisation. To obtain one single number that is able to capture the validity of a stacking strategy in terms of even distribution of the containers across the available blocks, those 26 standard deviations are averaged: the result is called Average of the Standard Deviation of Block Utilisation Per Day or, in short, AVG STD Per Day. Having an even distribution of the containers, signalled by a low value of AVG STD Per Day, is a preferred condition by port managers.

It is interesting to note that, throughout the four KPIs, all the three classes defined in the Performance Indicator Classification are represented: Total Distance and AVG STD Per Day are Yard-related, Total Rehandles is Container-related and Total Congestion is Resource-related. Even more important, the Matching Principle presented in section 3.5 is fully respected: as it appears from the definition of the KPIs, each of the four Criteria that make up the 7 selected Fuzzy Systems has a corresponding KPI. This constitutes another argument for the choice of said Fuzzy Systems. A summary of the KPIs is shown in Table 7.8.

Number	**KPI**	**Class**	**Correlated Criterion**
1	*Total Congestion*	Resource-related	Congestion
2	*Total Rehandles*	Container-related	Rehandles
3	*Total Distance*	Yard-related	Block-Gate Distance
4	*AVG STD Per Day*	Yard-related	Block Utilisation

Table 7.8 A summary of the defined KPIs

Another performance metric has been defined but it has not been included among the four main KPIs: Congestion at Retrieval. It has not been considered in the development of the Dynamic Systems but it has been recorded anyway during the training phase.

- Congestion at Retrieval: if a container is leaving the yard at a given minute, it is defined as the number of other containers being retrieved from the same block within the successive 5 minutes. For example, let's consider a container coded with the letter A, stacked at block 7 and whose *Real Time of Departure* is set at minute 60. The number of containers being retrieved between minute 60 and 65 (within 5 minutes of the departure of the mentioned container) is 6. Out of these 6, 2 are stacked at block 7. The value of Congestion at Retrieval associated to container A is then 2. If no containers are leaving in the successive 5 minutes or if they are located in different blocks, Congestion at Retrieval is set at 0.

7.2.4.2 The Ranking Approach

The four KPIs have been combined into one single index: this metric is able to evaluate the overall validity of a Fuzzy System considering Congestion, Rehandles, Distance and AVG STD Per Day at the same time. This is a rather novel approach in the literature related to the container allocation problem since only Rekik et al. (2018) proposed a single metric that incorporates more than one KPI not expressed in units of time.

The main obstacle to the creation of the index is the fact that the four KPIs have different units of measurement (Congestion is expressed through a queue of containers, Rehandles through a number of inefficient moves, Distance is given in metres and AVG STD Per Day is an average of multiple standard deviations) and different magnitudes. Using a min-max normalisation combined with a Utopia Point approach was attempted but eventually discarded, due to the excessive rescaling of the absolute values that it produced, with the risk of losing the physical sense of the results. A solution has finally been provided by the ranking approach: it uses the average forms of the KPIs (AVG Congestion, AVG Rehandles, AVG Distance and AVG STD Per Day, which is already an average) and it is applied to every Run of each event in the training phase (each Run includes the results, expressed via the four KPIs, of the application of the 7 selected Fuzzy Systems under the impact of the same *Event Matrix*). The ranking approach works in this way:

- Let's consider one KPI. The numerical values of the 7 Fuzzy Systems for that KPI are ranked from best to worst. Each value is then expressed through its rank, which can go from 1 (the best result) to 7 (the worst result).
- The procedure is repeated for all the KPIs.
- For each Fuzzy System, the average (Average Rank or AVG Rank) and standard deviation (STD Rank) of the ranks of the four KPIs are calculated. The Fuzzy System that has the lowest Average Rank is the best performing one. In case two or more Fuzzy Systems have the same Average Rank, the tie is broken using the standard deviation: the Fuzzy System with the lowest Average Rank and standard deviation of the ranks is considered the best.

The application of the ranking approach is shown in Table 7.9. The results refer to Run A for Event 1, which means that all the 7 Fuzzy System, indicated in the first column, have been tested with the same set of events of Type 1.

Fuzzy System	AVG Congestion	AVG Rehandles	AVG Distance	AVG STD	Rank Congestion	Rank Rehandles	Rank Distance	Rank AVG STD	AVG Rank	STD Rank
B1 S1	3,165496	1,105316	206,0544	11,61533	7	7	4	7	6,25	1,5
B2 S1	0,564694	0,976931	209,395	4,73509	3	1	7	1	3	2,82842
B3 S1	**0,414243**	**1,002006**	**202,9223**	**7,206283**	**2**	**2**	**1**	**3**	**2**	**0,81649**
B12 S1	2,007021	1,087262	205,8512	9,605078	6	6	3	5	5	1,41421
B13 S1	1,633902	1,058175	205,1182	10,72316	5	4	2	6	4,25	1,70782
B23 S1	0,064193	1,018054	206,9073	4,937984	1	3	6	2	3	2,16024
B123 S1	0,694082	1,077232	206,611	8,131764	4	5	5	4	4,5	0,57735

Table 7.9 Application of the Ranking Approach

The table shows that, for example, the Fuzzy System B1 S1 is the worst in terms of Congestion, Rehandles and AVG STD Per Day, and is the fourth best in terms of Distance. The result of the ranking approach is that the Fuzzy System B3 S1 is the best performing of the lot, since its AVG Rank of 2 is the lowest among the tested Fuzzy Systems.

7.2.5 Definition of the Best Fuzzy System for Each Event

The ranking approach described above only allows to find the best Fuzzy System in one Run of the training phase. Therefore, in the interest of finding the best Fuzzy System for each simulated Event, a new question arises: how is it possible to define a clear winner out of the 30 Runs for each Event? Three methods are proposed:

1. Statistical Approach. This method is the most grounded from a statistical point of view. The Average Rank of each Fuzzy System for all the 30 runs for Event Type is subjected to a univariate analysis of variance (also called univariate ANOVA) using the statistical software SPSS. The software first calculates the mean and standard deviation of Average Rank across the 30 Runs for each one of the 7 Fuzzy Systems. Then, a test, called Bonferroni post hoc test, is performed in order to understand whether there is a statistically significant difference between the previously calculated means. To find the best Fuzzy System, the following procedure is adopted: the Fuzzy System with the lowest mean of Average Rank is chosen as the candidate; then, the candidate is compared with all the remaining Fuzzy Systems in the results of the post-hoc test; if there is a statistically significant difference with all the other policies, the candidate is considered the best Fuzzy System for the Event; if there is no statistically significant difference with another (or more) Fuzzy System, it means that adopting one stacking strategy or the other is equivalent: therefore, both Fuzzy Systems are chosen to be the best for the Event under examination.
2. Descriptive Approach. It is the most direct of the methods and, for each Event, it counts the number of times out of the 30 Runs in which a Fuzzy System has proven to be better than the others in terms of Average Rank. In case of a draw, the best system is the one that also has the lowest STD Rank. If there is a clear, highly frequent winner, it is considered as the best Fuzzy System for that event. If two (or more) Fuzzy Systems end up competing, with both having roughly the same number of "winning" runs, they can be considered jointly as the best Fuzzy Systems for the event under examination.
3. Descriptive Approach from SPSS. This method does not employ the Average Rank and the ranking approach but it uses the average form of the four KPIs. The rationale behind this method is based on the matching principle described in Chapter 3 and it can be described by the expression "rescuing the affected KPI": the three simulated events and the four KPIs share the same classes (Yard-related, Container-related and Resource-related); when an Event belonging to a certain class happens, it is assumed that the KPIs of the same class are the most likely to be affected, since they rely on the same elements (for example, if Event 5 happens, which is Resource-related, the most affected KPI is assumed to be Congestion, which is the Resource-related KPI). Therefore, the best Fuzzy System for each Event is considered to be the Fuzzy System that provides the best results in terms of the affected KPI (following the

example, if Fuzzy System B23 S1 gives the lowest values of Congestion when Event 5 is happening, B23 S1 is chosen as the best Fuzzy System for Event 5). In this way, each best Fuzzy System can be seen as the allocation strategy that is able to address an eventual disturbance in the best fashion. To explore in detail the relationship between Events and KPIs, a multivariate statistical analysis of a database containing the results of the four KPIs (Average Congestion, Average Rehandles, Average Distance and AVG STD Per Day) for each Fuzzy System for all the 30 runs of every Event is conducted through the statistical software SPSS. Among the results, there are a series of graphs that show the relations between KPIs, Fuzzy Systems and Events: they are used as a helping tool to determine the best Fuzzy System for each Event.

In order to experiment the possibilities offered by a reactive stacking strategy, a different method of generating a Dynamic System is proposed: instead of combining the four KPIs, this newly defined Dynamic System is based exclusively on Average Rehandles. There is no need to introduce new Runs, since the results of training phase already include the KPI AVG Rehandles for each Fuzzy System. In this case, since only one KPI is considered, the ranking approach is not needed to find the best performing Fuzzy System of each Run. To determine which Fuzzy System is the best for each Event (across the respective 30 Runs), the same three methods described above are applied:

1. Statistical Approach. The statistical analysis is performed considering AVG Rehandles instead of AVG Rank.
2. Descriptive Approach. In this case, the best Fuzzy System for each Run is the one with the lowest AVG Rehandles value.
3. Descriptive Approach from SPSS. In this case, the KPI to be “rescued” is always AVG Rehandles, without changing according to the events. In this way, the matching principle is not valid anymore, but the method is still considered as a useful option to define the best Fuzzy System for each Event.

7.3 Training Phase Results

In this section, the results of the training phase are presented. They are split in two parts, one where the focus is on the combination of the four KPIs and the other one where the focus is exclusively on Rehandles. In both cases, the results are listed following the order of the three proposed approaches to define the best Fuzzy System for each Event. The full lists of training results can be provided upon request.

7.3.1 Training Results for the Combination of KPIs

7.3.1.1 Statistical Approach

For *Event 1*, the descriptive statistics of the Fuzzy Systems are shown in Table 7.10.

