ONLINE STACKING POLICIES FOR CONTAINER STORAGE OPTIMIZATION

CEYHUN GÜVEN

MAY 2014

ONLINE STACKING POLICIES FOR CONTAINER STORAGE OPTIMIZATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
IZMIR UNIVERSITY OF ECONOMICS

BY

CEYHUN GÜVEN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE
OF
MASTER OF SCIENCE
IN
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

MAY 2014

Approval of the Graduate School of Natural and Applied Sciences

_____

(Prof. Dr. Cüneyt Güzeliş)

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of **Master of Science** in **Intelligent Production Systems** option of **Intelligent Engineering Systems**.

_____

(Assoc. Prof. Dr. Arslan Örnek)

Head of Department

We have read the thesis entitled **Online Stacking Policies for Container Storage Optimization** prepared by **Ceyhun GÜVEN** under supervision of **Assoc. Prof. Dr. Deniz TÜRSEL ELİİYİ**, and we hereby agree that it is fully adequate, in scope and quality, as a thesis for the degree of **Master of Science** in **Intelligent Production Systems** option of **Intelligent Engineering Systems**.

_____

Assoc. Prof. Dr. Deniz TÜRSEL ELİİYİ

Supervisor

Examining Committee Members:
(Chairman, Supervisor and Members)

Assoc. Prof. Dr. Deniz TÜRSEL ELİİYİ _____
Industrial Engineering Dept., IUE

Asst. Prof. Dr. Erdinç ÖNER _____
Industrial Engineering Dept., IUE

Asst. Prof. Dr. Kaan KURTEL _____
Software Engineering Dept., IUE

ABSTRACT

ONLINE STACKING POLICIES FOR CONTAINER STORAGE OPTIMIZATION

Güven, Ceyhun

M.Sc. in Intelligent Engineering Systems

Graduate School of Natural and Applied Sciences

Supervisor: Assoc. Prof. Dr. Deniz TÜRSEL ELİİYİ

May 2014, 60 pages

There are three crucial resources at container terminals; the yard, cranes and the vehicles. The main objective of the terminal is the efficient use of these resources while performing different operations. Containers are stacked on top of each other at the yard for efficient space utilization. However, stacking cranes can only directly access those containers at the top of the stack. As a result, reshuffling/shifting occurs, defined as an unproductive move of a container required to access another stored underneath. In this thesis, we focus on increasing the efficiency of the yard via consideration of the container stacking problem for export containers at a container terminal. Different online stacking policies are proposed and evaluated through simulation. The simulation models were run with real data obtained from the Port of Izmir, Turkey. The results in terms of four performance measures are compared with random stacking as a base case, and discussed thoroughly.

Keywords: Container stacking; export containers; container terminals; heuristics, simulation.

ÖZ

KONTEYNER DEPOLAMA OPTİMİZASYONU İÇİN ÇEVRİMİÇİ İSTİFLEME POLİTİKALARI

Güven, Ceyhun

Akıllı Mühendislik Sistemleri Yüksek Lisans Programı

Fen Bilimleri Enstitüsü

Tez Danışmanı: Doç. Dr. Deniz TÜRSEL ELİİYİ

Mayıs 2014, 60 sayfa

Konteyner terminallerinde konteyner sahası, vinçler ve araçlar olmak üzere üç önemli kaynak bulunmaktadır. Terminalin temel amacı değişik operasyonları gerçekleştirirken bu kaynakların etkin kullanımını sağlamaktır. Konteynerler konteyner depolama sahasını verimli bir şekilde kullanmak için birbirleri üzerine istiflenirler. Ancak istifleme vinçleri sadece istifin en üstündeki konteynerlere direkt olarak erişebilmektedir. Bunun sonucunda bir başka konteynerin altında depolanmış olan konteynere ulaşmak için gerekli olan yeniden elleçleme hareketi meydana gelir. Bu tezde konteyner terminallerindeki depolama sahası verimliliğini artırmak üzere ihraç konteynerler için konteyner depolama optimizasyonu problemi ele alınmıştır. Problem için farklı istifleme politikaları önerilmiş ve benzetim yolu ile İzmir Limanı'ndan alınan gerçek veri setleri ile çalıştırılmıştır. Sonuçlar dört farklı performans ölçütü üzerinden karşılaştırılmış ve ayrıntılı bir şekilde tartışılmıştır.

Anahtar Kelimeler: Konteyner istifleme; ihraç konteynerler; konteyner terminalleri; sezgisel; benzetim.

ACKNOWLEDGEMENTS

First of all, I would like to express my deepest gratitude to my supervisor Assoc. Prof. Dr. Deniz Türsel Eliiyi who has supported me throughout my thesis and provided overall guidance of my work on this research. I feel motivated and encouraged every time I attend her meeting. She has been a tremendous mentor for me. I wish to thank her for her guidance, contribution, useful comments, remarks and writing of this master thesis. Without her guidance and persistent help this master thesis would not have been possible.

I would like to express my gratitude to Asst. Prof. Dr. Erdinç Öner for both his support and contribution in my MSc years and being my committee member. I would also like to thank Asst. Prof. Dr. Kaan Kurtel and Asst. Prof. Dr. Zeynep Sargut for serving as my committee members.

I would like to thank my manager Murat Özemre and my assistant manager Güner Mutlu at Bimar Information Technology Services S.A. for their support in this study. I would also like to thank my ex-colleagues Cem Nacar and Osman Titiz for their help on coding.

A special thanks to my friends Eftal Pehlivan, Kaan Peker, Kadir Aksu, Serhat Şentürk and Alper Sertsu for their friendship. I must express my gratitude to İlknur Özbey, my girlfriend, for her continued support and love.

Last but not the least important, I owe more than thanks to my family: my parents Münevver and Kaşif Güven for giving birth to me, supporting me spiritually throughout my life and for their endless love. Without their support, it is impossible for me to finish my college and graduate education seamlessly. My brother, Recep Güven is always by my side. I am very lucky to have such a super brother.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

TERMS AND ABBREVIATIONS

ASC	**:** Automated Stacking Crane, used for stacking and removing containers at the storage yard.
Bays	**:** Row of seven container stacks placed end-to-end.
Berth	**:** Place on quay for mooring and service of a single vessel.
Lane	**:** A group of container storage positions consisting of twelve bays.
CFS	**:** Container Freight Station, warehouse facility where containers are packed and unpacked.
Container	**:** A reusable standard-sized metal box used for carrying general cargo.
Container yard	**:** Container stacking area of the container terminal.
Dwell time	**:** The time that a container remains in the container yard.
Gate	**:** The entrance point of trucks entering and leaving the container terminal.
Ground position	**:** The area required for stacking a container.
Reshuffling	**:** An unproductive move of a container required to access another container stored underneath. It is also called as restacking, shifting, and rehandling.
YTT	**:** Yard Terminal Truck, used for transferring containers within the terminal area.
Quay	**:** The area parallel or perpendicular to the shoreline, accommodating vessels.
QC	**:** Quay Crane, the crane located on the quay for the purpose of loading and unloading containers.
Stack	**:** The stack of containers in the yard.
TEU	**:** Twenty-foot equivalent unit.
Vessel	**:** General term for a ship.

CHAPTER 1: INTRODUCTION

Intermodal freight transportation is the transportation of products and raw materials from origin to destination by a sequence of at least two combinations of transportation modes such as land, rail or maritime transport (Crainic and Kim, 2007). Maritime transport is a favored mode of intermodal freight transportation, in which the goods are usually carried by containers. Therefore, this mode of transportation has increased remarkably over the last few decades (Steenken et al., 2004).

A container is a reusable standard-sized metal box that can be easily transferred between different modes of transportation, and is used for transporting products and raw materials between point of origin and point of arrival. Use of containers reduces the amount of product packaging and possibility of damage. The term twenty-feet-equivalent unit (TEU) refers to one container with a length of twenty feet. The term is also used to define the capacity of container vessels and container terminals. It is usually assumed that incoming containers belong to one of two sizes, namely 20-feet and 40-feet, as these are the most common sizes. A 20-feet container occupies 1 TEU in the storage area, while a 40-feet container occupies 2 TEUs.

Table 1 illustrates the upward trend in container traffic in the ports of Turkey during the years 2004-2013. It can be observed that about 8 million TEU were handled in 2013, and the container traffic in Turkey grew about 50% over the last 9 years. As a result of this remarkable increase container turnover, the number of, and competition between container terminals is rapidly increasing. The operations in leading container ports cannot be carried out in an efficient manner without the use of appropriate scientific methods. Therefore, over the last few decades, the use of

operations research tools and techniques became crucial in container ports for sustaining efficient operations and compatibility.

Table 1 TEU traffic in the ports of Turkey

Year	TEU Handled
2004	3.081.315
2005	3.610.830
2006	3.822.727
2007	4.699.529
2008	5.228.154
2009	4.520.786
2010	5.865.785
2011	6.613.035
2012	7.256.417
2013	7.962.930

Source: "TÜRKLIM Handling Figures." In *Port Operators Association of Turkey*. 29 May 2014.

1.1. Container Terminal Operations

A container terminal is an interim storage area where container vessels dock on berths, unload inbound containers and load export or transit containers. Terminals include storage yards for temporary storage of the incoming containers. Figure 1 illustrates a schematic representation of typical operations and equipment in container terminals, including quay cranes for loading and unloading of the docked vessels, trucks and trailers for carrying containers within the terminal area, and rubber mounted gantry cranes (RMG) for stacking containers in the storage yard.

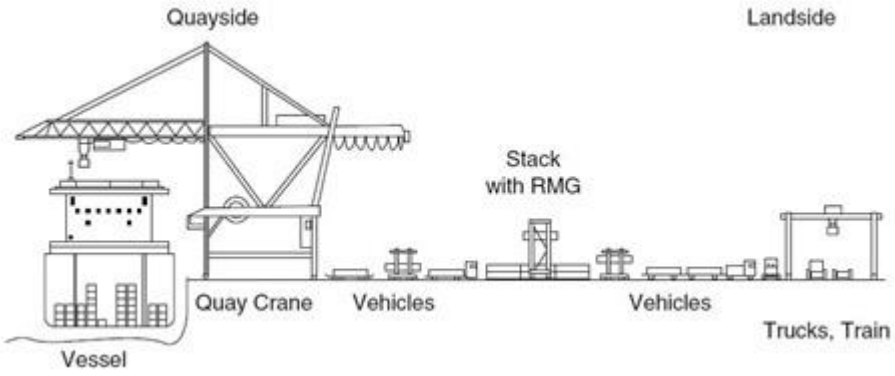

Figure 1 Schematic representation of a container terminal (Steenken et al., 2004)

Rashidi and Tsang (2013) categorized the decision problems in container terminals as in Figure 2. According to their classification, there are five main operations that affect efficiency and competitiveness.

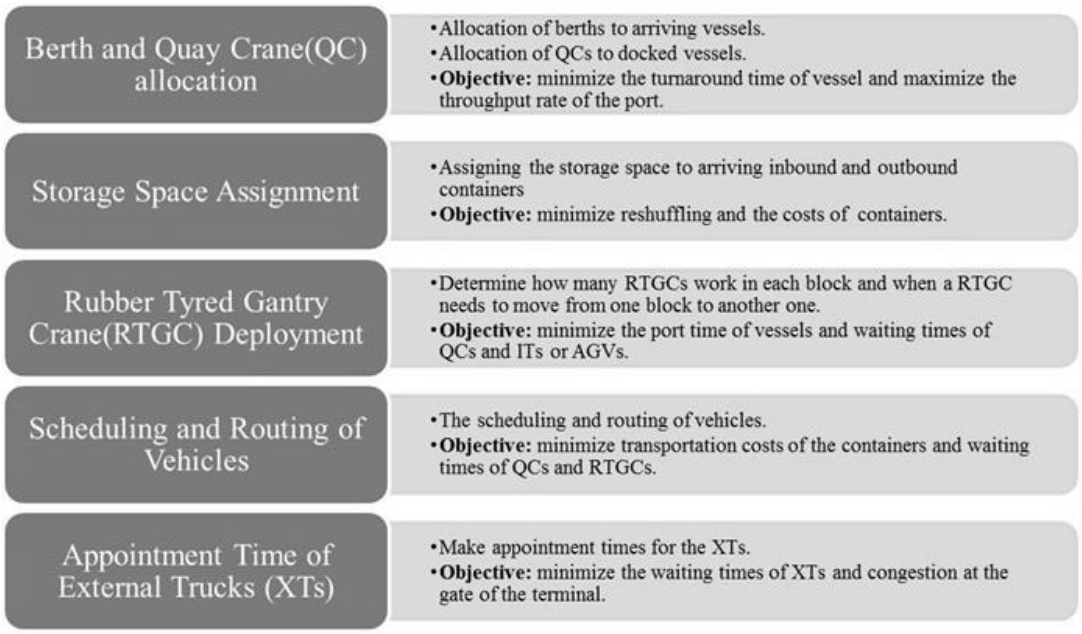

Figure 2 Decision problems in container terminals

Container terminals are places where containers are temporarily stored in storage yards and transshipped to the next location by truck, train or vessel. When a vessel arrives at a container terminal, it moors to the berth. Quay cranes (QCs) are allocated to the berths to unload import containers and load export containers. A QC takes a container from the vessel and places it onto the Yard Terminal Truck (YTT) that is waiting to transport it to the storage yard. Figure 3 indicates Quay Cranes unloading containers from vessel.

Figure 3 Quay Cranes (QCs) loading/unloading containers from vessel

Next, containers are transported from berths to the storage area by Yard Terminal Trucks (YTTs). A YTT takes the container from quay cranes and transfers it to its assigned block. Figure 4 shows examples of YTTs.