Descriptive Statistics

Dependent variable: AVG Rank

Fuzzy System	Mean	Std. Deviation	Runs
B1 S1	5,8500	,42345	30
B12 S1	4,6583	,41254	30
B123 S1	3,4583	,58753	30
B13 S1	4,6167	,33305	30
B2 S1	2,8833	,31984	30
B23 S1	3,1083	,37534	30
B3 S1	3,4250	,44601	30
Total	4,0000	1,07883	210

Table 7.10 Descriptive Statistics of the 7 Fuzzy Systems for Event 1

The Fuzzy System with the lowest mean of AVG Rank is B2 S1, which is the candidate to be the best Fuzzy System for Event 1. However, the post hoc test shows that there is no statistically significant difference between the means of B2 S1 and B23 S1. This is highlighted in Table 7.11. The fifth column shows the significance level: if it is above 0.05 it means that there is no significant difference between the means of the compared Fuzzy Systems. In this case, the significance level for the comparison between B2 S1 and B23 S1 is above the limit, at 0.846.

Multiple Comparison

Dependent Variable: AVG Rank

Bonferroni

(I)	(J)				95% Confidence Interval	
Fuzzy_System	Fuzzy_System	Mean Difference (I-J)	Std. Error	Sign.	Lower Bound	Upper Bound
B2 S1	B1 S1	-2,9667*	,10901	,000	-3,3021	-2,6313
	B12 S1	-1,7750*	,10901	,000	-2,1104	-1,4396
	B123 S1	-,5750*	,10901	,000	-,9104	-,2396
	B13 S1	-1,7333*	,10901	,000	-2,0687	-1,3979
	B23 S1	-,2250	,10901	,846	-,5604	,1104
	B3 S1	-,5417*	,10901	,000	-,8771	-,2063

Table 7.11 Post hoc test results for B2 S1 for Event 1

The results of the statistical analysis yield that the best Fuzzy System for Event 1 under a Statistical approach is either B2 S1 or B23 S1.

For *Event 2*, the descriptive statistics of the Fuzzy Systems are shown in Table 7.12.

Descriptive Statistics

Dependent variable: AVG Rank

Fuzzy System	Mean	Std. Deviation	Runs
B1 S1	6,1167	,17036	30
B12 S1	4,5417	,31543	30
B123 S1	2,9250	,40016	30
B13 S1	4,7833	,31303	30
B2 S1	2,6750	,19859	30
B23 S1	3,0833	,31026	30
B3 S1	3,8583	,21459	30
Total	3,9976	1,18048	210

Table 7.12 Descriptive Statistics of the 7 Fuzzy Systems for Event 2

The Fuzzy System with the lowest mean of AVG Rank is B2 S1, which is the candidate to be the best Fuzzy System for Event 2. From the post hoc test, the mean of B2 S1 shows a statistically significant difference with the means of all the other Fuzzy Systems. This is highlighted in Table 7.13.

Multiple Comparison

Dependent Variable: AVG Rank

Bonferroni

(I)	(J)				95% Confidence Interval	
Fuzzy_System	Fuzzy_System	Mean Difference (I-J)	Std. Error	Sign.	Lower Bound	Upper Bound
B2 S1	B1 S1	-3,4417*	,07357	,000	-3,6680	-3,2153
	B12 S1	-1,8667*	,07357	,000	-2,0930	-1,6403
	B123 S1	-,2500*	,07357	,017	-,4764	-,0236
	B13 S1	-2,1083*	,07357	,000	-2,3347	-1,8820
	B23 S1	-,4083*	,07357	,000	-,6347	-,1820
	B3 S1	-1,1833*	,07357	,000	-1,4097	-,9570

Table 7.13 Post hoc test results for B2 S1 for Event 2

The results of the statistical analysis yield that the best Fuzzy System for Event 2 under a Statistical approach is B2 S1 exclusively.

For *Event 5*, the descriptive statistics of the Fuzzy Systems are shown in Table 7.14. The Fuzzy System with the lowest mean of AVG Rank is B2 S1, which is the candidate to be the best Fuzzy System for Event 5. From the post hoc test, the mean of B2 S1 shows a statistically significant difference with the means of all the other Fuzzy Systems. This is highlighted in Table 7.15.

Descriptive Statistics

Dependent variable: AVG Rank

Fuzzy System	Mean	Std. Deviation	Runs
B1 S1	5,8583	,21459	30
B12 S1	4,4917	,26655	30
B123 S1	3,0583	,46277	30
B13 S1	4,8750	,40869	30
B2 S1	2,5000	,23672	30
B23 S1	3,0583	,24286	30
B3 S1	4,1417	,38665	30
Total	3,9976	1,15176	210

Table 7.14 Descriptive Statistics of the 7 Fuzzy Systems for Event 5

Multiple Comparison

Dependent Variable: AVG Rank

Bonferroni

(I)	(J)				95% Confidence Interval	
Fuzzy_System	Fuzzy_System	Mean Difference (I-J)	Std. Error	Sign.	Lower Bound	Upper Bound
B2 S1	B1 S1	-3,3583*	,08523	,000	-3,6206	-3,0961
	B12 S1	-1,9917*	,08523	,000	-2,2539	-1,7294
	B123 S1	-,5583*	,08523	,000	-,8206	-,2961
	B13 S1	-2,3750*	,08523	,000	-2,6372	-2,1128
	B23 S1	-,5583*	,08523	,000	-,8206	-,2961
	B3 S1	-1,6417*	,08523	,000	-1,9039	-1,3794

Table 7.15 Post hoc test results for B2 S1 for Event 5

The results of the statistical analysis yield that the best Fuzzy System for Event 5 under a Statistical approach is B2 S1 exclusively.

Finally, for the *No Event* scenario, the only Run shows that two Fuzzy Systems have the same AVG Rank: B2 S1 and B23 S1. This is highlighted in Table 7.16. The statistical analysis postulates that there is no significant difference between those two strategies, so both of them are considered as the best Fuzzy Systems for a *No Event* situation.

Fuzzy System	**Rank Congestion**	**Rank Rehandles**	**Rank Distance**	**Rank AVG STD**	**AVG Rank**	**STD Rank**
B1 S1	7	7	4	7	6,25	1,5
B2 S1	2	2	6	1	2,75	2,217356
B3 S1	3	3	5	4	3,75	0,957427
B12 S1	6	6	3	5	5	1,414214
B13 S1	5	5	2	6	4,5	1,732051
B23 S1	1	1	7	2	2,75	2,872281
B123 S1	4	4	1	3	3	1,414214

Table 7.16 AVG Rank and STD Rank for the 7 Fuzzy Systems under a No Event scenario

7.3.1.2 Descriptive Approach

For *Event 1*, the number of times (called Frequency of Victories) out of the 30 Runs in which each Fuzzy System has proven to be best in terms of Average Rank are listed in Table 7.17.

Fuzzy System	**Frequency of Victories**
B1 S1	0
B2 S1	18
B3 S1	2
B12 S1	0
B13 S1	0
B23 S1	6
B123 S1	4
Total Runs	**30**

Table 7.17 Frequency of Victories for Event 1

The Fuzzy System that has the highest Frequency of Victories is B2 S1 with 18. At the same time, there are other 3 Fuzzy Systems that have scored more than a victory, with two of them (B23 S1 and B123 S1) having a considerable frequency (6 and 4 respectively). From a purely descriptive point of view and with an arbitrary choice, B2 S1 is considered the best Fuzzy System for Event 1 alongside B23 S1: the 6 victories of B23 S1 have been regarded as not negligible. On the contrary, B123 S1 has been excluded.

For *Event 2*, the Frequency of Victories in terms of Average Rank for each Fuzzy System out of the 30 Runs are listed in Table 7.18.

Fuzzy System	Frequency of Victories
B1 S1	0
B2 S1	13
B3 S1	0
B12 S1	0
B13 S1	0
B23 S1	5
B123 S1	12
Total Runs	**30**

Table 7.28 Frequency of Victories for Event 2

In this case, there are two clear and highly frequent winners: B2 S1, with 13 victories, and B123 S1, with 12 victories. Therefore, the two Fuzzy Systems, B2 S1 and B123 S1, are considered the best Fuzzy Systems for Event 2.

For *Event 5*, the Frequency of Victories in terms of Average Rank for each Fuzzy System out of the 30 Runs are listed in Table 7.19.

Fuzzy System	Frequency of Victories
B1 S1	0
B2 S1	23
B3 S1	0
B12 S1	0
B13 S1	0
B23 S1	0
B123 S1	7
Total Runs	**30**

Table 7.39 Frequency of Victories for Event 5

The table shows that there are only two Fuzzy Systems with at least one victory: B2 S1, with 7, and B123 S1, with 23. In this case, the Frequency of Victories of B123 S1 is arbitrarily considered too big to be discarded. Hence, even for Event 5 the Descriptive Approach yields two best Fuzzy Systems: B2 S1 and B123 S1.

With regards to the *No Event* scenario, given the fact that there is only one Run available, the best Fuzzy System under a Descriptive Approach is defined as the strategy that has the lowest AVG Rank. Table 7.16 shows that there are two Fuzzy Systems with the lowest AVG Rank. The tie is broken using the same rationale adopted to count the Frequencies of

Victories: out of the two Fuzzy Systems with the same AVG Rank, the best one for No Event is the one with the lowest STD Rank, so B2 S1.

7.3.1.3 Descriptive Approach from SPSS

Prior to the description of the best Fuzzy Systems for the various Event Types, a brief digression on the graphical confirmation of the matching principle is needed. Four graphs are shown below in Figure 7.2, 7.3, 7.4 and 7.5. Each graph shows the mean values of a KPI calculated across the different Event Types, which populate the horizontal axis, for all the 30 Runs of the training phase. All those graphs tell that KPIs suffer the biggest variations under the effect of Events that belong to the same class (Container-related, Yard-related and Resource-related).

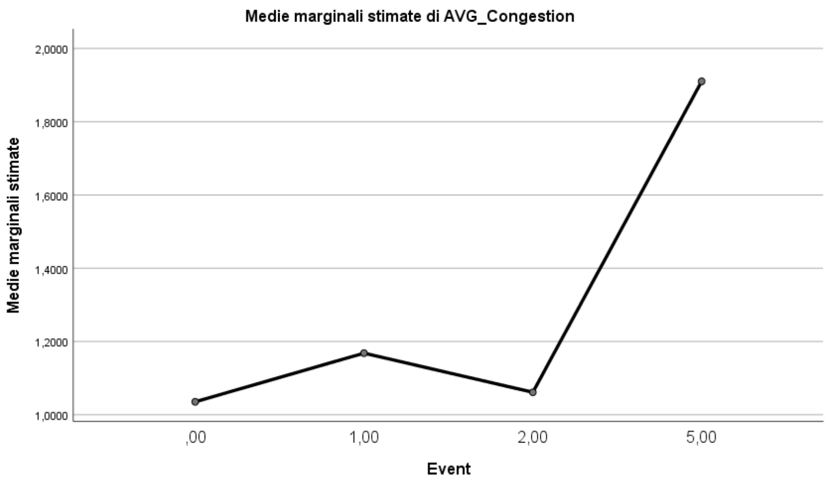


Figure 7.2 Estimated Marginal Means of AVG Congestion calculated for each Event Type. 0 means No Event

Figure 7.2 shows that the worst results in terms of AVG Congestion, a Resource-related KPI, are obtained when the yard operations are impacted by a Resource-related Event, Event 5.

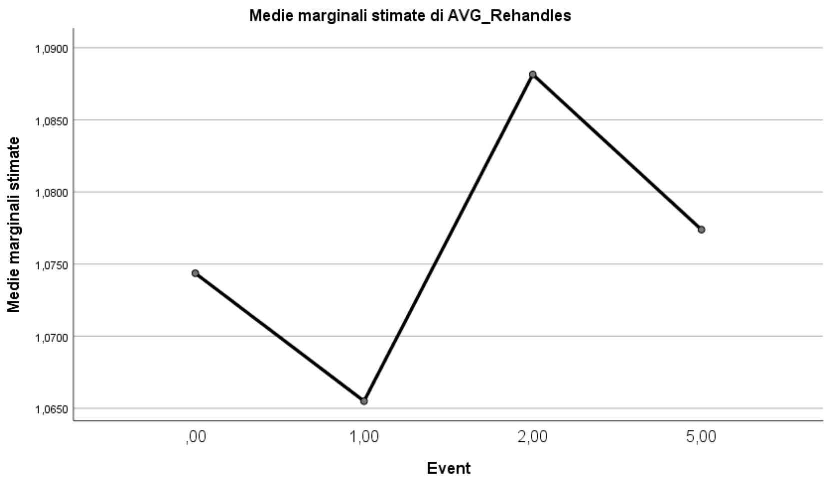


Figure 7.3 Estimated Marginal Means of AVG Rehandles calculated for each Event Type. 0 means No Event

Figure 7.3 shows that the worst results in terms of AVG Rehandles, a Container-related KPI, are obtained when the yard operations are impacted by a Container-related Event, Event 2.

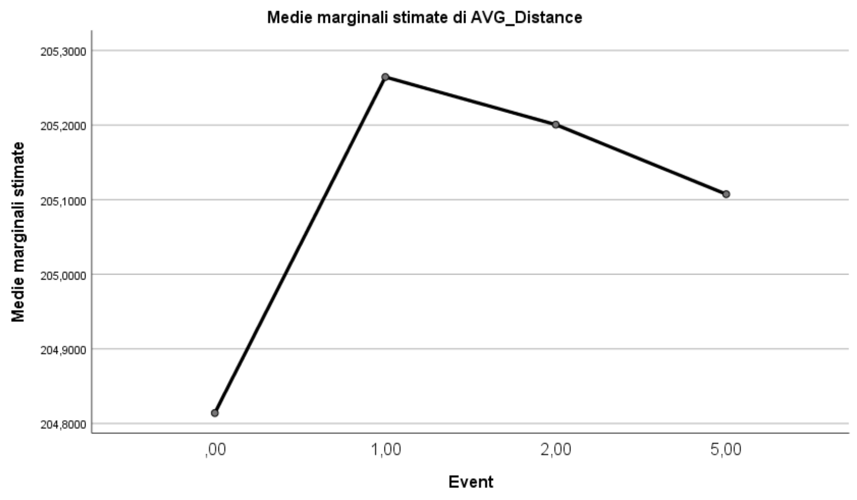


Figure 7.4 Estimated Marginal Means of AVG Distance calculated for each Event Type. 0 means No Event

Figure 7.4 shows that the worst results in terms of AVG Distance, a Yard-related KPI, are obtained when the yard operations are impacted by a Yard-related Event, Event 1.

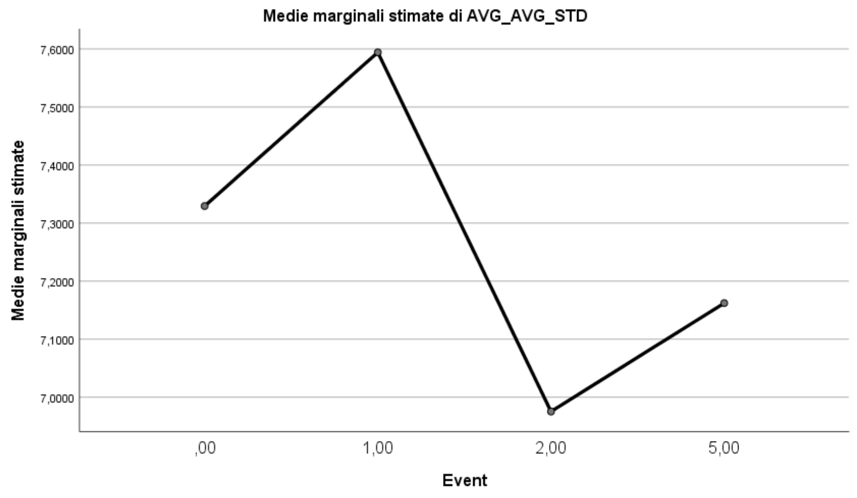


Figure 7.5 Estimated Marginal Means of AVG STD Per Day calculated for each Event Type. 0 means No Event

Figure 7.5 shows that the worst results in terms of AVG STD Per Day, a Yard-related KPI, are obtained when the yard operations are impacted by a Yard-related Event, Event 1.

It is now possible to address the issue of the best Fuzzy System according to Descriptive Approach from SPSS. When *Event 1* happens, a Yard-related Event, the most affected KPIs, which need to be rescued, are the Yard-related ones: AVG Distance and AVG STD Per Day. To examine the performances of the 7 different Fuzzy Systems, two graphs are used. They are depicted in Figure 7.6 and 7.7 and they report the estimated marginal means of both

KPIs, calculated across the 30 Runs for each Event Type and differentiated for each Fuzzy System.

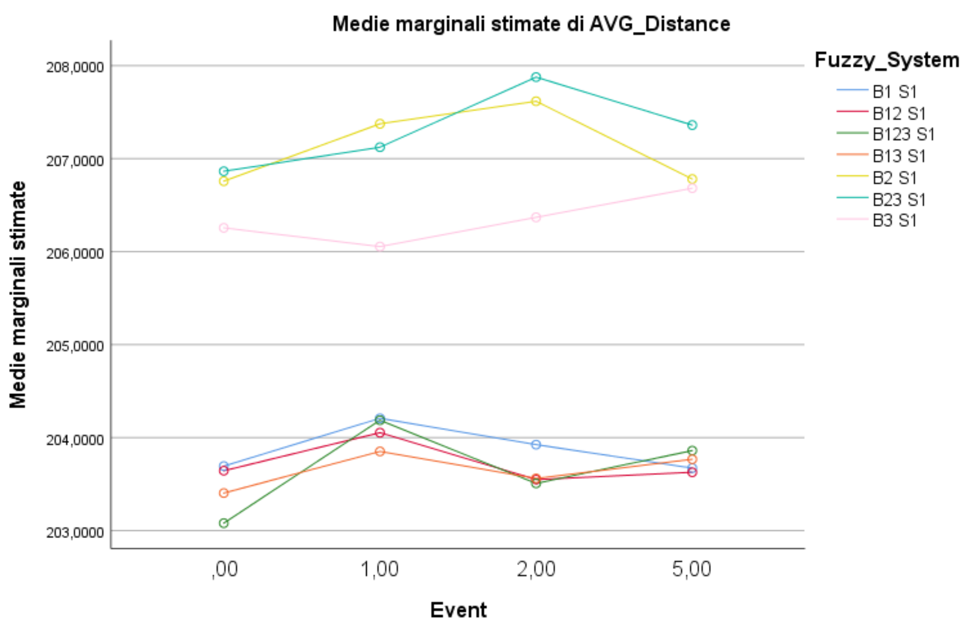


Figure 7.6 Estimated Marginal Means of AVG Distance calculated for each Event Type and differentiated for the 7 Fuzzy Systems. 0 means No Event