Figure 4 Yard Terminal Truck (YTT)

Different types of yard cranes exist for stacking and removing containers at the storage yard. Among these, Automated Stacking Cranes (ASCs) are rail-mounted cranes that move along the specified lane. They are used for stacking and removing containers from the stacks in the lane and are completely automated in all of their operations. Other types include Rubber-Tired Gantry or Rail-Mounted Gantry cranes (RTGs or RMGs). Once a container arrives at its stacking lane in the storage area, the yard crane lifts the container from the truck and stacks it to the storage position as shown in Figure 5. When necessary, the crane is also used to restack the containers. Containers are temporarily stored at their storage positions until their departure times.

Figure 5 Automated Stacking Crane (ASC)

1.2. Focus of the Master Thesis

In this study, we focus on storage space assignment, and study the problem of determining the stacking positions for export containers in the storage yard of a container terminal.

Containers are stacked on top of each other in order to utilize the yard efficiently. However, stacking cranes can only directly access those containers at the top of each stack. As a result, reshuffling/shifting occurs, which involves moving one or several containers on top of the stack to reach a container below. This unproductive move of containers is very costly, as well as having a negative effect on the operational efficiency of the container terminals in terms of crane and operator workloads.

We focus on increasing the efficiency of the yard via consideration of online stacking policies for container storage optimization for export containers at a container terminal. An online algorithm is one that receives a sequence of requests as they arrive, and performs an immediate action in response to each request. The objective of the problem in this thesis is to minimize container storage and retrieval times through avoidance of reshuffles, resulting in more efficient loading/unloading operations, balanced workload of cranes, less crane travel, and in turn minimizing the dwell time of containers. The main inputs are the size, weight, discharge port/location, destined vessel/vehicle of the container, and the expected departure time. The policies developed in this thesis for export containers are directly applicable to transit containers, as well.

The remainder of this thesis is organized as follows. Chapter 2 introduces a review of the literature on related previous work about storage and stacking logistics and yard crane scheduling. Chapter 3 introduces problems encountered in container terminals, and our problem definition. Online container stacking algorithms are proposed and discussed in Chapter 4. Computational results are presented in Chapter 5. Finally, conclusions and future works are discussed in Chapter 6.

CHAPTER 2: LITERATURE REVIEW

There are many studies in literature on different aspects of container terminal operations. The relevant work is summarized in this chapter.

Steenken et al. (2004) studied the main logistics operations at container terminals. They provided a wide-ranging review of methods for optimization of these operations. In their study, the operations at a container terminal were classified as berth allocation, stowage planning, crane scheduling, terminal transport optimization, and storage and stacking logistics, where storage logistics were examined under two classes: storage (yard) planning and scattered stacking. In storage planning, the location of each container in the storage yard is allocated and reserved before the vessel's arrival. Reservations for containers can be based on discharge port, container size and container weight, depending on the stacking policy. On the other hand, in scattered stacking, a container's location is not determined before the vessel's arrival. Instead, the storage location is determined in real time after the vessel berths. In a later study, Stahlbock and Voss (2008) concentrated on container terminal operations as an extension of the earlier review by Steenken et al. (2004).

Castilho and Daganzo (1993) focused on import container operations at marine terminals. Stack heights and stacking policy are influential in efficient retrieval of containers from stacks. The authors considered two strategies: the first aimed to keep all stacks at an equal height, while the second segregated containers according to arrival times. The height and the width of a bay were the significant variables that influenced the expected number of reshuffles needed to retrieve a container. Kim (1997) studied different stack configurations and estimated the expected number of reshuffles needed to retrieve a random container from the stack, and the total expected number of reshuffles to retrieve all the containers from the bay. Kim and Kim (1998) developed a cost model to determine the optimal amount

of storage space and the optimal number of transfer cranes for import containers. The space cost, the fixed cost of transfer cranes, the variable cost of transfer cranes and outside truck costs were considered in their model. A later study by Kim and Kim (1999) concentrated on the allocation of storage space for import containers using a segregation strategy. Stacking newly arrived containers on the top of containers that arrived earlier was not allowed in the segregation strategy. Storage locations were allocated for each arriving vessel in order to minimize the expected number of re-shuffles. Vessel arrivals, i.e. the arrivals of the import containers were assumed to be constant, cyclic, or dynamic. Each case was analyzed and appropriate solution methods were suggested.

Chen et al. (2000) provided an empirical study on yard operations at a Taiwanese container terminal. They quantified unproductive movements of containers undertaken in quay transfer operations for both discharge and loading operations. The number of shift moves was examined for each factor, i.e., the storage density, the volume of containers loaded and the volume of containers discharged. Decision rules for the location of export containers based on tonnage were derived by Kim et al. (2000). The weight distribution of containers was assumed to be known, and dynamic programming was used to determine the storage location of export containers in order to minimize the expected number of reshuffling moves for loading.

Zhang et al. (2010) studied on the optimizing the block size in container yards. In order to determine the block size at a container terminal they proposed several optimization models. These problems are minimizing the weighted total expected yard crane cycle time, maximizing storage capacity, and minimizing the weighted expected truck waiting time. Çelik (2013) developed a mathematical model for optimal storage of transit containers in the container yard. The model aims at minimizing the total vertical and horizontal transportation cost where vertical cost includes the cost of reshuffling, and uses an interval scheduling perspective for modeling the arrival and departure times of the vessels.

Simulation is a widely-used and appropriate tool for analyzing container stacking, as the problem involves uncertain arrival times. Especially for export containers, the arrival times are not known, and cannot be estimated most of the time.

That is why, the problem is handled as an online decision making problem, and different policies are developed. Our study introduces such policies for container stacking, and simulation is used for performance analysis. Stacking policies for containers in automated container terminals were also simulated by Dekker et al. (2006). Different stacking policies for containers were simulated and compared with a base case, in which the containers were stacked randomly. In category stacking, containers of the same category (e.g., weight class, destination, size of containers) were assumed to be interchangeable, and could be stacked on top of each other. Category stacking was found to have a much better performance than random stacking, although it includes very simple rules for stacking. They also took into account extra factors such as considering the workload of each automated stacking crane, horizontal distance travelled in the terminal, preference for ground locations, in order to increase the performance of the system.

Another simulation study for automated container terminals was made by Duinkerken et al. (2001). The model of a quay transport system using automated guided vehicles (AGVs) was integrated into the container stacking area in their study. During simulation, restacking or rehandling was necessary if one or more container was on the top of the desired container. Two methods of restacking were considered: In proactive stacking, operation occurred when the stacking crane was idle, which in turn could reduce the stack response time. On the other hand, in reactive restacking, the operation occurred at the time that the lower container is needed to be retrieved. Four different stacking policies were developed in the simulation model.

Saanen and Dekker (2006a, b) aimed to identify a set of rules to use stacking space more efficiently without incurring an increase in costs per move or a decrease in performance. For this purpose, a reference case was set and alternative cases were suggested. They considered a container terminal with RTGs, and simulated each movement of the YTTs and RTGs. They developed a simulation model based on stacking algorithms and compared several stacking policies.

In this thesis, we introduce new online policies for stacking export containers, and similar to the previous studies, we test the performance of these policies by comparing them with random stacking as a base case. The difference of our study

from previous ones, and in turn its contribution, lies in proposing a policy that considers multiple objectives while stacking containers. As opposed to the previous simulation-based studies, a heuristic approach is integrated into the simulation with the intention of obtaining better results. The details of the problem and the developed heuristics will be explained in Chapters 3 and 4.

CHAPTER 3: PROBLEM DEFINITION

Berth and quay crane allocation, storage space assignment, RTG deployment, scheduling, routing of vehicles and appointment time of external trucks are the five main decision problems that are encountered in container terminals that affect efficiency and competitiveness. In this study, we focus on storage space assignment, and study the online problem of determining stacking positions for export containers in the storage yard of a container terminal. In the rest of this chapter, the inputs, outputs, objectives, definition and the terms that are used will be explained.

The yard is a temporary storage area where containers remain until they are transported to their next location by truck, train or vessel. Containers are usually stored in multiple-level stacks for efficient usage of the storage area. However, stacking cranes can only directly access those containers at the top of the stack. As a result, reshuffling occurs, defined as an unproductive move of a container required to access another stored underneath. The numbers of reshuffles have a negative impact on the operational efficiency of container terminal in terms of cranes and operators' workloads. Therefore, while determining the stacking positions of the containers, the objective of the problem is defined as to minimize container storage and retrieval times through avoidance of reshuffles, resulting in more efficient loading/unloading operations, balanced workload of cranes, less crane travel, and in turn minimizing the dwell time of containers.

A container's position in the yard is denoted by its lane, bay, stack and tier identifiers. A lane is defined by its bays (length) and stacks (width). Stacking yards are usually divided into multiple lanes, each consisting of a number of bays. A bay is composed of several stacks of a certain size which called a tier, and holds containers of the same size. These definitions are illustrated in Figure 6.

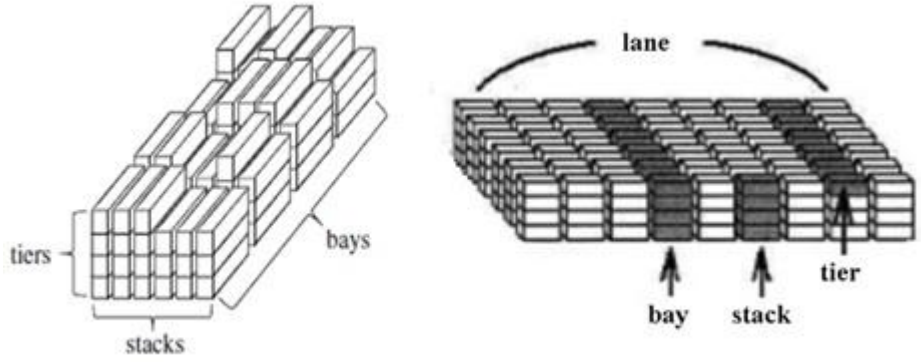

Figure 6 Illustration of a lane in the stacking yard (Source: Kim and Günther, 2005)

A storage position should be assigned to each container arriving at the container terminal (from the landside or seaside), and the minimum number of reshuffles is aimed while removing the container from its assigned location at its departure time. The assignment decision is based on various parameters like the container's port of destination/discharge, its arrival and departure time. This problem is called the Container Storage Optimization Problem (CSOP) or Container Stacking Problem (CSP), which is one of the most common and important problems at container terminals.

Mainly, four types of containers are stored at the yard, which are import, export, transit and coast-trade containers. Import containers are discharged from the vessels and then are stored until they depart from the port by rail or road transport. Export containers enter the port by rail or road transport, and they are stored in the stacking yard until they are loaded onto vessels. Transshipment or transit containers arrive with a vessel and depart with another vessel, while being stored in the stacking yard in the meantime. Finally, coast-trade containers are transferred from one of the Turkey ports to another one.

As we focus on the stacking of export containers to their storage positions in this thesis, the flow of this type of containers is analyzed in detail. Export containers can pass through different processes from origin until being loaded onto the vessels, which are shown in Figure 7.

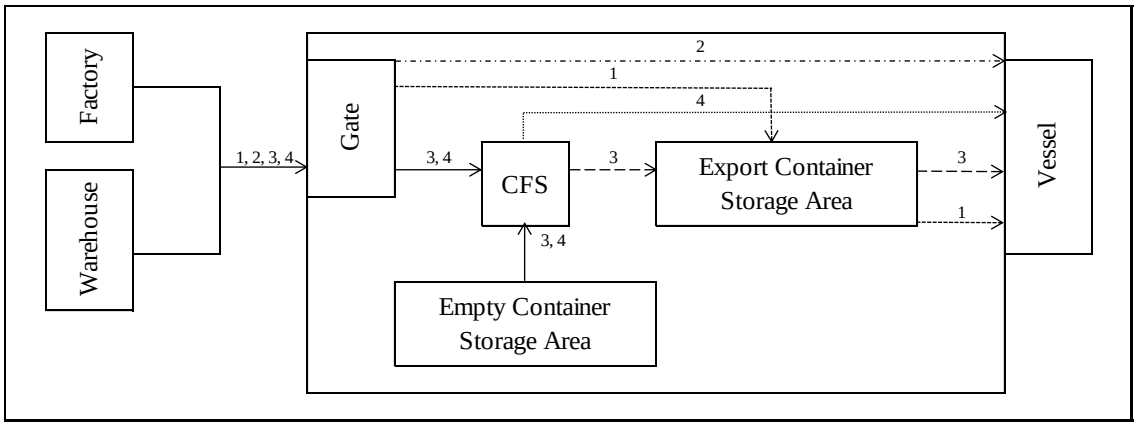

Figure 7 Export container flow

Different flows are possible based on the figure:

- An export container is brought to the container port from a factory or warehouse. The container is first stored at the yard until its destined vessel arrives at the port, and then it is loaded onto the vessel (1).
- An export container is brought to the container port from its origin of transport, and is directly loaded onto an awaiting vessel without any storage (2).
- While some containers enter the port in a loaded state (as above), some are filled at the port area. In such a case, the goods are brought to the port area from the origin of their transport. These goods are loaded into an empty container at the Container Freight Station (CFS), and then the filled container is stored in the export container storage area. Finally, this container is taken from the storage area and loaded onto the vessel (3).
- The last type of flow occurs if the container is loaded directly onto an awaiting vessel after being filled at the CFS (4).

The aim of the Container Stacking Problem (CSP) for export containers, inputs, outputs and the constraints are illustrated in Figure 8.