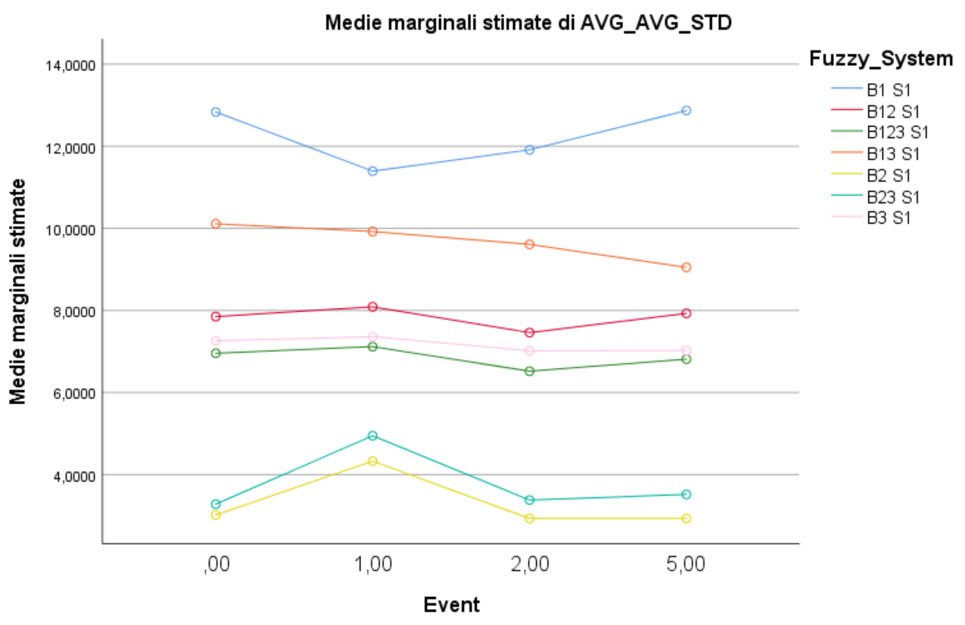


Figure 7.7 Estimated Marginal Means of AVG STD Per Day calculated for each Event Type and differentiated for the 7 Fuzzy Systems. 0 means No Event

Figure 7.7 shows that the best Fuzzy System in terms of AVG STD Per Day during an Event of Type 1 is B2 S1, represented by the yellow line. Figure 7.6 indicates that the best Fuzzy System in terms of AVG Distance during an Event of Type 1 is B13 S1, represented by the orange line. However, given the poor performance of B13 S1 with regards to AVG STD Per

Day (the second worst Fuzzy System) and the fact that B2 S1 results to be the best Fuzzy System for Event 2 and No Event (see next sections), with the aim of creating a Dynamic System that is able to compete and outperform a static allocation strategy, the two best Fuzzy Systems for Event 1 are chosen as such: B2 S1 and B123 S1. B123 S1, indicated by the green line in both figures, is the third best Fuzzy System for both KPIs.

When *Event 2* happens, a Container-related Event, the most affected KPI, which needs to be rescued, is a Container-related one: AVG Rehandles. The estimated marginal means of AVG Rehandles, calculated across the 30 Runs for each Event Type and differentiated for each Fuzzy System, are shown in Figure 7.8.

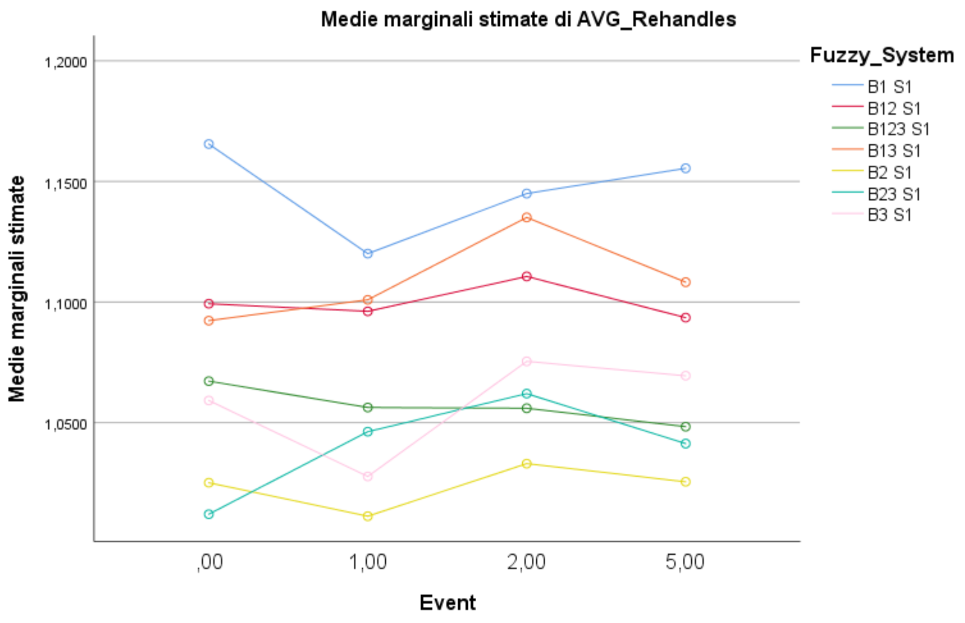


Figure 7.8 Estimated Marginal Means of AVG Rehandles calculated for each Event Type and differentiated for the 7 Fuzzy Systems. 0 means No Event

Figure 7.8 shows that the best Fuzzy System in terms of AVG Rehandles during an Event of Type 2 is B2 S1, represented by the yellow line.

When *Event 5* happens, a Resource-related Event, the most affected KPI, which needs to be rescued, is a Resource-related one: AVG Congestion. The estimated marginal means of AVG Congestion, calculated across the 30 Runs for each Event Type and differentiated for each Fuzzy System, are shown in Figure 7.9.

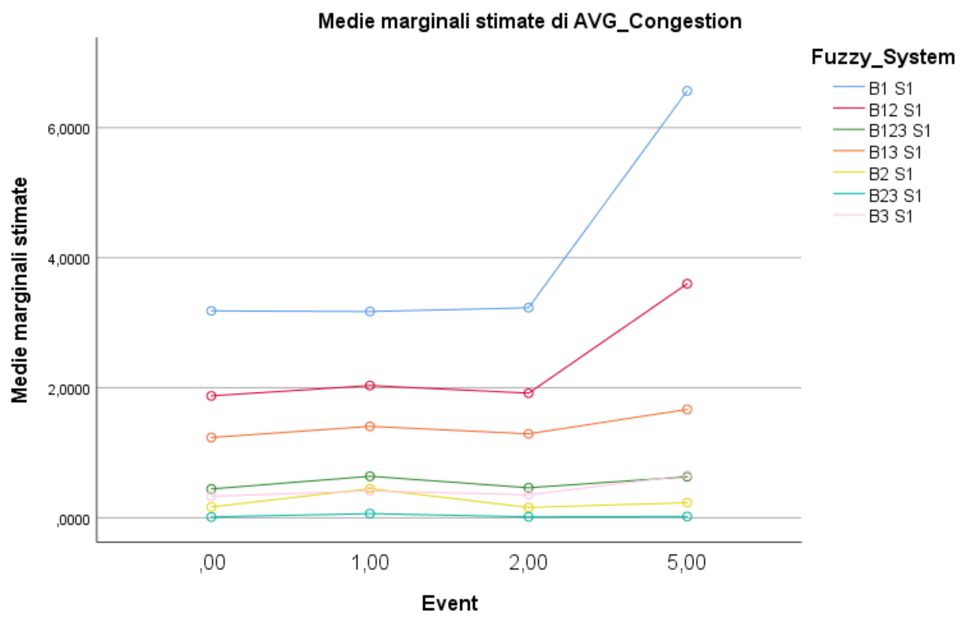


Figure 7.9 Estimated Marginal Means of AVG Congestion calculated for each Event Type and differentiated for the 7 Fuzzy Systems. 0 means No Event

Figure 7.9 shows that the best Fuzzy System in terms of AVG Congestion during an Event of Type 5 is B23 S1, represented by the light sea green line.

When no events happen in the yard, it is not possible to apply the matching principle that has driven the current approach. Therefore, in order to indicate a best Fuzzy System for a *No Event* scenario, Table 7.16 is used: it indicates the best Fuzzy System in terms of AVG Rank as B2 S1. The theoretical argument for this choice is that when no events happen, there are no KPIs that need to be rescued. Hence, all the KPIs assume the same importance and they can be evaluated together with a ranking approach.

7.3.2 Training Results for Rehandles

7.3.2.1 Statistical Approach

For *Event 1*, the descriptive statistics of the Fuzzy Systems are shown in Table 7.20. The Fuzzy System with the lowest mean of AVG Rehandles is B2 S1, which is the candidate to be the best Fuzzy System for Event 1. However, the post hoc test shows that there is no statistically significant difference between the means of B2 S1 and B3 S1.

Therefore, the two best Fuzzy Systems for Event 1 under a Statistical approach for Rehandles are B2 S1 and B3 S1.

Descriptive Statistics

Dependent variable: AVG Rehandles

Fuzzy System	Mean	Std. Deviation	Runs
B1 S1	1,12006018	,025986486	30
B12 S1	1,09612170	,022166716	30
B123 S1	1,05626881	,027484126	30
B13 S1	1,10090674	,028322806	30
B2 S1	1,01120027	,027222531	30
B23 S1	1,04623872	,025355676	30
B3 S1	1,02768305	,031062136	30
Total	1,06549706	,046257368	210

Table 7.20 Descriptive Statistics of the 7 Fuzzy Systems for Event 1

For *Event 2*, the descriptive statistics of the Fuzzy Systems are shown in Table 7.21.

Descriptive Statistics

Dependent variable: AVG Rehandles

Fuzzy System	Mean	Std. Deviation	Runs
B1 S1	1,14496824	,018940739	30
B12 S1	1,11059846	,017501808	30
B123 S1	1,05596790	,021638206	30
B13 S1	1,13507188	,026683568	30
B2 S1	1,03299900	,018160205	30
B23 S1	1,06201939	,021658530	30
B3 S1	1,07539285	,015216135	30
Total	1,08814539	,044242640	210

Table 7.21 Descriptive Statistics of the 7 Fuzzy Systems for Event 2

The Fuzzy System with the lowest mean of AVG Rehandles is B2 S1, which is the candidate to be the best Fuzzy System for Event 2. From the post hoc test, the mean of B2 S1 shows a statistically significant difference with the means of all the other Fuzzy Systems.

Therefore, the best Fuzzy System for Event 2 under a Statistical approach for Rehandles is B2 S1.