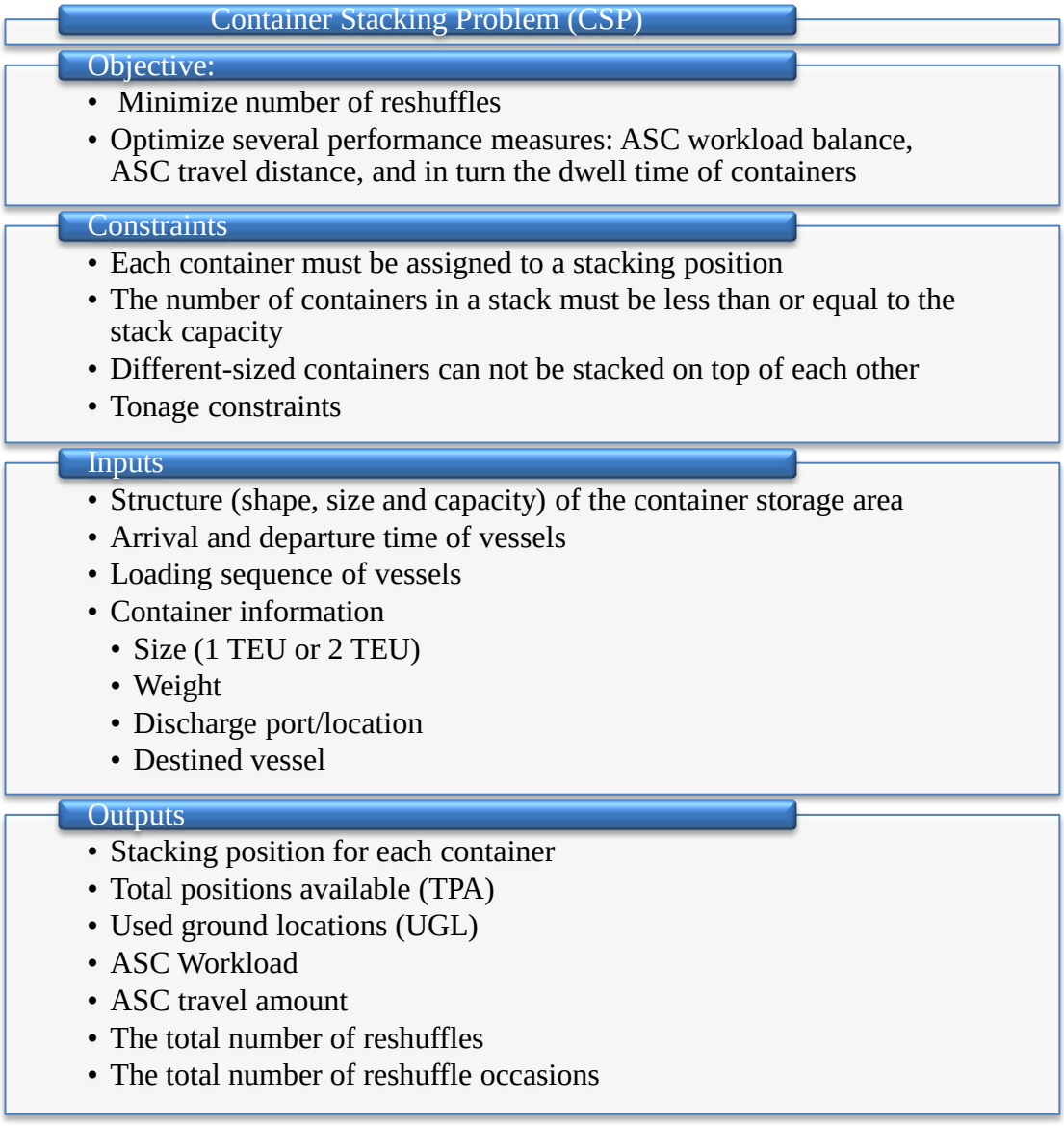

Figure 8 Container stacking problem specifications

A schematic overview of the container terminal considered in this study is provided in Figure 9. The layout of the terminal is in line with the layout of the Port of Izmir. There are 24 lanes, each equipped with a single stacking crane. Although the type of cranes can be changed very easily in simulation, we assume ASCs in our study as it involves the latest technology and is the fastest. As it can be seen from the figure, lanes are either horizontal or vertical to the waterfront, as three sides of the yard are covered with berthing positions.

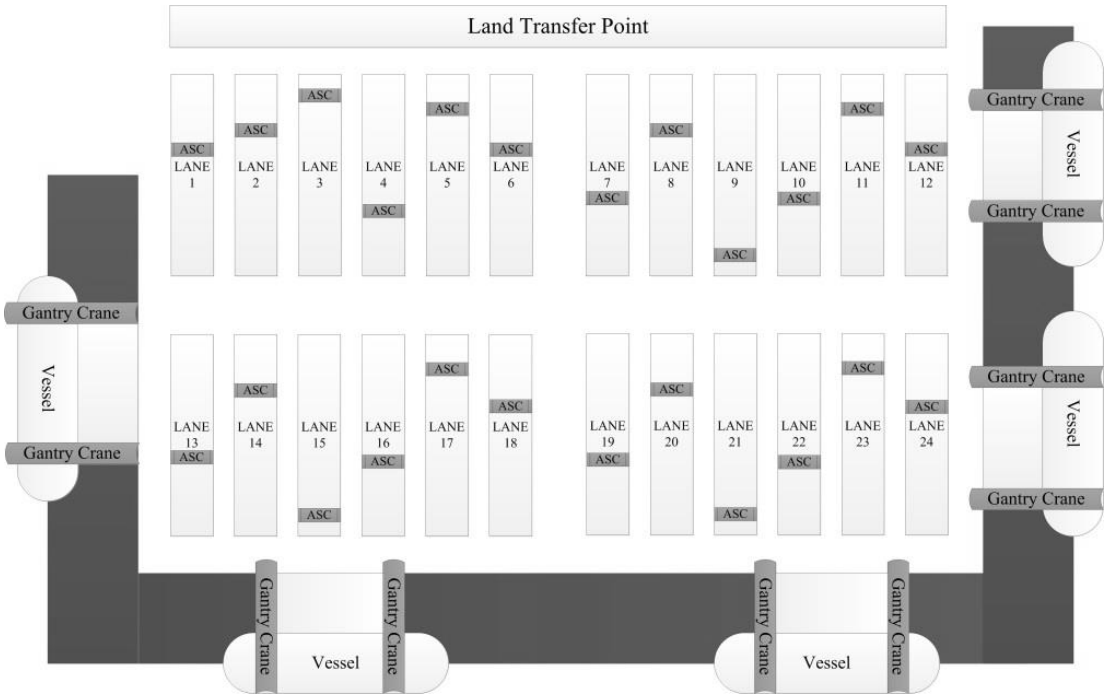

Figure 9 Schematic overview of the container terminal

A schematic overview of a single lane is presented on the next page in Figure 10. One bay of a lane consists of seven stacks. Each lane consists of twelve bays, six of which are reserved for 1-TEU containers and the other six for 2-TEU containers, summing up to 18 TEUs in length. Maximum stacking height is assumed as four containers for all lanes. Thus, the theoretical container capacity of the yard is 24 x 18 x 7 x 4 = 12,096 TEUs (twenty-feet-equivalent units), which correspond to 12,096 standard 20-feet containers.

The ASC is mounted on rails that are placed on both sides of the lane. One side is used for truck way, and all loading and unloading operations are made from this side. The movement of the ASC along the lane is counted as unproductive, as this move does not involve handling of a container. The movement of the operator cabin across the lane is for loading and unloading operations, as well as the vertical movement for handling. These movements are counted as productive.

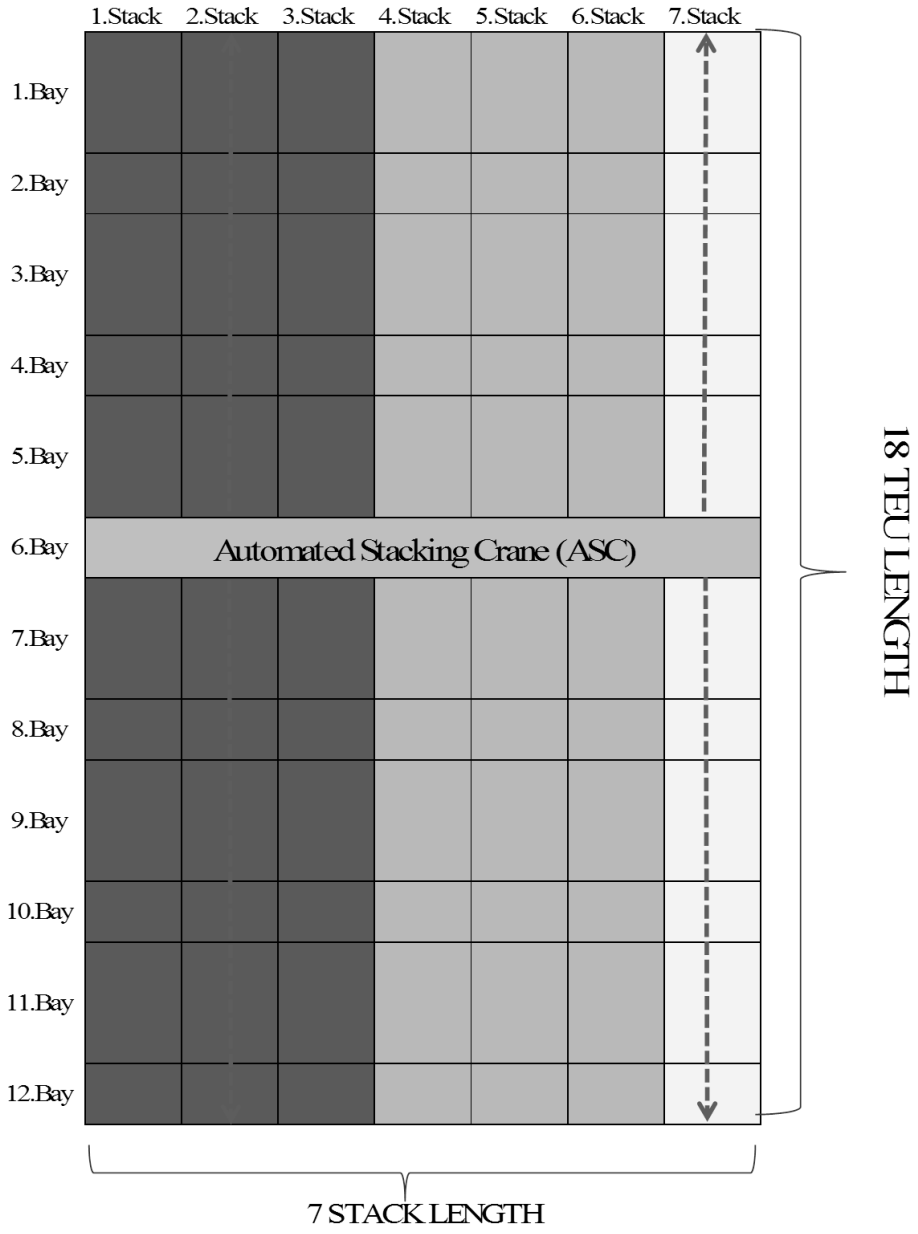

Figure 10 Schematic overview of a lane

CHAPTER 4: ONLINE CONTAINER STACKING POLICIES

Online container stacking policies and stacking heuristics are presented in this section. Container stacking policies involve these heuristic algorithms to determine the storage position of each individual container, considering several operational constraints and multiple performance objectives. In other words, each policy is used to determine where to store an arriving container.

We assume that the incoming containers belong to one of two sizes, namely 20-feet and 40-feet. A 20-feet container occupies 1 TEU in the storage area, while a 40-feet container occupies 2 TEUs. Containers of different sizes cannot be stacked on top of each other and cannot be stacked in the same bay due to physical restrictions, and to prevent possible damage. Yard Terminal Trucks (YTTs) transport the containers from the berths and land transfer points to the lanes and vice versa. The number of YTTs is assumed to be sufficient to cope with the transportation of the containers in the terminal area. Our solution methods for container storage optimization problem will be presented in the following sections.

4.1. Policy 1: Random Stacking

In this policy, we present a simplistic random stacking algorithm to be used as a base case in our comparisons.

The algorithm finds a position for an arriving container by randomly selecting a new lane, bay and stack. If the stack is not full, and the containers in the stack are of the same size as the arriving container, then an acceptable position has been found and the container can be placed in this stack. Else, a new random stack is considered.

The steps of the random stacking heuristic algorithm are:

Random Stacking:

1. Select a random position (lane, bay, and stack)
2. Check whether or not the selected bay size is the same as the arriving container's size. (20-feet or 40-feet).
    a. If the selected bay size ≠ size of the arriving container, it cannot be placed in this stack. Return to Step 1.
    b. If the selected bay size = size of the arriving container, check whether or not the selected stack is full.
        i. If full, return to Step 1.
        ii. If not full, stack the arriving container in this position.

4.2. Policy 2: Attribute-based Stacking

With this policy, we consider several attributes of the containers while determining a stacking position. The first attribute is the expected departure time (EDT) of the destined vessel of the container. Secondly, the destined vessel of each container is considered. Thirdly, the port of discharge (POD) order of the container is considered while stacking export containers of the same vessel, as it affects the order in which the containers are loaded onto the vessel. For example, let the POD of container A, which will leave on vessel B, is Hamburg. Furthermore, let Hamburg be the fifth stop of vessel B after leaving the port of Izmir. In such a case, the POD order of container A is determined as 5. This information helps to identify the best stacking location for a container. A container with a POD order 1 needs to be loaded last onto the vessel.

Finally, we consider the weight of the container as another attribute, and define an operational constraint that allows stacking containers on top of each other only if the weight range of the stack remains below 3 tons. This 3-ton rule, although very limiting, is in line with the current practice at the port and is used for comparison purposes.

In this policy, the containers are not stacked at randomly chosen positions. The developed heuristic algorithm finds positions for each container by searching

each bay and each stack in ascending order for feasible empty positions. According to the above considerations, note that an arriving container should be placed on top of a non-empty stack if it is destined for the same vessel, its POD order is less than the POD order of the containers already in the stack, and its weight does not violate the 3-ton constraint for the stack for avoiding any reshuffling. Based on this idea, we propose the following heuristic algorithm to determine the storage position of each arriving container.

Attribute-based Stacking:

1. Get the relevant information of the container: Container size (20 or 40 feet), destined vessel, weight, EDT, and POD order.
2. Check each bay of matching container size in ascending order and check whether the selected bay is empty or not.
    a. If empty, place the arriving container to the first tier of the first stack in ascending order.
    b. If not empty, check each stack in order in ascending order of the selected bay for empty positions.
        i. If stack empty, place the arriving container in this stack.
        ii. If not empty, check whether destined vessel of the arriving container is identical to the vessel of the containers in the stack, or not.
            1. If the vessel is identical, check if the POD order of the topmost container of the stack is greater than or equal to the POD order of the arriving container.
                a. If yes, check for the 3-ton constraint.
                    i. If satisfied, place the arriving container in this stack.
                    ii. If violated, return Step 2.b to check another stack of the selected bay.
                b. If no, return Step 2.b.
            2. If the vessel is not identical, check whether the EDT of the arriving container is earlier than or equal to the EDT of the topmost container of the stack or not.

a. If yes, check for the 3-ton constraint.
    i. If satisfied, place the container in this stack.
    ii. If violated, return Step 2.b to check another stack of the selected bay.
b. If no, return Step 2.b

4.3. Policy 3: Multi-objective Stacking

As it was stated before, as soon as an export container enters the container terminal from the landside, we have the following pertaining information:

- Arrival time
- Container identification number
- Size (1 TEU or 2 TEU)
- Container weight
- Destined vessel, POD order

Since the arrival time of the container is known after the arrival occurs, the problem should be handled in an online manner. Container identification number consists of a four-letters owner code (prefix), a six-digit serial number and a check digit. The owner code shows the company that owns the container. The check digit is used for identification of mistypes in the container number entry.

Containers of different sizes cannot be stacked on top of each other and in the same bay as it was stated before. Also, the weight of the container is considered as another attribute. However, as opposed to the 3-ton rule of the second policy, our third policy utilizes a more efficient and relaxed stacking approach regarding container weights. As we have learned from the port authorities, in vessel loading, the containers are mostly classified into one of the three weight groups as light, medium and heavy as shown in Table 2. And within each weight class, the containers with identical vessel and POD order are interchangeable when being loaded onto the vessel. Based on this classification, in our third policy, the containers can be stacked on top of each other only if their weight class is identical. Note that this rule is more

relaxed than the 3-ton rule, but is equally effective in preventing damage and avoiding any reshuffling moves when loading onto the vessel.

Table 2 Weight classification of containers

Weight Class	Corresponding weight
Light	< 15 tons
Medium	between 15 and 25 tons
Heavy	>25 tons

Based on the annual turnover of light, medium and heavy containers, most of the containers are identified as heavy and medium-weight. As a result, in policy 3, three stacks are reserved for heavy containers in each bay, while three are allocated for medium containers and 1 for light containers. The allocated stacks can be seen in Figure 10. 7 stacks of containers are placed side by side each with a height of 4 containers in each bay, and there are 12 bays in every lane. The truck way is assumed to be on the left side of the lane. The three closest stacks of each bay (which have the darkest shade) are for heavy containers. The reason that the closest stacks to the truck way are chosen for storing the heavy containers is to minimize the movement of the ASC across the lane with a heavy load with the intention of minimizing equipment depreciation. The next three stacks are for medium-weight containers, and the last stack that is farthest from the truck way is reserved for light containers.

In our multi-objective stacking policy, a composite score is computed for each available position that is feasible for stacking the incoming container. Feasibility is determined by the size of the container as well as the destined vessel and the POD order. That is, a feasible available position is a position that has the same size, vessel and POD order attributes as the incoming container. The online stacking rule of the heuristic then assigns the container to the position having the highest score. Many positions may be available in several different lanes; all are included in the algorithm.

The composite score is computed by considering four different objectives, and in turn four distinct performance measures. First objective is to minimize the total travel distance of the ASC along the lane when a container is entering or leaving

the lane. Second objective tries to balance the workload of the ASCs in order to utilize this expensive equipment. Third objective tries to assign each incoming container to a stack of proper weight class as much as possible. The final objective tries to utilize the storage yard by preferring to place the arriving container in non-empty stacks rather than empty ones.

The first objective aims at the efficient use of the stacking cranes. ASCs or stacking cranes in general, are the most frequently used equipment in container terminals for container handling. The efficiency of yard operations depends on the productivity of these ASCs. In our study, retrieval and stacking operations within each lane is performed by a single ASC, and the movement of an ASC along the lane takes time. Therefore, the distance that the ASC traverses along the lane is crucial as this unproductive time of travel affects the utilization performance.

Based on this observation, our first objective is to minimize the total travel distance of the ASC along the lane when a container is entering or leaving the lane. In order to employ this objective, the heuristic assigns a *Crane Movement Score* to each available stacking position based on the distance (in terms of traversed number of bays) that the associated ASC takes. These scores are shown in Table 3. Based on this scoring system, for example if a position is available in a lane for storing an incoming container, and the ASC of that lane is currently 10 bays away from that position, the associated *Crane Movement Score* of that position becomes 20. If more than one feasible position is available for stacking the incoming container in different lanes, all of them receive a *Crane Movement Score*, by considering the movement of the crane in their own lanes.

Table 3 Crane Movement Scores

Distance Traveled by ASC (in # of bays traversed)	Crane Movement Score
0	100
1	92
2	84
3	76
4	68
5	60
6	52
7	44
8	36
9	28
10	20
11	12
12	4

The ASCs are crucial equipment for the overall performance of the container terminal, and one of the most important performance measures of a container terminal is therefore ASC workload. Our second objective tries to balance the workloads of the ASCs in order to utilize this expensive equipment in a balanced manner to maximize overall system reliability. The hourly ASC workload is defined as the number of retrieval or stacking movements of the ASC during the last one hour. Each retrieval or stacking movement increases the workload of the corresponding ASC by one unit.

Based on this workload measure, the heuristic algorithm assigns an *ASC Workload Score* to each feasible and available position, considering the workload of the associated ASC, as in Table 4. An ASC workload of zero means that the ASC did not handle any container in the last hour, in which case any position in that lane receives a corresponding perfect score of 100. The practical handling capacity of an ASC is given as 15 containers/hour.

Table 4 ASC Workload Scores

# of Containers Handled	ASC Workload Score
0	100
1	93
2	86
3	79
4	72
5	65
6	58
7	51
8	44
9	37
10	30
11	23
12	16
13	9
14	2
15	0

The third objective tries to assign each incoming container to a stack of proper weight class as much as possible. As stated before, and as it can be seen from Figure 10, light containers should be stacked in the 7^{th} stack of the lane as much as possible, while medium-weights should be assigned to the 6^{th}, 5^{th} and 4^{th} stacks, and heavy ones should be stacked in the 3^{rd}, 2^{nd} and 1^{st} stacks.

In order to maximize proper stacking according to this policy, the heuristic algorithm assigns a *Weight Score* to each feasible and available position as shown in Table 5. Based on this scoring system, for instance, if an arriving container is of the light weight class, an available position in stack 7 will receive a score of 100 whereas another available position in stack 1 will receive a score of 50. On the other hand, an incoming container of heavy weight will result in a score of 100 for an available position in stack 1.

Table 5 Weight Scores

Weight Class	Stack ID	Weight Score
Light	7	100
Light	6	50
Light	5	50
Light	4	50
Light	3	50
Light	2	50
Light	1	50
Medium	7	15
Medium	6	100
Medium	5	100
Medium	4	100
Medium	3	50
Medium	2	50
Medium	1	50
Heavy	7	15
Heavy	6	50
Heavy	5	50
Heavy	4	50
Heavy	3	100
Heavy	2	100
Heavy	1	100

The final objective of this policy tries to utilize the storage yard in an efficient manner by preferring to place the arriving container in non-empty stacks rather than empty ones. This objective also aims to increase the number of empty ground locations. For this purpose, the heuristic algorithm assigns a Stacking Score to each feasible and available position by considering the current tier level of the stack at that position. Based on the scores that are shown in Table 6, an available position that has 3 containers will receive a perfect score of 100, as the incoming container will fill top this stack, and it will not be available anymore. In contrast, an empty available position will receive a score of 0, as assigning the arriving container in this position will decrease the number of empty ground locations by one. In-between tier levels are not differentiated by the algorithm, and receive a medium score of 50.

Table 6 Stacking Scores

Tier Level of Stack	Stacking Score
3	100
2	50
1	50
0	0

Once a feasible and available position receives scores from all objectives, a composite score is computed as the convex combination of these scores by the heuristic. Note that different weights in the convex combination will lead to focusing on different (conflicting) objectives, and therefore different assignments. In this respect, several weight combinations can be tried. Finally, the heuristic algorithm compares the composite scores of all feasible positions, and selects the highest one to assign the incoming container.

A simulation experiment is carried out to assess the performance of the developed policies. The results are presented in the next chapter.

CHAPTER 5: COMPUTATIONAL RESULTS

A simulation model was developed for evaluating the performance of the policies presented in the previous chapter. A computational experiment was designed and carried out for this purpose.

To be used in the experiment, real data pertaining to export container arrivals and departures is gathered for the Port of Izmir, Turkey from Bimar Information Technology Services S.A. The data include container identification number, arrival time, weight and size of container, destined vessel, expected departure time and port of destination information. Arrival times of containers are not stored. For this reason, the arrival times are generated from the exponential distribution, and randomness in data follows. While generating data, two constraints were applied. First, the arrival time of a container cannot be later than its departure time. Second, the difference between departure time and arrival time cannot be larger than 15 days. The data set contains 11 vessels and 1300 containers that will arrive by a truck or train and depart from the terminal by a vessel, corresponding to approximately a week's duration in the actual system. The simulation model is run for ten replications for all incoming containers. The storage position of each container can only be determined after it arrives at the terminal. A warm-up period is used to build up the initial conditions of the storage yard by assuming 130 (%10 of total containers) containers' arrival. These containers are assumed to be stacked on the storage area and remain until their departure time.

The simulation models employing all three policies are coded with C# programming language on a Microsoft Visual Studio 2013 platform. The simulation is executed on a Core 2 Duo 3.4 GHz Win7 PC with 8-GB memory. The stacking methods change from policy to policy. There are different methods used for container stacking, container retrieving, etc. in the online stacking heuristics, as

explained in the previous chapter. The codes for container stacking methods for Policies 1, 2 and 3 are provided in Appendix 1.

The simulation model was verified and validated. A statistical analysis was performed to ensure that 10 replications were sufficient, and the sufficiency was validated for all policies (See Appendix 2). All replications were finished successfully. All incoming containers leave the system by a vessel. Each container is moved at least two times; once for stacking and at least once for retrieval. If there is a reshuffle move for that container, there are more than two moves. Similarly, only one storage position is assigned to a single container. No containers could be assigned to a storage position until the container in that position is removed.

Verification is a check of having built the model correctly. It assures that the conceptual model is reflected accurately in the simulation model. That is, verification answers to the following question: Is the conceptual model accurately represented by the operational model?

Many steps are followed through the verification process:

1. The port authorities checked the operational model.
2. A flow diagram including each logically possible action was developed.
3. The model output for acceptability (reasonableness) was examined via comparing the number of arriving containers vs. the total ASC workload at the end of simulation, which was verified to be two times the total number of arriving containers plus the number of reshuffles.
4. The input parameters were printed at the end of the simulation, and they had not been changed inadvertently. These include the size, weight, destined vessel, POD information of the containers.
5. The debugger is an essential component of successful simulation model building. Simulation analysts may make mistakes and commits logical errors when building a model. The debugger helps in finding and correcting those errors. The built-in debugger of Visual Studio was used in checking the model.

Validation is the overall process of comparing the model and its behavior to the real system and its behavior. In this respect, validation checks whether the model

is an accurate representation of the real system. As an aid in our validation process, the three-step approach by Naylor and Finger (1967) was used. The steps are:

1. The simulation model that has high face validity was built for a real-world problem in container terminals of Turkey.
2. Model assumptions were validated:
    a. Structural assumptions:
        i. Containers of different sizes cannot be stacked on top of each other and cannot be stacked in the same bay.
    b. Data assumptions:
        i. Real data was gathered for the Port of Izmir, and the randomly generated arrival times were in line with the real data.
3. The model input – output transformations to corresponding input – output transformations for the real system were compared. The model was tested with a large group of data containing approximately 75,000 containers' arrivals. Therefore, all possible states were tested with this large data.

5.1. Computational Results

First, the simulation results for Policy 1 and Policy 2 are presented in Tables 7 and 8, for comparing the two policies in terms of several performance measures.

The first measure is the number of ground locations used for containers to be stacked. This measure illustrates how much of the total number 252 ground positions are used for stacking. Both policies use the same number of ground locations to place the containers, which is also equal to the total number of available ground locations, computed by 3 lanes, 7 stacks and 12 bays. Hence, it can be said that the yard is completely used, and no stack positions are left empty. However, this does not mean that the policies use all ground locations to the full height. There are 1008 positions available for the incoming containers at the yard in the simulation model. Note also that the exact stacking positions of the containers may be different with each policy.