For *Event 5*, the descriptive statistics of the Fuzzy Systems are shown in Table 7.22. The Fuzzy System with the lowest mean of AVG Rehandles is B2 S1, which is the candidate to be the best Fuzzy System for Event 5. From the post hoc test, the mean of B2 S1 shows a statistically significant difference with the means of all the other Fuzzy Systems.

Therefore, the best Fuzzy System for Event 5 under a Statistical approach for Rehandles is B2 S1.

Descriptive Statistics

Dependent variable: AVG Rehandles

Fuzzy System	Mean	Std. Deviation	Runs
B1 S1	1,15543313	,007994287	30
B12 S1	1,09351387	,004260974	30
B123 S1	1,04831160	,024857887	30
B13 S1	1,10822467	,021676569	30
B2 S1	1,02547643	,005421242	30
B23 S1	1,04129054	,018967003	30
B3 S1	1,06944166	,026914437	30
Total	1,07738456	,045488845	210

Table 7.22 Descriptive Statistics of the 7 Fuzzy Systems for Event 5

Finally, for the <u>*No Event*</u> scenario, Table 7.16 shows that the best Fuzzy System with regards to AVG Rehandles is B23 S1. As a matter of fact, the rank of its AVG Rehandles is 1. For this reason, B23 S1 is considered the best Fuzzy System for <u>*No Event*</u>.

7.3.2.2 Descriptive Approach

For <u>*Event 1*</u>, the Frequency of Victories in terms of AVG Rehandles for each Fuzzy System out of the 30 Runs are listed in Table 7.23.

Fuzzy System	**Frequency of Victories**
B1 S1	0
B2 S1	20
B3 S1	5
B12 S1	0
B13 S1	0
B23 S1	4
B123 S1	1
Total Runs	**30**

Table 7.23 Frequency of Victories for Event 1

The Fuzzy System that has the highest Frequency of Victories is B2 S1 with 20. At the same time, there are other 3 Fuzzy Systems that have scored more than a victory (B3 S1, B23 S1 and B123 S1). However, since B2 S1 has proven to have the lowest AVG Rehandles value for 2/3 of the Runs, it is arbitrarily considered a clear winner. For this reason, B2 S1 is considered the best Fuzzy System for Event 1 under a Descriptive Approach.

For *Event 2*, the Frequency of Victories in terms of AVG Rehandles for each Fuzzy System out of the 30 Runs are listed in Table 7.24.

Fuzzy System	**Frequency of Victories**
B1 S1	0
B2 S1	22
B3 S1	1
B12 S1	0
B13 S1	0
B23 S1	4
B123 S1	3
Total Runs	**30**

Table 7.24 Frequency of Victories for Event 2

The Fuzzy System that has the highest Frequency of Victories is B2 S1 with 22. It is therefore considered a clear winner. For this reason, B2 S1 is considered the best Fuzzy System for Event 2 under a Descriptive Approach.

For *Event 5*, the Frequency of Victories in terms of Average Rehandles for each Fuzzy System out of the 30 Runs are listed in Table 7.25.

Fuzzy System	**Frequency of Victories**
B1 S1	0
B2 S1	19
B3 S1	2
B12 S1	0
B13 S1	0
B23 S1	5
B123 S1	4
Total Runs	**30**

Table 7.25 Frequency of Victories for Event 5

The Fuzzy System that has the highest Frequency of Victories is B2 S1 with 19. At the same time, there are other 3 Fuzzy Systems that have scored more than a victory, with two of them (B23 S1 and B123 S1) having a considerable frequency (5 and 4 respectively). From a purely descriptive point of view and with an arbitrary choice, B2 S1 is considered the best Fuzzy System for Event 5. B23 S1 and B123 S1 are both discarded since their Frequency of

Victories are considered not sufficient to allow them to rise to the status of best Fuzzy System, given that the Frequency of Victories of B2 S1 is almost four and five times bigger.

With regards to the *No Event* scenario, given the fact that there is only one Run available, the best Fuzzy System under a Descriptive Approach for Rehandles is defined as the strategy that has the lowest rank for AVG Rehandles. Table 7.16 shows that that system is B23 S1.

7.3.2.3 Descriptive Approach from SPSS

As mentioned above, considering AVG Rehandles as the sole KPI to determine the best Fuzzy System for each Event makes the matching principle invalid: there is no more link between the class of the Event that is happening and the KPI that is altered and needs to be rescued. However, the use of estimated marginal means is still considered a valid option to find the best Fuzzy System for every Event Type. Hence, Figure 7.8, which reports the estimated marginal means of AVG Rehandles, calculated across the 30 Runs for each Event Type and differentiated for each Fuzzy System, is still useful.

The graph shows that:

- The Best Fuzzy System for a *No Event* scenario is B23 S1, represented by the light green sea line in correspondence with the value 0 on the horizontal axis.
- The Best Fuzzy System for *Event 1* is B2 S1, represented by yellow line in correspondence with the value 1 on the horizontal axis
- The Best Fuzzy System for *Event 2* is B2 S1, represented by yellow line in correspondence with the value 2 on the horizontal axis
- The Best Fuzzy System for *Event 5* is B2 S1, represented by yellow line in correspondence with the value 5 on the horizontal axis

7.4 The Construction of the Fuzzy Selector and the Dynamic Systems

The output of the training phase is a list of best Fuzzy Systems for each Event Type, according to the three proposed approaches. These Fuzzy Systems are then combined together to construct multiple Dynamic Fuzzy Allocation Systems (in short Dynamic System). These Dynamic Systems are supported by a so-called Fuzzy Selector, which constitutes the backbone of the proposed dynamic Decision Support System (DSS) for the allocation of incoming containers. The idea that lies behind the creation of the Fuzzy Selector is the following: if there are different Fuzzy Systems that perform best under different disturbances, the Fuzzy Selector should be able to recognize the event that is happening at any given time and react immediately by choosing the Fuzzy System that gives the best results in those conditions. The underlying assumption is that a Fuzzy Selector that adopts the best possible stacking strategy under each condition should perform better than the traditional fixed allocation policies. It is worth noting that the Selector is given the adjective Fuzzy because it selects between different, crisp, Fuzzy Systems and not because there is a form of fuzziness in the choice.

To implement the Fuzzy Selector, the Block and Stack Assignment functions in the model have been modified: an input variable which indicates the Event Type that is occurring at any given minute has been added. Each function than selects the fuzzy Block and Stack Assignment policies according to the value of said input variable.

For the Combination of KPIs, there are three different lists of best Fuzzy Systems for each Event, one for each of the proposed approaches. This generates three different Dynamic Systems: for each one of the three, when a certain event is happening, the best Fuzzy System according to the specific approach is selected and applied. The name of the Dynamic System originates from the approach applied to find the best Fuzzy Systems for each Event. In a few cases there are two best Fuzzy Systems for the same Event Type. What could possible be done? The proposed solution is to choose randomly between the two: when the relative disturbance is occurring, one of the two Fuzzy Systems is picked randomly; during a successive occurrence of the same event, the random selection might choose the other Fuzzy System. The outline of the three Dynamic Systems for the Combination of KPIs is shown in Table 7.26: each column indicates which Fuzzy Systems are adopted for each Event Type.

DYNAMIC SYSTEMS FOR COMBINATION OF KPIs			
Event Type	**Dynamic System from Statistical Approach**	**Dynamic System from Descriptive Approach**	**Dynamic System from Descriptive Approach from SPSS**
No Event	B2 S1 or B23 S1	B2 S1	B2 S1
Event 1	B2 S1 or B23 S1	B2 S1 or B23 S1	B2 S1 or B123 S1
Event 2	B2 S1	B2 S1 or B123 S1	B2 S1
Event 5	B2 S1	B2 S1 or B123 S1	B23 S1

Table 7.26 The composition of the three Dynamic Systems for Combination of KPIs

The same consideration expressed for Combination of KPIs are valid for Rehandles. Three Fuzzy Systems for Rehandles have been developed and are presented in Table 7.27.

DYNAMIC SYSTEMS FOR REHANDLES			
Event Type	**Dynamic System from Statistical Approach**	**Dynamic System from Descriptive Approach**	**Dynamic System from Descriptive Approach from SPSS**
No Event	B23 S1	B23 S1	B23 S1
Event 1	B2 S1 or B3 S1	B2 S1	B2 S1
Event 2	B2 S1	B2 S1	B2 S1
Event 5	B2 S1	B2 S1	B2 S1

Table 7.27 The composition of the three Dynamic Systems for Rehandles

It is interesting to note that the structures of the Dynamic Systems for Rehandles from the Descriptive Approach and from the Descriptive Approach from SPSS are exactly the same: the same Fuzzy Systems are applied during the occurrence of the same Events.

7.5 Testing Phase

To evaluate the validity of the proposed dynamic Decision Support System (DSS) for the allocation of incoming containers under the effect of disruptive events, based on a Fuzzy Selector able to implement different Dynamic Systems, a specific campaign of testing is put in place. The aim of this campaign is to verify whether a Dynamic stacking System, which switches in real time between stacking policies depending on what is happening in its surroundings, yields better results than a traditional, Static Allocation System that keeps the same policy notwithstanding eventual disturbances.

7.5.1 Selection of the Allocation Systems to be Tested

For both Combination of KPIs and Rehandles, the same rationale is applied to the selection of the allocation systems for the testing round: all the three Dynamic Systems and the single, static, Fuzzy Systems that compose them, are put to the test. Moreover, two real-life practices, based on randomness, have been added:

- Random Stacking: it is a quite common procedure in small-to-medium, non-automated container terminals. During the Block Assignment phase, a block is chosen randomly to allocate an incoming container: if the block is not full, the container is directed towards it; it the block is not available, another block is selected randomly. Once the container arrives in front of the block, it is time for Stack Assignment: within the chosen block, a stack is selected entirely randomly among the available ones.