The next measure is used for evaluating the overall travel distance of the ASCs along lanes when handling containers. As expected, Policy 2 outperforms random stacking in terms of this measure. The distance (in terms of number of

traversed bays) averaged over all replications and three ASCs decreases from 2124 to 1613 when attribute-based stacking is performed.

As expected, Policy 2 outperforms random stacking in terms of the number of reshuffles performed. The average number of reshuffling moves over all replications decreases by approximately 65% when attribute-based stacking is performed. This result is expected as attribute-based stacking seeks to minimize the number of reshuffles. In the simulation model, reshuffling moves are defined as unproductive moves when a container is to be removed from the stack. For instance, if the crane should reach container A, above which there is another container (B) in the stack, first container B is removed and temporarily put on another stack, then A is removed from the stack, and then B is put back to the top of its original stack. In this example, 2 reshuffling moves are made for handling container A. Obviously, the number of reshuffling increases if there are more containers on top of container A.

Table 7 Simulation results of Policy 1

Policy 1: Random Stacking	*Replications*									
Performance Measures	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Used Ground Locations (UGL)	252	252	252	252	252	252	252	252	252	252
ASC Travel Distance Along Lane	2087	2099	2186	2115	2125	2089	2175	2095	2141	2097
Number of Reshuffles Performed	1290	1086	1115	1010	894	972	1035	824	1069	1087
Number of Reshuffle Occasions	778	664	690	640	579	599	672	534	683	657
ASC-1 Workload	1380	1307	1196	1308	1184	1171	1152	1145	1210	1245
ASC-2 Workload	1239	1232	1298	1197	1167	1294	1264	1196	1251	1282
ASC-3 Workload	1269	1145	1219	1103	1141	1105	1217	1081	1206	1158

Table 8 Simulation results of Policy 2

Policy 2: Attribute-based Stacking	*Replications*									
Performance Measures	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Used Ground Locations (UGL)	252	252	252	252	252	252	252	252	252	252
ASC Travel Distance Along Lane	1745	1558	1836	1583	1496	1476	1795	1417	1575	1625
Number of Reshuffles Performed	479	461	365	390	283	277	323	229	476	399
Number of Reshuffle Occasions	267	275	218	262	183	172	196	164	291	264
ASC-1 Workload	1468	1340	1361	1173	1183	1262	1284	1337	1290	1326
ASC-2 Workload	869	954	854	916	909	940	952	852	979	934
ASC-3 Workload	740	765	748	899	789	673	685	638	719	737

The next measure counts the events of reshuffling rather than moves. A reshuffle occasion is said to happen when one or more reshuffling moves are required to retrieve a container from the stack. Note that for the above example,

number of reshuffling occasions is counted as 1, disregarding how many containers are on top of container A. It can be observed from the tables that Policy 2 performs better than random stacking in terms of the total number of reshuffle occasions, as this measure drops by approximately 65% on average when attribute-based stacking is used.

The workload of each ASC is also computed by the simulation model over the simulation period. The total number of handling moves, including reshuffling, is illustrated with this measure. Although the workloads of some ASCs seem to be higher with Policy 2 than Policy 1, the average workload over all ASCs and all replications is actually lower. This result will be clarified later in this chapter.

Next, the results for Policy 3 are presented. As this policy involves a multi-objective approach, the coefficients of the objectives in the convex combination are important for computing a composite score for each feasible position in the yard. In order to observe the effects of different coefficient schemes, 10 different scenarios are generated. The coefficients or weights of each objective under each scenario are presented in Table 9. One can observe that, while Scenario 1 assigns identical coefficients to all objectives, Scenario 2 favors minimizing ASC workload and Scenario 3 favors minimizing the crane movement. Several coefficient combinations are determined for simulation through interviews with the port authorities.

Table 9 Weights of objectives in 10 scenarios for Policy 3

	Crane Movement	**ASC Workload**	**Weight**	**Stacking**
Scenario 1	0.25	0.25	0.25	0.25
Scenario 2	0.1	0.7	0.1	0.1
Scenario 3	0.7	0.1	0.1	0.1
Scenario 4	0.1	0.1	0.1	0.7
Scenario 5	0.4	0.4	0.1	0.1
Scenario 6	0.1	0.4	0.4	0.1
Scenario 7	0.3	0.3	0.3	0.1
Scenario 8	0.1	0.1	0.7	0.1
Scenario 9	0.5	0.5	0	0
Scenario 10	0	0	0.5	0.5

The detailed results of the simulation runs for Policy 3 under the scenarios in Table 9 are presented in Tables 10 through 19. In these tables, the computed values for each of the aforementioned performance measures are listed for each replication.

Table 10 Simulation results of Policy 3 – Scenario 1

Policy 3: Multi-objective - Scenario 1	*Replications*									
Performance Measures	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Used Ground Locations (UGL)	252	252	252	252	252	252	252	252	252	252
Total Positions Available (TPA)	1008	1008	1008	1008	1008	1008	1008	1008	1008	1008
ASC Travel Distance Along Lane	1588	1364	1762	1333	1161	1275	1695	1199	1357	1313
Number of Reshuffles Performed	486	521	476	441	320	348	304	311	513	456
Number of Reshuffle Occasions	251	324	270	284	223	219	176	191	310	274
ASC-1 Workload	1169	1090	1190	1001	1029	1068	1135	1196	922	1091
ASC-2 Workload	1099	1083	1008	1015	1069	869	1007	938	1027	978
ASC-3 Workload	816	946	876	1023	820	1009	760	775	1162	985

Table 11 Simulation results of Policy 3 – Scenario 2

Policy 3: Multi-objective - Scenario 2	*Replications*									
Performance Measures	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Used Ground Locations (UGL)	252	252	252	252	252	252	252	252	252	252
Total Positions Available (TPA)	1008	1008	1008	1008	1008	1008	1008	1008	1008	1008
ASC Travel Distance Along Lane	1613	1303	1777	1340	1163	1252	1713	1210	1383	1311
Number of Reshuffles Performed	481	466	484	427	311	336	320	321	482	382
Number of Reshuffle Occasions	248	300	265	277	214	206	185	199	286	228
ASC-1 Workload	1086	1066	1013	950	1205	1102	1166	1070	958	1051
ASC-2 Workload	1026	1127	1093	999	871	964	903	946	1086	1048
ASC-3 Workload	967	871	976	1076	833	868	849	903	1036	881

Table 12 Simulation results of Policy 3 – Scenario 3

Policy 3: Multi-objective - Scenario 3	*Replications*									
Performance Measures	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Used Ground Locations (UGL)	252	252	252	252	252	252	252	252	252	252
Total Positions Available (TPA)	1008	1008	1008	1008	1008	1008	1008	1008	1008	1008
ASC Travel Distance Along Lane	1565	1308	1754	1297	1156	1217	1689	1199	1309	1272
Number of Reshuffles Performed	438	523	488	438	333	313	311	307	434	424
Number of Reshuffle Occasions	232	327	273	291	227	201	180	191	264	246
ASC-1 Workload	1220	1170	1202	1000	866	1143	1200	906	1006	1114
ASC-2 Workload	1117	1124	1022	989	1167	881	842	794	903	896
ASC-3 Workload	699	827	862	1047	898	887	867	1205	1123	1012

Table 13 Simulation results of Policy 3 – Scenario 4

Policy 3: Multi-objective - Scenario 4	*Replications*									
Performance Measures	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Used Ground Locations (UGL)	252	252	252	252	252	252	252	252	252	252
Total Positions Available (TPA)	1008	1008	1008	1008	1008	1008	1008	1008	1008	1008
ASC Travel Distance Along Lane	1601	1370	1759	1293	1171	1235	1685	1209	1340	1292
Number of Reshuffles Performed	496	534	480	481	338	300	317	316	420	386
Number of Reshuffle Occasions	257	330	269	314	233	195	179	194	258	233
ASC-1 Workload	1226	1128	1185	1050	1053	1049	1138	1077	922	1098
ASC-2 Workload	1096	986	967	1093	979	888	1014	947	980	946
ASC-3 Workload	772	1018	926	936	904	961	763	890	1116	940

Table 14 Simulation results of Policy 3 – Scenario 5

Policy 3: Multi-objective - Scenario 5	*Replications*									
Performance Measures	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Used Ground Locations (UGL)	252	252	252	252	252	252	252	252	252	252
Total Positions Available (TPA)	1008	1008	1008	1008	1008	1008	1008	1008	1008	1008
ASC Travel Distance Along Lane	1582	1328	1760	1303	1156	1251	1694	1203	1331	1312
Number of Reshuffles Performed	479	486	483	454	303	305	301	320	456	464
Number of Reshuffle Occasions	248	308	270	285	212	192	175	201	276	279
ASC-1 Workload	1163	1066	1207	973	900	1086	1133	941	1081	1094
ASC-2 Workload	1108	1047	987	963	1126	840	1009	1011	983	1027
ASC-3 Workload	806	971	887	1116	875	977	757	966	990	941

Table 15 Simulation results of Policy 3 – Scenario 6

Policy 3: Multi-objective - Scenario 6	*Replications*									
Performance Measures	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Used Ground Locations (UGL)	252	252	252	252	252	252	252	252	252	252
Total Positions Available (TPA)	1008	1008	1008	1008	1008	1008	1008	1008	1008	1008
ASC Travel Distance Along Lane	1618	1440	1805	1350	1153	1304	1679	1204	1353	1354
Number of Reshuffles Performed	493	625	475	514	340	428	336	338	492	571
Number of Reshuffle Occasions	254	380	269	331	236	256	187	211	302	329
ASC-1 Workload	1150	1092	1150	1030	1233	1100	1129	947	1042	1169
ASC-2 Workload	1071	1067	953	1092	914	1043	1001	963	947	1034
ASC-3 Workload	870	1064	970	990	791	883	804	1026	1101	966

Table 16 Simulation results of Policy 3 – Scenario 7

Policy 3: Multi-objective - Scenario 7	*Replications*									
Performance Measures	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Used Ground Locations (UGL)	252	252	252	252	252	252	252	252	252	252
Total Positions Available (TPA)	1008	1008	1008	1008	1008	1008	1008	1008	1008	1008
ASC Travel Distance Along Lane	1609	1355	1779	1341	1154	1254	1690	1198	1334	1342
Number of Reshuffles Performed	485	595	458	521	353	383	341	339	472	550
Number of Reshuffle Occasions	252	355	261	331	244	228	192	294	300	319
ASC-1 Workload	1141	1123	1181	1087	1086	1088	1156	1011	964	1112
ASC-2 Workload	1092	1150	936	939	977	872	1016	1140	1064	1056
ASC-3 Workload	850	920	939	1093	888	1021	767	786	1042	980

Table 17 Simulation results of Policy 3 – Scenario 8

Policy 3: Multi-objective - Scenario 8	*Replications*									
Performance Measures	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Used Ground Locations (UGL)	252	252	252	252	252	252	252	252	252	252
Total Positions Available (TPA)	1008	1008	1008	1008	1008	1008	1008	1008	1008	1008
ASC Travel Distance Along Lane	1607	1364	1795	1349	1153	1235	1690	1204	1351	1329
Number of Reshuffles Performed	499	608	484	524	359	300	341	333	476	522
Number of Reshuffle Occasions	257	363	274	339	251	195	192	203	298	317
ASC-1 Workload	1149	1074	1184	1043	1090	1049	1156	948	1001	1146
ASC-2 Workload	1116	1076	996	1064	981	888	1016	1015	1029	1064
ASC-3 Workload	832	1056	902	1015	886	961	767	968	1044	910

Table 18 Simulation results of Policy 3 – Scenario 9

Policy 3: Multi-objective - Scenario 9	*Replications*									
Performance Measures	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Used Ground Locations (UGL)	252	252	252	252	252	252	252	252	252	252
Total Positions Available (TPA)	1008	1008	1008	1008	1008	1008	1008	1008	1008	1008
ASC Travel Distance Along Lane	1587	1342	1755	1326	1151	1230	1680	1208	1332	1307
Number of Reshuffles Performed	495	492	494	445	322	283	306	316	432	422
Number of Reshuffle Occasions	255	304	275	292	224	181	178	200	269	259
ASC-1 Workload	1129	1101	1143	982	1014	1080	1136	935	865	1016
ASC-2 Workload	1152	1066	1004	986	1018	858	1010	1018	1019	1095
ASC-3 Workload	812	923	945	1075	888	943	758	961	1146	909

Table 19 Simulation results of Policy 3 – Scenario 10

Policy 3: Multi-objective - Scenario 10	*Replications*									
Performance Measures	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Used Ground Locations (UGL)	252	252	252	252	252	252	252	252	252	252
Total Positions Available (TPA)	1008	1008	1008	1008	1008	1008	1008	1008	1008	1008
ASC Travel Distance Along Lane	1582	1383	1752	1308	1160	1247	1694	1215	1331	1315
Number of Reshuffles Performed	470	530	487	460	330	316	305	320	498	444
Number of Reshuffle Occasions	245	329	268	300	227	193	178	197	297	276
ASC-1 Workload	1223	1331	1337	1064	1301	1371	1307	1368	1304	1110
ASC-2 Workload	1160	1305	1043	1264	1147	917	1017	1072	1080	1321
ASC-3 Workload	685	492	705	730	480	626	579	478	712	611

While the above tables can be used for investigating the value of each performance measure under each scenario and every replication of the simulation runs, the following two tables summarize the results from all scenarios. Table 20 compares the first two policies in terms of average, maximum and minimum values of the measures, and Table 21 does the same for all scenarios of Policy 3. The variation of measures between replications can be observed with this presentation.

Table 20 Summary of results for Policy 1 and Policy 2

Policies		*Used Ground Locations (UGL)*	*ASC Travel Distance*	*# Reshuffles*	*# Reshuffle Occasions*	*ASC Workload*
Policy 1	***Avg.***	252 / 252	2124	1041	651	1214
	Max	252 / 252	2186	1290	778	1380
	Min	252 / 252	2087	824	534	1081
Policy 2	***Avg.***	252 / 252	1613	366	229	988
	Max	252 / 252	1836	479	291	1468
	Min	252 / 252	1417	229	164	638

Table 21 Summary of results for Policy 3

Policy 3 Scenarios		*Used Ground Locations (UGL)*	*ASC Travel Distance*	*# Reshuffles*	*# Reshuffle Occasions*	*ASC Workload*
Scenario 1	***Avg.***	252 / 252	1405	418	252	1005
	Max	252 / 252	1762	521	324	1196
	Min	252 / 252	1161	304	176	760
Scenario 2	***Avg.***	252 / 252	1407	**401**	**241**	**1000**
	Max	252 / 252	1777	484	300	**1205**
	Min	252 / 252	1163	311	185	**833**
Scenario 3	***Avg.***	252 / 252	**1377**	**401**	243	**1000**
	Max	252 / 252	1754	523	327	1220
	Min	252 / 252	1156	307	180	699
Scenario 4	***Avg.***	252 / 252	1396	407	246	1002
	Max	252 / 252	1759	534	330	1226
	Min	252 / 252	1171	300	179	763
Scenario 5	***Avg.***	252 / 252	1392	405	245	1001
	Max	252 / 252	1760	486	308	1207
	Min	252 / 252	1156	301	175	757
Scenario 6	***Avg.***	252 / 252	1426	461	276	1020
	Max	252 / 252	1805	625	380	1233
	Min	252 / 252	1153	336	187	791
Scenario 7	***Avg.***	252 / 252	1406	450	278	1016
	Max	252 / 252	1779	595	355	1181
	Min	252 / 252	1154	339	192	767
Scenario 8	***Avg.***	252 / 252	1408	445	269	1014
	Max	252 / 252	1795	608	363	1184
	Min	252 / 252	1153	300	192	767
Scenario 9	***Avg.***	252 / 252	1392	**401**	244	**1000**
	Max	252 / 252	1755	495	304	1152
	Min	252 / 252	1151	283	178	758
Scenario 10	***Avg.***	252 / 252	1399	416	251	1005
	Max	252 / 252	1752	530	329	1371
	Min	252 / 252	1160	305	178	478

In the tables, the ASC workload is averaged over all cranes, while the minimum and maximum values over the replications are also presented.

Looking at the results in Table 20, one can see that Policy 1 and Policy 2 use the same number of ground locations, as it was stated before. While the ASCs traverse between 2087 and 2186 bays in random stacking, the variation is more with Policy 2; the values change between 1417 and 1836. The decrease in average traveling distance and the increase in variation are expected. Policy 1 selects a

random (uniform) position for stacking, resulting in higher distances of crane movement, and little variation among replications. However, as Policy 2 seeks to group containers, random arrivals result in high variation of crane movements in different replications.

The average number of reshuffles drops drastically under Policy 2, the decrease is approximately 65%. Similarly, the number of reshuffle occasions drop from 651 to 229, corresponding to again a 65% decrease. As a result of the dramatic decrease in reshuffling moves, the average workload of the ASCs decrease to 988 from 1214, this corresponds to a decrease of nearly 19%.

As an expected result, it is obvious that Policy 2 outperforms Policy 1 in every respect.

When Table 21 is analyzed, one can observe that different scenarios result in different values of performance measures. In the table, the best average values and variation in each performance measure are identified in shaded cells and with bold characters. Scenarios 2, 3 and 9 seem to be the dominating ones compared to the others. Recall that these scenarios include the following convex combination coefficients for the objectives:

	Crane Movement	**ASC Workload**	**Weight**	**Stacking**
Scenario 2	0.1	0.7	0.1	0.1
Scenario 3	0.7	0.1	0.1	0.1
Scenario 9	0.5	0.5	0	0

Scenario 2 favors the balanced workload objective, and performs best in this respect according to the results in Table 21. As well as yielding the lowest average workload among all scenarios, it yields the lowest difference between the workloads of the ASCs in each replication, as can be observed from Tables 10 to 19. The lowest average workload value of 1000 is also achieved by Scenarios 3 and 9, although their variations are higher.

As average workload is closely correlated with average reshuffling, the minimum average workload of 1000 containers is accompanied by the minimum reshuffling moves and occasions in Scenario 2. Very close results are obtained with

Scenario 3, as well, although a slightly higher average number of reshuffle occasions is output.

Scenario 3 also performs best in terms of crane movement, as expected. Scenario 9 is better than Scenario 2 in this respect, as it gives a higher weight to this objective. It can be concluded that Scenario 3 serves best in terms of the defined performance measures as a result of the simulation experiment for Policy 3.

The performances of Scenarios 2, 3 and 9 together with Policy 1 and Policy 2 in terms all performance criteria are charted in Figures 11 through 14.

When the results of Policy 2 and the best scenarios of Policy 3 are compared, nearly a 15% savings in crane movement is achieved with Scenario 3 over Policy 2. As the heuristic algorithm of attribute-based stacking does not explicitly consider crane movement, this result is expected. Although our simulation model does not explicitly consider handling times of containers, the decrease in crane movement implies a nearly-equivalent decrease in the overall handling time, and a higher number of containers handled per unit time. Therefore, it can be stated that containers are handled faster with Policy 3 than Policy 2.

However, Scenario 3 achieves this decrease at a cost of higher number of reshuffling moves and occasions when compared to Policy 2. Note that the sole focus of attribute-based stacking is on minimizing the number of reshuffling, whereas Policy 3 considers several objectives at once. For the same reason, the workload average of Policy 2 also seems better than Policy 3 on average. On the other hand, the workload balance of ASCs with Policy 2 is much worse than Policy 3. This result can be observed from Tables 20 and 21 in terms of average values, and from the previous tables for each replication and ASC.

In terms of computation time, all policies perform very similarly and the simulation run times are ignorable. Therefore we can conclude that the decision makers can use Policy 2 or Policy 3 to make their stacking decisions. None dominates the other when all performance measures are considered.