- Semi-Random Stacking: it is the stacking strategy currently in use at the port of Arica. The Block Assignment phase is performed exactly in the same way as for the Random strategy: a block is chosen at random among the available ones. The Stack Assignment phase, however, is performed in a sequential manner: the first stack is filled up to the maximum height; after that, the same procedure is repeated again, starting from the first available slot, in the stack next to the previous one. The strategy is clarified in Figure 7.10.

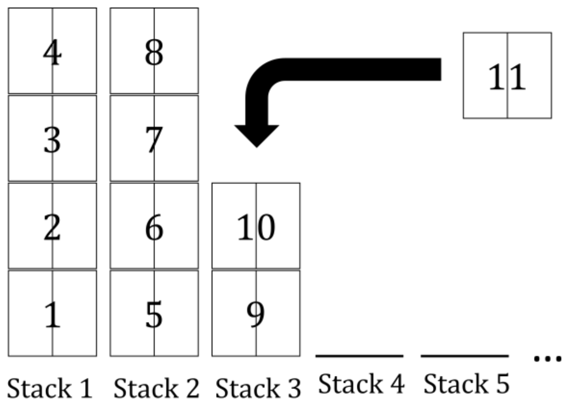


Figure 7.10 A representation of the Semi-Random Stacking strategy. The numbers on the containers indicate the order of arrival. The represented stacks have a maximum height of four tiers

The selected allocation systems for the testing phase for either Combination of KPIs and Rehandles are collected in Table 7.28 and 7.29 respectively.

Category	Allocation System
Dynamic	Dynamic System from Statistical Approach
Dynamic	Dynamic System from Descriptive Approach
Dynamic	Dynamic System from Descriptive Approach from SPSS
Static	B2 S1
Static	B23 S1
Static	B123 S1
Random	Random Stacking
Random	Semi Random Stacking

Table 7.28 Tested Allocation systems for Combination of KPIs

Category	Allocation System
Dynamic	Dynamic System from Statistical Approach
Dynamic	Dynamic System from Descriptive Approach or Dynamic System from Descriptive Approach from SPSS
Static	B2 S1
Static	B3 S1
Static	B23 S1
Random	Random Stacking
Random	Semi Random Stacking

Table 7.29 Tested Allocation systems for Rehandles

8 Allocations Systems are tested for Combination of KPIs and 7 for Rehandles: in fact, the two Dynamic Systems for Rehandles, from Descriptive Approach and from Descriptive Approach from SPSS, have exactly the same composition in terms of single Fuzzy Systems so there is no point in testing them both.

7.5.2 Characteristics of the Events

Dynamic and Static Systems have to be compared under the effect of disturbances, so a series of events has to be simulated. In order to do this, a set of traditional *Event Matrices* is generated through the respective Matlab script. Only Events 1, 2 and 5 are included in the matrices because they are the events that have been included in the training phase and for which the Static Fuzzy Systems have been tested. The characteristics of the events that have been included in the *Event Matrices* are:

- Event 1 (Blocking a block). *Event Duration*: 235 minutes. Every time the event happens, 1 out of the 9 blocks is randomly selected to be shut down. Not more than one block is allowed to be closed at a time in order to avoid reaching the full capacity of the yard. There are no restrictions to the moments in which the Event can occur: it may or may not happen during the unloading time of a vessel; in the former case, the Event has an impact on the operations in the yard but in the latter the disturbance has no effect.
- Event 2 (Traffic Jam). *Event Duration* = 240 minutes. *Jam Delay* = 240 minutes. There are no restrictions to the moments in which the Event can occur, not even the time windows introduced during the training phase. For the sake of simplicity, a traffic jam can happen at any time.
- Event 5 (Crane Breakdown). *Event Duration* = 235 minutes. Every time the event happens, 1 out of the 3 reach-stackers that serve the allocation phase at the blocks is

selected to be considered out of work. Not more than one reach-stacker is assumed to break down at any given time. There are no restrictions to the moments in which the Event can occur: it may or may not happen during the unloading time of a vessel.

As for the way in which the *Event Matrix* is generated, stated in Chapter 6, only one event can happen at a single time (minute) at the yard. In order to have periods of time in which no events happen in the yard, the values of *Threshold Vector* for Event 1, 2 and 5 are set at 0.7.

7.5.3 Final Testing Outline

Each one of the 8 Allocations Systems for Combination of KPIs is tested:

- 30 times. In each one of these times, also called Runs in the testing phase, all the 8 Allocation Systems are tested using the same *Event Matrix*, which is indicated by one or more capital letters (Run A, Run B, Run C etc.) and is a combination of Event 1, 2, 5 and No Event. Each *Event Matrix* is different from the other thanks to the randomness in the generation of the events.

In its their own right, each one of the 7 Allocation Systems for Rehandles is tested:

- 30 times. In each one of these times, also called Runs in the testing phase, all the 7 Allocation Systems are tested using the same *Event Matrix*, which is indicated by one or more capital letters (Run A, Run B, Run C etc.) and is a combination of Event 1, 2, 5 and No Event. Each *Event Matrix* is different from the other thanks to the randomness in the generation of the events. The set of 30 *Event Matrices* is different from the set of matrices for Combination of KPIs.

The number of Runs of both cases, 30, has been designed in order to have statistically significant results.

7.5.4 Performance Evaluation

For both Combination of KPIs and Rehandles, the performances are evaluated only for the containers that have their *Arrival Times* and *Real Times of Departure* comprised between minute 35200 (the arrival time of vessel number 10) and minute 72000 (the end of day 50): in this way, performances are assessed with an overall yard utilisation above 70%.

For the Combination of KPIs, each Allocation System in each Run is evaluated through all the four main KPIs introduced in the training phase: Total Congestion, Total Rehandles, Total Distance and AVG STD Per Day. They are also expressed in their average form: AVG Congestion, AVG Rehandles, AVG Distance and AVG STD Per Day (which is already an average). To determine the best Allocation System for each single Run, the Ranking Approach is applied to AVG Congestion, AVG Rehandles, AVG Distance and AVG STD Per Day, in the same way as described in section 7.2.4.2. It may be evident but it is very important to note that the Dynamic Systems are evaluated on the grounds of the same method (Ranking Approach) that has been used to define the best performing Fuzzy Systems that compose them.

For Rehandles, the evaluation is way simpler: the Allocations Systems are evaluated through AVG Rehandles.

7.6 Testing Results

The results of the testing phase are presented for Combination of KPIs first and for Rehandles later. A brief commentary is provided for both. The complete lists of results files can be provided upon request.

7.6.1 Testing Results for Combination of KPIs

The simplified appearance of one instance of the results file is shown in Table 7.30.

Category	Allocation System	Run	Rank Congestion	Rank Rehandles	Rank Distance	Rank AVG STD Per Day	AVG Rank	STD Rank
Dynamic	Dynamic from Stat.	C	2	2	3	2	2,25	0,5
Dynamic	Dynamic from Desc.	C	3	3	6	4	4	1,414214
Dynamic	Dynamic from SPSS	C	4	4	5	3	4	0,816497
Static	B2 S1	C	5	5	8	1	4,75	2,872281
Static	B23 S1	C	1	1	2	5	2,25	1,892969
Static	B123 S1	C	6	6	1	8	5,25	2,986079
Random	Random	C	8	7	4	6	6,25	1,707825
Random	Semi-Random	C	7	8	7	7	7,25	0,5

Table 7.30. Example of simplified version of the results file for Combination of KPIs

With this outline, for Run C all the three Dynamic Systems, the three Static Systems and the two Random Systems are compared at the same time. In this case, the best performing allocation system is the Dynamic System from Statistical Approach, since it has both the lowest AVG Rank and STD Rank (the Static System B23 S1 has the same value of AVG Rank but a higher STD Rank).

A first way to address the results of the training phase is to count the number of times (also called Victories) in which every single Allocation System has proven to be the best performing one in terms of AVG Rank for a single Run. It is a very similar procedure of the one proposed for the Descriptive Approach for the training phase. The results are collected in Table 7.31.

Allocation System	**Victories**	**Total Victories for Category**
Dynamic System from Stat. Approach	11	22
Dynamic System from Descr. Approach	2	
Dynamic System from Descr. Approach from SPSS	9	
B2 S1	3	8
B23 S1	5	
B123 S1	0	
Random	0	0
Semi Random	0	
Draws	0	0
Total Runs	30	

Table 7.31 Results collection for Combination of KPIs: the frequency of victories on the grounds of AVG Rank

The table shows quite clearly how the Dynamic Systems clearly outperform the Static ones in terms of frequency of victories: 22 to 8. The Random Systems appear not competitive at all, with a null frequency. This means that a Dynamic System is the best of a single Run, in terms of AVG Rank, more than 2 out of 3 times. Moreover, the two single Allocation Systems with the highest number of victories are the Dynamic System from Statistical Approach and the Dynamic system from Descriptive Approach from SPSS, with 11 and 9 respectively.

These results, however, are deceptive: comparing all the three Dynamic Systems at the same time alters the results. The AVG Rank and the ranking approach that generates it, in fact, are relative performance indicators: they show how good a certain Allocation System is compared to others. If the Dynamic Systems perform similarly, the close values of their KPIs might increase the difference with a Static System: let’s consider a Dynamic System that, for a certain KPI, as a Rank of 1; the other two Dynamic Systems perform rather well, with a Rank of 2 and 3; the closest Static System has a rank of 4; however, if the first Dynamic System was to be compared directly with the Static Systems, discarding the other Dynamic

Systems, the rank of the latter would immediately drop to 2, with a significant impact on the AVG Rank.

This is exactly what happens if the Dynamic System from Statistical Approach, which has the lowest rank for Run C in the outline shown in Table 7.30, is compared directly to the Static and Random Systems. The direct comparison is represented in Table 7.32.

Category	**Allocation System**	**Run**	**Rank Congestion**	**Rank Rehandles**	**Rank Distance**	**Rank AVG STD Per Day**	**AVG Rank**	**STD Rank**
Dynamic	Dynamic from Stat.	C	2	2	3	2	2,25	0,5
Static	B2 S1	C	3	3	6	1	3,25	2,061553
Static	B23 S1	C	1	1	2	3	1,75	0,957427
Static	B123 S1	C	4	4	1	6	3,75	2,061553
Random	Random	C	6	5	4	4	4,75	0,957427
Random	Semi-Random	C	5	6	5	5	5,25	0,5

Table 7.32. Example of the direct comparison of one Dynamic System with the Static and Random systems

It is interesting to see how the relative ranks of the Dynamic System remain the same. What changes is the Rank of AVG STD Per Day of the Static System B23 S1, dropping down from 5 to 3: as a matter of fact, the allocation systems that occupied rank 2 and 3 were the other two Dynamic Systems. As a result, the AVG Rank of B23 S1 is reduced to 1.75, the best of the tested systems for Run C.

Proceeding in the way described in Figure 7.30 is wrong: each Dynamic System is compared not only with Static and Random Systems but also with other Dynamic Systems, which is not wrong *per se*, but it defeats the clear purpose of the testing phase. The solution, then, is to evaluate each single Dynamic System on the grounds of AVG Rank with Static and Random Systems exclusively.

For the *Dynamic System from Statistical Approach*, the frequency of victories over the 30 runs is shown in Table 7.33: the Dynamic System is the best for 16 times. On the other hand, a Static System is the best for 14 times, 8 times with B2 S1 and 6 times with B23 S1. A more significant way of assessing performances is calculating mean and standard deviation of AVG Rank across the 30 Runs for each Allocation System, without considering the other Dynamic Systems. The relative results are listed in Table 7.34 and appear to be rather promising: the Dynamic System has the lowest mean and by a certain margin. Moreover, it has the second lowest STD, which indicates a good consistency. It is also interesting to note the poor performances of the Random System, whose mean values are almost two times the mean of the Dynamic System.

Allocation System	**Victories**	**Total Victories for Category**
Dynamic System from Stat. Approach	16	16
B2 S1	8	
B23 S1	6	14
B123 S1	0	
Random	0	0
Semi Random	0	
Draws	0	0
Total Runs	30	

Table 7.33 Frequency of victories for Dynamic System from Stat. Approach

Allocation System	**Mean AVG Rank**	**Standard Deviation**
Dynamic System from Stat. Approach	2,344828	0,4299802
B2 S1	2,672414	0,4868973
B23 S1	2,62931	0,4846791
B123 S1	3,931034	0,3126539
Random	4,672414	0,4914603
Semi Random	4,741379	0,4794098

Table 7.34 Mean and Standard Deviation of AVG Rank for the Dynamic System from Stat. Approach evaluation

For the *Dynamic System from Descriptive Approach*, the frequency of victories over the 30 run is shown in Table 7.35: the Dynamic System is the best performing one only 8 times. The two Static Systems outperform the Dynamic one both as a category and as single Allocation Systems: B2 S1 is the winner for 11 times while B23 S1 for 10 times. The results coming from the descriptive statistics, presented in Table 7.36, are not encouraging: the Dynamic System shows both a larger mean and standard deviation than the two best performing Static Systems, B2 S1 (the lowest mean) and B23 S1. However, there is still no comparison between the current practice at the port of Arica and the Dynamic System.

Allocation System	**Victories**	**Total Victories for Category**
Dynamic System from Descr. Approach	8	8
B2 S1	11	
B23 S1	10	21
B123 S1	0	
Random	0	0
Semi Random	0	
Draws	1	1
Total Runs	30	

Table 7.35 Frequency of victories for Dynamic System from Descr. Approach

Allocation System	**Mean AVG Rank**	**Standard Deviation**
Dynamic System from Descr. Approach	2,633333	0,511646
B2 S1	2,391667	0,4238744
B23 S1	2,516667	0,495381
B123 S1	3,975	0,2734675
Random	4,7	0,4275028
Semi Random	4,758333	0,4756272

Table 7.36 Mean and Standard Deviation of AVG Rank for the Dynamic System from Descr. Approach evaluation

Finally, for the *Dynamic System from Descriptive Approach from SPSS*, the frequency of victories over the 30 runs is shown in Table 7.37: the Dynamic System is the best for 9 times, around one third of the total Runs. It is almost a tie with two of the three Static Systems: B2 S1 is the winner for 10 times, while B23 S1 for 9 times. The same pattern is followed with the means of the AVG Rank, listed in Table 7.38: Dynamic System, B23 S1 and B2 S1 have very similar means, with the latter being the best of the three; on a brighter note, the Dynamic System has the lowest STD of the three. The Dynamic Systems proves again to outperform considerably the two Allocation Systems based on a certain degree of randomness.

Allocation System	**Victories**	**Total Victories for Category**
Dynamic System from Descr. Approach from SPSS	9	9
B2 S1	10	
B23 S1	9	19
B123 S1	0	
Random	0	0
Semi Random	0	
Draws	2	2
Total Runs	30	

Table 7.37 Frequency of victories for Dynamic System from Descr. Approach from SPSS

Allocation System	**Mean AVG Rank**	**Standard Deviation**
Dynamic System from Descr. Approach from SPSS	2,55	0,4224314
B2 S1	2,533333	0,4583203
B23 S1	2,541667	0,4309339
B123 S1	3,933333	0,3211867
Random	4,691667	0,4719883
Semi Random	4,741667	0,4571432

Table 7.38 Mean and Standard Deviation of AVG Rank for the Dynamic System from Descr. Approach from SPSS evaluation

7.6.2 Testing Results for Rehandles

The interpretation of the testing results for Rehandles is much more direct since it involves only one KPI, AVG Rehandles. The mean and standard deviation of AVG Rehandles have been calculated for every Allocation System across the 30 runs. The results are shown in Table 7.40.

Allocation System	**Mean AVG Rehandles**	**Standard Deviation**
Dynamic System from Stat. Approach	1,056791314	0,032305481
Dynamic System from Descr. Approach / Dynamic System from Descr. Approach from SPSS	1,044543798	0,032181214
B2 S1	1,060493325	0,038363476
B3 S1	1,075937009	0,033040465
B23 S1	1,06928266	0,025990631
Random	1,557131987	0,028693182
Semi Random	1,535310523	0,033113874

Table 7.40 Descriptive Statistics (mean and STD) of AVG Rehandles

The table shows that the two Dynamic Systems have the two lowest means. The difference with the means of B2 S1, B3 S1 and B23 S1 are rather reduced, in the order of 1/100. This means, anyway, that the two Dynamic Systems appear to behave better under the effect of disturbances: they generate less Rehandles. They also have lower STDs than their Static counterparts. A difference with the case of Combination of KPIs, here all the three Static Systems look almost equally competitive. Finally, the Dynamic Systems are still clearly outperforming the random-based strategies.

Just from a descriptive point of view it could be interesting to see the frequencies of victory of the single Dynamic Systems compared to the Static and Random one. In this case, an Allocation System is a winner when it has the lowest AVG Rehandles value of the Run.

The data for the Dynamic System from Statistical Approach are listed in Table 7.41.

Allocation System	**Victories**	**Total Victories for Category**
Dynamic System from Stat. Approach	10	10
B2 S1	10	20
B3 S1	6	
B23 S1	4	
Random	0	0
Semi Random	0	
Draws	0	0
Total Runs	30	

Table 7.41 Frequency of victories for Dynamic System from Stat. Approach for Rehandles

The table shows that the highest frequencies are split between a Dynamic System and a Static one (B2 S1). The Dynamic System yields a lower value of AVG Rehandles 10 times out of 30.

The data for the Dynamic System from Descriptive Approach / Descriptive Approach from SPSS are listed in Table 7.42. In this case, the Dynamic System is a clear winner in more than half of the Runs, which is a considerable result: its frequency of victory (16) is the double of the second most frequent winner (B2 S1). More than in half of the Runs the Dynamic System has proven to generate the lowest level of AVG Rehandles.

Allocation System	Victories	Total Victories for Category
Dynamic System from Descr. Approach / Dynamic System from Descr. Approach from SPSS	16	16
B2 S1	8	14
B3 S1	4	
B23 S1	1	
Random	0	0
Semi Random	0	
Draws	0	0
Total Runs	30	

Table 7.41 Frequency of victories for Dynamic System from Descriptive Approach / Descriptive Approach from SPSS for Rehandles

7.6.3 Final Considerations on the Results of the Testing Phase

For the Combination of KPIs, only the *Dynamic System from Statistical Approach* proved to behave better under the effect of disturbances than the fixed Fuzzy Systems that constitute it. It did not only have the lowest mean of AVG Rank but it also was the best performing system in around half of the testing Runs. This is not totally unexpected since this Dynamic System has been constructed with a careful statistical procedure. The *Dynamic System from Descriptive Approach* behaved rather poorly, both in comparison with the Fuzzy Systems that are employed to build it and with the other Dynamic Systems. It was not competitive in either the statistics or the frequency of victories. One reason for this could be that adopting a simple descriptive approach to construct a Dynamic System that focuses on a complicated Performance Indicator such as the AVG Rank (which combines four different KPIs) might not be a good strategy. Another possible explanation could be found in the rationale adopted for the selection of the best Fuzzy Systems at the end of the training phase: some of the choices were made arbitrarily. The *Dynamic System from Descriptive Approach from SPSS* performed almost on par with its relative Fuzzy Systems. This could be a hint that the matching principle and the “KPI rescue” concept could be valid.

With regards to the Rehandles side of testing, the two proposed Dynamic Systems outperformed their respective fixed Fuzzy Systems. In particular, the *Dynamic System from Descriptive Approach / from Descriptive Approach from SPSS* proved to be the best allocation strategy in a little more than half of the Runs. It is worth noting that this Dynamic System has a structure that resembles a fixed Fuzzy System, with a change of policy only when no events are happening. This might suggest that even the smallest changes of stacking strategy might prove beneficial in a container terminal.