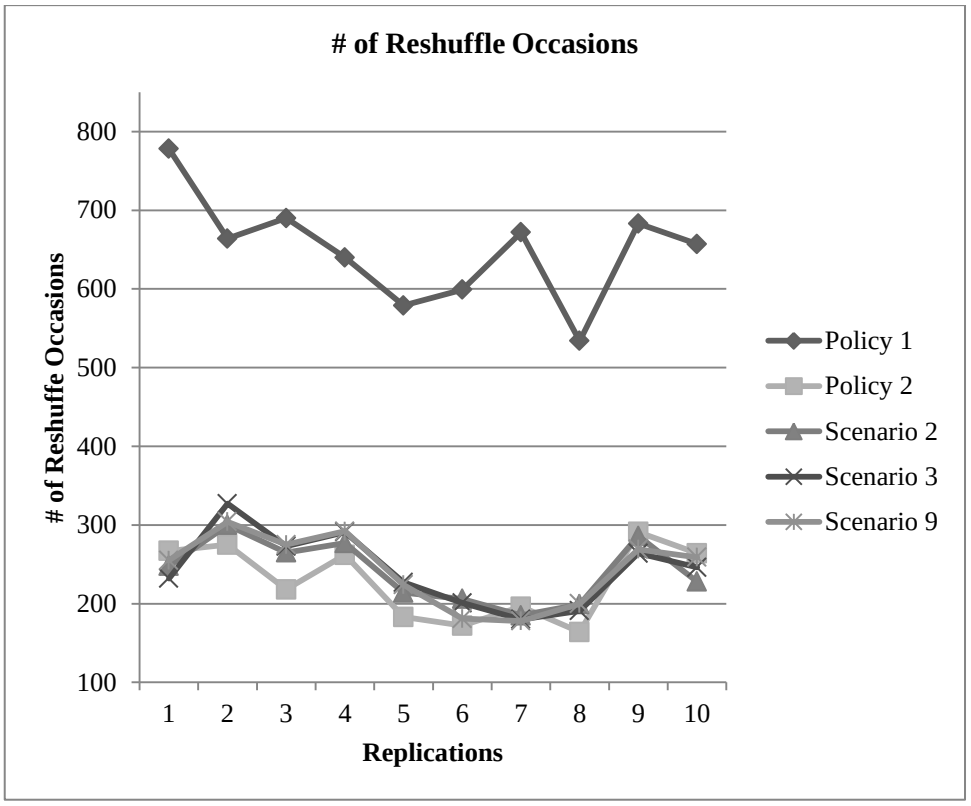

Figure 11 Number of reshuffle occasions

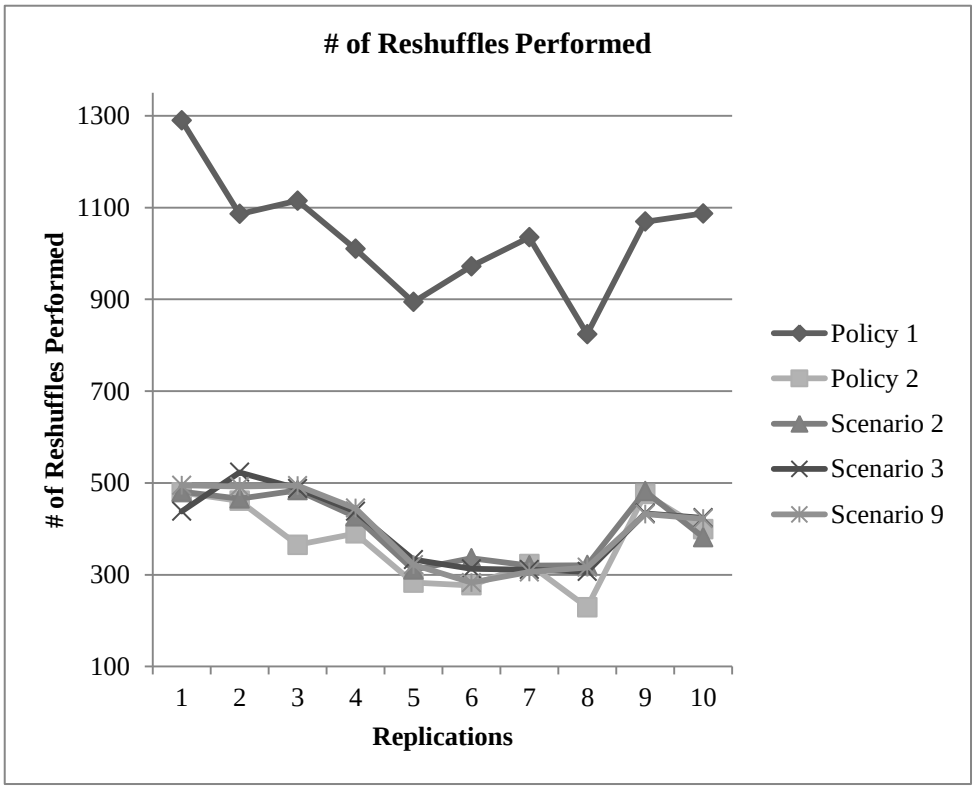

Figure 12 Number of reshuffles performed

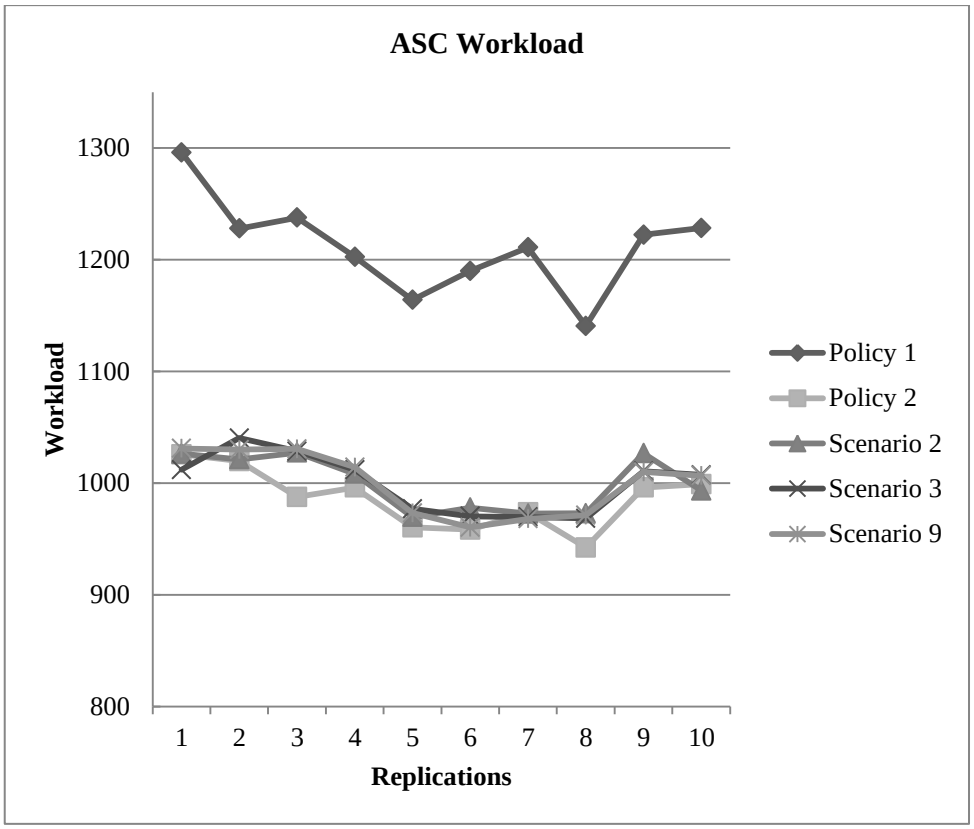

Figure 13 ASC workload

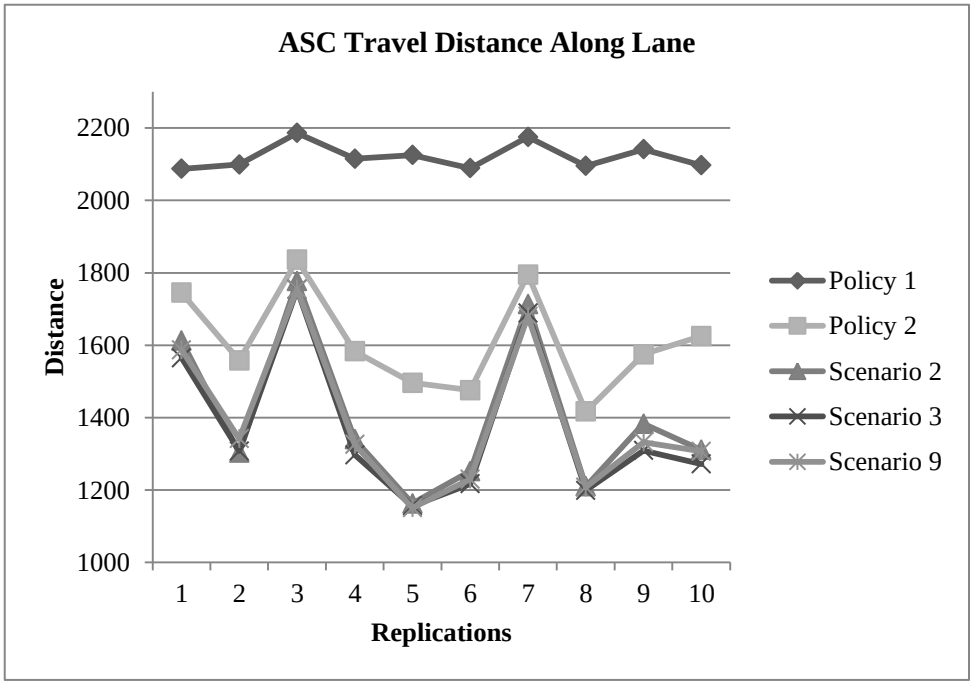

Figure 14 ASC travel distance along lane

5.2. Statistical Analysis

As there are multiple policies in our study, Bonferroni Approach to Multiple Comparisons is used for comparing the performances of the policies (Banks et al., 2010). The results and the discussion of the previous section are valid only when there is a statistically significant difference in the performance of different policies and scenarios. Therefore, we explain the Bonferroni procedure and present our results in this section. To apply the Bonferroni Approach, the following inequality is used.

Bonferroni Inequality: P (all statements S_i are true, i= 1, …, C) $\geq 1-\sum_{j=1}^{C}\alpha_j = 1-\alpha_E$,

where $\alpha_E = \sum_{j=1}^{C}\alpha_j$ is called the overall error probability.

This can be restated as:

P (one or more statements S_i is false, i= 1, …, C) $\leq \alpha_E$.

The Bonferroni approach to multiple confidence intervals is based on Bonferroni Inequality. An advantage of this approach is that it holds whether the models for the alternative designs are run with independent sampling or with common random numbers. On the other hand, the disadvantage of the Bonferroni approach in making a large number of comparisons is that the width of each confidence interval is increased. For instance, a given set of data and a large sample size, a 99.5% confidence interval will be $Z_{0.0025}/Z_{0.025} = 2.807/1.96 = 1.43$ times longer than a 95% confidence interval. For small sample sizes, a sample of size 5, a 99.5% confidence interval will be $t_{0.0025,4}/t_{0.025,4} = 5.598/2.776 = 1.99$ times longer than an individual 95% confidence interval. The width of a confidence interval is a measure of the precision of the estimate. Due to these reasons, it is suggested that the Bonferroni approach be used only if a small number of comparisons are being made. Twenty comparisons are the upper limit of the Bonferroni approach.

The Bonferroni Inequality is used when comparing alternative systems designs. It compares all designs to one specific design. First, a confidence interval is constructed for parameter $\theta_i - \theta_j$. The proper procedure to use depends on the goal of the simulation analyst. Some possible goals are the following:

- Estimation of each parameter, θ_i;
- Comparison of each performance measure θ_i to control θ_1 (where θ_1 could represent the mean performance of an existing system);
- All pairwise comparisons, $\theta_i - \theta_j$, for $i \neq j$;
- Selection of the best θ_i (largest or smallest).

There are at least three possible ways of using the Bonferroni Inequality when comparing *K* alternative system designs:

1. *Individual confidence interval:* This type of comparison is used to estimate multiple parameters of a single system. A $100(1-\alpha_i)\%$ confidence interval for parameter θ_i is constructed for (*i*=1, 2 … *K*). The number of intervals is *C* = *K*.
2. *Comparison to an existing system:* In this type of comparison, one system design especially the existing system design is compared by number of system designs. A $100(1-\alpha_i)\%$ confidence interval for $\theta_i - \theta_1$ is constructed for (*i*=1, 2 … *K*) using the inequality. θ_1 represents the mean performance of an existing system. The number of intervals is *C* = *K*-1. This type of procedure is often used to compare several system designs.
3. *All pairwise comparison:* In this type of comparison, all designs are compared to each other. A $100(1-\alpha_{ij})\%$ confidence interval for $\theta_i - \theta_j, i \neq j$, is constructed. The number of confidence intervals is *C* = *K* (*K*-1)/2.

In our model, there are four performance measures and 12 different stacking policies overall. Due to the insights from the previous section, we analyze the differences between policy 1 (P1), policy 2 (P2), policy 3 – scenario 2 (S2), policy 3 – scenario 3 (S3), and policy 3 – scenario 9 (S9) statistically. Since it is appropriate to use all pairwise comparison when comparing alternative system designs, the third alternative of Bonferroni approach mentioned above is performed.

For an overall confidence level of 95%, the overall error probability is $\alpha_E = 0.05$ and C=10 confidence intervals are to be constructed. Let $\alpha_i = 0.05/10 = 0.005$. Then, we use of Bonferroni inequality to construct 10 confidence intervals with 0.005, and degrees of freedom $\nu = 10 - 1 = 9$. The value of

$t_{\alpha_i,R-1} = t_{0.0025,9} = 3.69$ is obtained from the *t* distribution table. The point estimates and standard error calculations are included in Appendix 2.

The computed confidence intervals with an overall confidence coefficient of at least 95% are presented in Tables 22 through 25. ASC Workload, number of reshuffles performed and number of reshuffle occasions comparison tables have similar results. Notice that the confidence interval for P2-S2, P2-S3, P2-S9, S2-S3, S2-S9 and S3-S9 contain zero; thus, there is no statistically significant difference between P2-S2, P2-S3, P2-S9, S2-S3, S2-S9 and S3-S9. The other confidence intervals lie completely to the right of 0, indicating a statistically significant difference. For the last measure, notice that the confidence interval for S2-S9 and S3-S9 contain zero in Table 25; thus, there is no statistically significant difference between S2-S9 and S3-S9. All other confidence intervals lie completely to the right of 0, stating statistically significant differences.

The results of this section confirm that Policy 2 and Policy 3 are superior to Policy 1 in every measure. However, the differences between these two dominant policies do not seem to be statistically significant at the 95% confidence level. Only for the ASC travel distance measure, Policy 3 is proven to be superior to Policy 2, as the differences between P2-S2, P2-S3, and P2-S9 are found to be significant.

Once again, it can be concluded that P2, S2, S3 or S9 can be used by the decision makers to obtain good stacking solutions. For different daily needs and changing strategies of the port, the different policies may be preferred.

Table 22 Comparison of policies for ASC workload

Differences	Lower Bound	Upper Bound
P1-P2	199.89	252.51
P1-S2	178.97	246.03
P1-S3	174.91	250.09
P1-S9	179.76	245.44
P2-S2	**-32.14**	**4.74**
P2-S3	**-31.40**	**4.00**
P2-S9	**-29.91**	**2.71**
S2-S3	**-13.34**	**13.34**
S2-S9	**-12.23**	**12.43**
S3-S9	**-9.42**	**9.62**

Table 23 Comparison of policies for number of reshuffle occasions

Differences	Lower Bound	Upper Bound
P1-P2	363.39	477.41
P1-S2	337.79	479.81
P1-S3	326.43	486.37
P1-S9	333.08	478.72
P2-S2	**-43.81**	**20.61**
P2-S3	**-54.14**	**26.14**
P2-S9	**-46.86**	**17.86**
S2-S3	**-20.97**	**16.17**
S2-S9	**-21.68**	**15.88**
S3-S9	**-16.76**	**15.76**

Table 24 Comparison of policies for number of reshuffles performed

Differences	Lower Bound	Upper Bound
P1-P2	584.29	755.71
P1-S2	536.59	737.81
P1-S3	523.94	750.66
P1-S9	538.28	736.72
P2-S2	**-84.79**	**19.19**
P2-S3	**-93.80**	**28.40**
P2-S9	**-90.45**	**25.45**
S2-S3	**-39.93**	**40.13**
S2-S9	**-35.66**	**36.26**
S3-S9	**-28.46**	**28.86**

Table 25 Comparison of policies for ASC travel distance

Differences	Lower Bound	Upper Bound
P1-P2	370.20	650.40
P1-S2	488.41	940.39
P1-S3	522.46	966.14
P1-S9	511.70	946.50
P2-S2	97.93	310.27
P2-S3	129.15	338.85
P2-S9	121.86	315.74
S2-S3	3.14	56.66
S2-S9	**-12.95**	**42.35**
S3-S9	**-33.78**	**3.38**

CHAPTER 6: CONCLUSION AND FUTURE WORK

In this master thesis, we have proposed three online policies for stacking export containers in an automated container terminal. The objective of the problem is to minimize container storage and retrieval times through avoidance of reshuffles, resulting in more efficient loading/unloading operations, balanced workload of cranes, less crane travel, and in turn minimizing the dwell time of containers. We propose three stacking heuristics with a number of stacking policies. We employ these heuristics in a simulation model in order to test the performance of policies in terms of several performance measures.

Real data is obtained from the container terminal of Izmir, Turkey, and used as input for experimentation of the developed simulation model. The results are tabulated and discussed. Random stacking is used as a base case for comparison. The results indicate that the number of reshuffles can be reduced by 65% when attribute-based stacking is used. Several scenarios are tried for the multi-objective stacking policy regarding different coefficients for the objectives. The overall performance of this policy seems to be worse than attribute-based stacking in terms of reshuffling, as it also has different considerations such as balancing the workload of ASCs, ASC traveling distance and stacking order.

In terms of the total amount of crane movement, the third policy seems to be superior to the other two policies. We can thereby conclude that, when reshuffling is the single most important criterion, it is best to use the second policy. When other considerations come into picture, the third policy should be preferred by the decision makers. It can also be a good idea to switch between policies on a periodic or seasonal basis, as objectives and considerations may over time.

As most of the container terminals in Turkey currently make their container stacking decisions in a non-systematic manner, policies similar to the ones developed in this thesis can be very useful. The current practice at Izmir Port, as well as the majority of ports in Turkey, is based solely on operator experience. Moreover, many authorities claim that they operate in an optimal manner when stacking is concerned. This is partly because they are not aware of the techniques in literature, and partly due to the fact that they do not believe they could use the existing methods. Our study contributes to practice in this respect, by developing applicable policies that can be easily understood, modified and utilized by the decision makers.

A future research topic can include several different layouts with various types of stacking cranes to include uniform or unrelated equipment speeds into simulation. Layout and the number of cranes can affect reshuffling and workload measures. The simulation experiment can be performed with the data of a whole working year to obtain further results.

In addition, total utilization of the storage yard and the number of containers at the yard as a function of time can be used as performance measures. Also, if possible, each performance measure can be specified with its corresponding cost and the objective function can be to minimize the total cost of operations.

The policies developed in this thesis include simplistic rules for stacking the incoming export containers. The study can be extended to include import containers. Different policies apply to import containers, as the expected departure times (with trucks) cannot be estimated accurately. Developing a regression model for estimating the retrieval durations of different customers based on real data may be an effective approach. For transit containers, the policies in this thesis directly apply.

REFERENCES

Banks, J., Carson, J. S., Nelson, B. L., & Nicol, D. M. (2010). *Discrete-Event System Simulation, International Edition*. Upper Saddle River, New Jersey: Pearson Education.

Chen, T., Lin, K., & Juang, Y.-C. (2000). Empirical studies on yard operations Part 2: Quantifying unproductive moves undertaken in quay transfer operations. *Maritime Policy & Management*, 27(2), 191-207.

Cheung, R. K., Li, C. L., & Lin, W. (2002). Interblock crane deployment in container terminals. *Transportation Science*, *36*(1), 79-93.

Crainic, T. G., & Kim, K. H. (2007). Chapter 8 Intermodal Transportation. *Handbooks in Operations Research and Management Science*.

Çelik B. (2013). *A container storage problem in port operations*. M.S. Thesis, Izmir University of Economics, Izmir, Turkey.

De Castillo, B., & Daganzo, C. F. (1993). Handling strategies for import containers at marine terminals. *Transportation Research Part B: Methodological*, 27(2), 151-166.

Dekker, R., Voogd, P., & Asperen, E. (2006). Advanced methods for container stacking. *OR Spectrum*, *28*(4), 563-586.

Duinkerken, M. B., Evers, J. J. M., & Ottjes, J. A. (2001). A simulation model for integrating quay transport and stacking policies on automated container terminals. In *Proceedings of the 15th European Simulation Multiconference*, 909-916.

Hwan Kim, K., & Bae Kim, H. (1999). Segregating space allocation models for container inventories in port container terminals. *International Journal of Production Economics, 59*(1), 415-423.