Finally, the fact that the best performing Dynamic System for the complicated Combination of KPIs came from the Statistical Approach while the best performing Dynamic System for the single KPI AVG Rehandles came from a descriptive approach, could suggest the existence of a link between the complexity of the addressed Performance Indicator and the approach to the selection of the best Fuzzy Systems: the more complex the metric is, the more carefully planned the construction of the Dynamic Systems should be.

CHAPTER 8

Conclusions and Recommendations for Further Research

This is the closing chapter of this work. It includes the conclusions and a list of ideas that could lead to further research on the topic.

8.1 Conclusions

This work has addressed the real-time container allocation problem by constructing a dynamic Decision Support System (DSS) that, through a specific Fuzzy Selector, is able to choose between different stacking strategies according to the events and disturbances that might be occurring at a container terminal. A thorough research of the existing literature on the topic of real-time stacking strategies has generated a novel classification of allocation Criteria, Performance Indicators and Events. A "matching principle" has also been introduced: the three elements described above can be grouped into the same three main classes (Container-related, Yard-related and Resource-related) according to which part of the yard they are referring to. This classification has been adapted to the real case study of the Port of Arica, Chile and a series of specific Criteria have been developed with the aid of Fuzzy Logic. Fuzzy Logic, which works with sets with unclear boundaries, is considered particularly suitable for an environment subjected to high uncertainty such as a container terminal under the effect of disruptions. The proposed Criteria have been combined to form Decision Rules, called Fuzzy Systems, which are the bricks that are used to build the DSS. To understand how those systems react to disturbances, a Matlab model that simulates the operations in the container terminal has been implemented. Moreover, a set of events have been generated through a dedicated Matlab script. These tools have been used to define a training program: its aim is to define the best performing Fuzzy System for each Event. The definition of "best" Fuzzy System has been demanded to two different performance metrics: the average number of rehandles (AVG Rehandles) and a combination of four main KPIs (AVG Congestion, AVG Rehandles, AVG Distance and AVG STD Per Day) performed through a novel Ranking approach. Those best Fuzzy Systems have been combined adopting three different approaches to create a series of Dynamic Systems. The engine of those Dynamic Systems is represented by the Fuzzy Selector: it is able to recognize in real time the event or the disturbance that is impacting the yard and it changes the stacking strategy accordingly. Finally, the Dynamic Systems have been tested under the assumption that they should perform better than the single Fuzzy Systems that compose them. The results of the testing show that, effectively, there are Dynamic Systems that perform better than their respective Fuzzy Systems. For a DSS focused on a combination of the four main KPIs, the best performing Dynamic System has been constructed with a carefully planned statistical approach; on the other hand, when the DSS aims at reducing one single KPI such as AVG Rehandles, the best Dynamic System can be built with a more direct descriptive approach. Finally, all the proposed Dynamic systems outperformed the semi-random stacking strategy that is currently being adopted by the Port of Arica.

8.2 Recommendations for Further Research

This work relates to many aspects of the operations at a container terminal, providing significant scope for further research. Some of the main ideas are listed below:

- Extending the training phase by introducing an extension of simulated events
- Considering further relevant criteria that may be of relevance in context of the container space allocation problem
- Exploring multi-objective techniques to consider multiple KPIs into one single Performance Index
- Reviewing the design of the Fuzzy Selector when systems perform with similar quality and providing a more refined selection process, possibly by means of weights
- Providing an in-depth analysis of the relationship between event characteristics, Fuzzy System design and performance, taking into considerations learning-based approaches
- Adapting a Fuzzy Logic selector approach to other environments where uncertainty is high and disruptions are frequent, e.g. commodity stacking in industrial, or where the goods are stored in stacks, e.g. certain car storages work as horizontal stacks

Bibliography

Borgman B., van Asperen E., Dekker R. (2010), Online rules for container stacking, *OR Spectrum*, **32**, 687-716

Chen L., Lu Z., (2012), The storage location assignment problem for outbound containers in a maritime terminal, *Int. J. Production Economics*, **135**, 73-80

De Castilho B., Daganzo, C.F., (1993), Handling strategies for import containers at marine terminal, *Transportation Research Part B,* **27(2)**, 151–166

Dekker R., Voogd P., van Asperen E., (2006), Advanced methods for container stacking, *OR Spectrum,* **28(4)**, 563–586

Duinkerken M.B., Evers J.J., Ottjes J.A., (2001), A simulation model for integrating quay transport and stacking policies in automated terminals, *Proceedings of the 15th European simulation multiconference (ESM2001)*, Prague, Czech Republic

Gharehgozli A., Zaerpour N., (2018), Stacking outbound barge containers in an automated deep-sea terminal, *European Journal of Operational Research*, **267**, 977-995

Grötschel M., Krumke S.O., Rambau J., (2001), *Online Optimization of Large Scale Systems*, Springer

Guerra-Olivares R., Smith N.R., Gonzàlez-Ramìrez R.G., Garcìa-Mendoza E., Càrdenas-Barron L.E., (2018), A heuristic procedure for the outbound container space assignment problem for small and midsize maritime terminals, *Int. J. Mach. Learn. & Cyber*., **9,** 1719-1732

Güven C., Eliiyi D.T., (2018), Modelling and optimisation of online container stacking with operational constraints, *Maritime Policy & Management*, **46:2**, 201-216

Jang J.-S. R., Gulley N., (1995), *MATLAB Fuzzy Logic Toolbox*, The Mathworks Inc.

Kim K.H, (1997), Evaluation of the number of re-handles in container yards, *Computers & Industrial Engineering*, **32(4)**, 701–711

Kim K.H., Kim H.B., (1998), The optimal determination of the space requirement and the number of transfer cranes for import containers, *Computers Ind. Eng.*, **35**, 427-430

Kim K.H., Kim H.B, (1999), Segregating space allocation models for container inventories in port container terminals, *International Journal of Production Economics*, **59(1)**, 415–423

Kim K.H., Park Y.M., Ryu K.R, (2000), Deriving decision rules to locate export containers in container yard, *European Journal of Operational Research*, **124(1)**, 89–101

Kozan E., Preston P., (1999), Genetic algorithms to schedule container transfers at multimodal terminals, *Intl. Trans. in Op. Res.*, **6**, 311-329

Maldonado S., Gonzàlez-Ramìrez R.G., Quijada F., Ramirez-Nafarrate A., (2019), Analytics meets port logistics: A decision support system for container stacking operations, *Decision Support Systems*, **121**, 84-93

Meydanoglu E.S.B., (2009), The role of Supply Chain event management systems for supply chain risk management, *European and Mediterranean Conference on Information Systems 2009*, Izmir, Turkey

Park R.C.T., Kim Y.H., Ryu K.R., (2011), Dynamic adjustment of container stacking policy in an automated container terminal, *International Journal of Production Economics*, **133(1)**, 385–392

Petering M.E.H., (2015), Real-time container storage location assignment at an RTG-based seaport container transshipment terminal: problem description, control system, simulation model, and penalty scheme experimentation, *Flex Serv Manuf J,* **27**, 351–381

Petering M.E.H., Wu Y., Li W., Goh M., de Souza R., Murty K.G., (2017), Real-time container storage location assignment at a seaport container transshipment terminal: dispersion levels, yard templates, and sensitivity analyses, *Flex Serv Manuf J,* **29**, 369–402

Rekik I., Elkosantini S., Chabchoub H., (2016), Toward a knowledge based multi-agent architecture for the reactive container stacking in port terminals, *International Conference on Artificial Intelligence and Soft Computing*, Zakopane, Poland

Rekik I., Elkosantini S., Chabchoub H., (2018), A case based heuristic for container stacking in seaport terminals, *Advanced Engineering Informatics,* **38**, 658-669

Ries J., González-Ramírez R.G., Miranda P., (2014), A fuzzy logic model for the container stacking problem at container terminals, *Comput. Sci.*, **8760**, 93–111.

Saanen Y.A., Dekker R., (2006) Intelligent stacking as way out of congested yards? Part 1 and Part 2, *Port Technol. Int.*, **31,** 87–92 and **32**, 80-86

Sculli D., Hui C.F., (1988), Three dimensional stacking of containers, *OMEGA,* **16(6)**, 585–594

Stahlbock R., Voß S., (2008), Operations Research at Container Terminals: A literature update, *OR Spectrum,* **30(1)**, 1–52

Steenken D., Voß S., Stahlbock R, (2004), Container terminal operations and operations research - a classification and literature review, *OR Spectrum*, **26(1)**, 3–49

Taleb-Ibrahimi M., De Castilho B., Daganzo C.F, (1993), Storage space vs handling work in container terminals, *Transportation Research Part B*, **27(1)**, 13–32

Tapia F., Covarrubias R., Miranda P., González-Ramírez R.G., (2013), On the storage space allocation problem, *22nd International Conference on Production Research, ICPR*

TPA website Arica map, website of the Terminal Puerto Arica, *http://portal.tpa.cl/tpaweb/nuestras-ventajas/*, 2020 (accessed on the 1st of March 2020)

TPA website Yard Layout, website of the Terminal Puerto Arica, *http://portal.tpa.cl/tpaweb/nuestro-puerto/*, 2020 (accessed on the 1st of March 2020)

United Nations Conference on Trade and Development, (2018), *UNCTAD Annual Report*, United Nations.

Vis I.F.A., de Koster R., (2003), Transshipment of containers at a container terminal: An overview, *European Journal of Operational Research,* **147,** 1–16

Zadeh L.A. (1965), Fuzzy Sets, *Information and Control*, **8(3)**, 338–353

Zhang C., Liu J., Wan Y.W., Murty K.G., Linn R.J, (2003), Storage space allocation in container terminals, *Transportation Research Part B,* **37(10)**, 883–903

Zhen L., (2014), Container yard template planning under uncertain maritime market, *Transportation Research Part E,* **69**, 199-217