Kim, K. H. (1997). Evaluation of the number of rehandles in container yards. *Computers & Industrial Engineering*. 32(4), 701-711.

Kim, K., & Günther, H. O. (2006). *Container Terminals and Cargo Systems*. Berlin: Springer.

Kim, K. H., & Kim, H. B. (1998). The optimal determination of the space requirement and the number of transfer cranes for import containers. *Computers & Industrial Engineering*, 35(3) 427-430.

Kim, K. H., & Kim, K. Y. (1999). An optimal routing algorithm for a transfer crane in port container terminals. *Transportation Science*, *33*(1), 17-33.

Kim, K. H., Park, Y. M., & Ryu, K.-R. (2000). Deriving decision rules to locate export containers in container yards. *European Journal of Operational Research, 124*(1), 89-101.

Linn, R., Liu, J. Y., Wan, Y. W., Zhang, C., & Murty, K. G. (2003). Rubber tired gantry crane deployment for container yard operation. *Computers & Industrial Engineering*, *45*(3), 429-442.

Narasimhan, A., & Palekar, U. S. (2002). Analysis and algorithms for the transtainer routing problem in container port operations. *Transportation science*, *36*(1), 63-78.

Naylor, T. H., & Finger, J. M. (1967). Verification of computer simulation models. *Management Science, 14*(2), B-92.

Ng, W. C. (2005). Crane scheduling in container yards with inter-crane interference. *European Journal of Operational Research*, *164*(1), 64-78.

Ng, W. C., & Mak, K. L. (2005). Yard crane scheduling in port container terminals. *Applied Mathematical Modelling*, *29*(3), 263-276.

Rashidi, H., & Tsang, E. P. K. (2013). Novel constraints satisfaction models for optimization problems in container terminals. *Applied Mathematical Modelling*, 37(6), 3601-3634.

Saanen Y. A. & Dekker R. (2006a). Intelligent stacking as way out of congested yards? Part 1. *Port Technology International*, 31, 87-92.

Saanen Y. A. & Dekker R. (2006b). Intelligent stacking as way out of congested yards? Part 2. *Port Technology International*, 32, 80-86.

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 operation and operations research-a classification and literature review. *OR spectrum*, *26*(1), 3-49.

TURKLIM Handling Figures. In *Port Operators Association of Turkey*. Retrieved May 29, 2014, from http://www.turklim.org/en/site/yukbilgileri.

Vis, I. F. (2006). A comparative analysis of storage and retrieval equipment at a container terminal. *International Journal of Production Economics*, *103*(2), 680-693.

Vis, I. F., & Carlo, H. J. (2010). Sequencing two cooperating automated stacking cranes in a container terminal. *Transportation Science*, *44*(2), 169-182.

Zhang, C., Chen, W., Shi, L., & Zheng, L. (2010). A note on deriving decision rules to locate export containers in container yards. *European Journal of Operational Research*, *205*(2), 483-485.

Zhang, C., Wan, Y. W., Liu, J., & Linn, R. J. (2002). Dynamic crane deployment in container storage yards. *Transportation Research Part B: Methodological*, *36*(6), 537-555.

APPENDICES

APPENDIX – 1.A. Container Stacking Method of Policy 1

```
public void KonteynerEkle(Container con)
{
    lastContainer = con;
    bool added = false;

    SetWorkLoad(con.ArrivalTime);

    int baytobeadded = 0;
    while (!added)
    {
        Random random = new Random();
        baytobeadded = random.Next(0, BayCount);

        if (baytobeadded == BayCount)
            break;
        var query = Bays[baytobeadded].Stacks.Any(x => x.Tiers.Any(y => y.Container == null)) && Bays[baytobeadded].Type == con.Type;

        if (query)
        {
            int stacktobeadded = random.Next(0, 7);

                for (int i = 0; i < 4; i++)
                {
                    if (Bays[baytobeadded].Stacks[stacktobeadded].Tiers[i].Container == null)
                    {
                        Bays[baytobeadded].Stacks[stacktobeadded].Tiers[i].Container = con;
                        added = true;
                        break;
                    }
                }
                if (added)
                {
                    break;
                }
            }

    }
```

```
            if (added)
            {
                if (LastBayId != baytobeadded)
                    ascTravelledDistance++;
                LastBayId = baytobeadded;
                SetAscWorkLoad(baytobeadded, con.ArrivalTime);
            }
```

APPENDIX – 1.B. Container Stacking Method of Policy 2

```
        public void KonteynerEkle(Container con)
        {
            lastContainer = con;
            bool added = false;
            int baytobeadded = 0;

            for (int x = 0; x < BayCount; x++)
            {
                if (Bays[x].Type == con.Type)
                {
                    var count = from b in Bays[x].Stacks
                                select (from a in b.Tiers
                                        where a.Container != null
                                        select a).ToList().Count;

                    if (count.ToList()[0] > 0)
                    {
                        var query = from b in Bays[x].Stacks
                                    where (from a in b.Tiers
                                           where (a.Container != null) ? (a.Container.Vessel == con.Vessel & a.Container.Destination ==
con.Destination) : false
                                           select a).ToList().Count > 0
                                    select b;

                        if (query.ToList().Count > 0)
                        {
```

```
                        bool check = false;
                        foreach (Stack s in Bays[x].Stacks)
                        {
                            if (added)
                                continue;

                            var sum = from b in s.Tiers
                                      where b.Container != null
                                      select b.Container.Weight;

                            foreach (Tier t in s.Tiers)
                            {

                                if (added)
                                {
                                    baytobeadded = x;
                                    continue;
                                }

                                if (t.Container == null)
                                {
                                    if (sum.Count() == 1 && (con.Weight >= sum.Sum() / sum.Count() - 3000) && con.Weight <= (sum.Sum() /
sum.Count() + 3000))

                                    {
                                        t.Container = con;
                                        check = true;
                                        added = true;
                                        continue;
                                    }
                                    else if (sum.Count() > 1 && ((sum.Min() + 3000) >= con.Weight && (sum.Max() - 3000) <= con.Weight))
                                    {
                                        t.Container = con;
                                        check = true;
                                        added = true;
                                        continue;
                                    }

                                    if (sum.Count() == 0)
                                    {
```

```
                                t.Container = con;
                                check = true;
                                added = true;
                                continue;
                            }
                        }
                    }
                }
            }
            else
            {
                Bays[x].Stacks[0].Tiers[0].Container = con;
                added = true;
                continue;
            }

            if (added)
                continue;

        }
    }
```

APPENDIX – 1.C. Container Stacking Method of Policy 3

```
    public Summary GetLocation(Container con, bool sameDestination)
    {
        var list = Bays.Where(x => x.Type == con.Type && x.Stacks.Any(y => y.Tiers.Any(t => t.Container != null && (!sameDestination ||
t.Container.Destination == con.Destination) )))
            .SelectMany(b => b.Stacks.SelectMany(y => y.Tiers.Where(t => t.Container == null)));


        if (list.Count() == 0)
            list = Bays.Where(x => x.Type == con.Type && x.Stacks.All(y => y.Tiers.All(t => t.Container == null)))
                .SelectMany(b => b.Stacks.SelectMany(y => y.Tiers.Where(t => t.Container == null)));
        ;
```

```
            int weightGroup = con.Weight > 25000 ? 3 : (con.Weight > 15000 ? 2 : 1);//Weight gruplarının tanımlanması.

            var list2 = (from tier in list
                         let tierCount = tier.Stack.Tiers.Count(y => y.Container != null)
                         let tierWeight = tier.Stack.Tiers.Where(y => y.Container != null).Sum(x => x.Container.Weight)
                         let weightScore = weightGroup == 1 ? (tier.Stack.Id == 7 ? 100 : 50) : (weightGroup == 2 ? (tier.Stack.Id == 7 ? 15
: (new List<long>() { 4, 5, 6 }.Contains(tier.Stack.Id) ? 100 : 50)) : (new List<long>() { 1, 2, 3 }.Contains(tier.Stack.Id) ? 100 :
(tier.Stack.Id == 7 ? 15 : 50)))
                         let emptyTier = tier.Stack.Tiers.FirstOrDefault(y => y.Container == null)
                         let tierScore = emptyTier.Id == 4 ? 100 : (emptyTier.Id > 1 ? 50 : 0)
                         let asc = GetAscByBayId(tier.Stack.Bay.Id)
                         let ascWorkLoadScore = 100 - asc.Works.Count * 7
                         let ascMoveScore = 100 - (Math.Sqrt(Math.Pow(asc.LastPosition - tier.Stack.Bay.Id, 2)) * 8)
                         where tier.Id == 1 ? true : tier.Stack.Tiers.Any(x => x.Id == tier.Id - 1 && x.Container != null)//possible
tierlarda konteynerlerin konulabilecegi yerleri filtreliyoruz
                         select new Summary()//filtrelenen tierların scorelari hesaplaniyor
                         {
                             BayId = tier.Stack.Bay.Id,
                             StackId = tier.Stack.Id,
                             TierId = tier.Id,
                             Container = tier.Container,

                             Point = ((0.3)*weightScore) + ((0.1)*tierScore) + ((0.3)*ascWorkLoadScore) + ((0.3)*ascMoveScore)
                         }).ToList();


            var bayAndStack = list2.OrderByDescending(x => x.Point).ThenByDescending(x => x.TierId).FirstOrDefault();//en yuksek puan almis
yerleri sıralıyor ve aynı puan almıs birden fazla yer varsa en üst katı tercih ediyor

            return bayAndStack;//konteyerin positionu donduruyor bay,stack,tier
        }

        public void KonteynerEkle(Container con)//sonraki butonun a bastıgımızda cagırılan konteyner
        {
            lastContainer = con;

            SetWorkLoad(con.ArrivalTime);//butun asc lerin suanki workloadlarını refresh ediyor

            var bayAndStack = GetLocation(con, true);
```

```
    if (bayAndStack == null)//uygun position bulunmazsa destination u goz ardı edıp aynı gemi ile gideceklere bakıyoruz
    {
        bayAndStack = GetLocation(con, false);
    }

    if (LastBayId != bayAndStack.BayId)
        ascTravelledDistance++;

    LastBayId = bayAndStack.BayId;

    Bays.Where(x => x.Id == bayAndStack.BayId)
        .SelectMany(x => x.Stacks.Where(y => y.Id == bayAndStack.StackId)
            .Select(z => z.Tiers.Where(t => t.Id == bayAndStack.TierId)).FirstOrDefault()).FirstOrDefault().Container = con;

    SetAscWorkLoad(bayAndStack.BayId, con.ArrivalTime);//ascson position guncelleme

    KonteynerGoster(false, false, false);

    UsedGroundLocation();
    AscTravelledDistance();
    StackUtilization();
    ReshuffleOccasion();
}
```

APPENDIX – 2 Statistical Analysis for Sufficient Number of Replications

Power and Sample Size

```
1-Sample Z Test

Testing mean = null (versus not = null)
Calculating power for mean = null + difference
Alpha = 0,05  Assumed standard deviation = 27,22
```

Difference	Sample Size	Target Power	Actual Power
800	2	0,95	1

Figure 15 ASC workload analysis on Policy 2

Power and Sample Size

```
1-Sample Z Test

Testing mean = null (versus not = null)
Calculating power for mean = null + difference
Alpha = 0,05  Assumed standard deviation = 212,85
```

Difference	Sample Size	Target Power	Actual Power
1100	2	0,95	1,00000

Figure 16 ASC travel distance analysis on Policy 3 – Scenario 3

Power and Sample Size

1-Sample Z Test

Testing mean = null (versus not = null)
Calculating power for mean = null + difference
Alpha = 0,05 Assumed standard deviation = 45,33

Difference	Sample Size	Target Power	Actual Power
150	2	0,95	0,996734

Figure 17 Number of reshuffle occasions analysis on Policy 3 – Scenario 9

Power and Sample Size

1-Sample Z Test

Testing mean = null (versus not = null)
Calculating power for mean = null + difference
Alpha = 0,05 Assumed standard deviation = 74,831

Difference	Sample Size	Target Power	Actual Power
300	2	0,95	0,999896

Figure 18 Number of reshuffles performed analysis on Policy 3 – Scenario 2

APPENDIX – 3 Statistical Comparison of Policies on Performance Measures

Table 26 Comparison of policies for ASC travel distance

	ASC Travel Distance Along Lane					Differences									
Replication	P1	P2	S2	S3	S9	P1 - P2	P1-S2	P1-S3	P1-S9	P2-S2	P2-S3	P2-S9	S2-S3	S2-S9	S3-S9
1	2087	1745	1613	1565	1587	342	474	522	500	132	180	158	48	26	-22
2	2099	1558	1303	1308	1342	541	796	791	757	255	250	216	-5	-39	-34
3	2186	1836	1777	1754	1755	350	409	432	431	59	82	81	23	22	-1
4	2115	1583	1340	1297	1326	532	775	818	789	243	286	257	43	14	-29
5	2125	1496	1163	1156	1151	629	962	969	974	333	340	345	7	12	5
6	2089	1476	1252	1217	1230	613	837	872	859	224	259	246	35	22	-13
7	2175	1795	1713	1689	1680	380	462	486	495	82	106	115	24	33	9
8	2095	1417	1210	1199	1208	678	885	896	887	207	218	209	11	2	-9
9	2141	1575	1383	1309	1332	566	758	832	809	192	266	243	74	51	-23
10	2097	1625	1311	1272	1307	472	786	825	790	314	353	318	39	4	-35
Sample mean	2120.90	1610.60	1406.50	1376.60	1391.80	510.30	714.40	744.30	729.10	204.10	234.00	218.80	29.90	14.70	-15.20
Sample variance	1278.77	19622.93	46715.17	45305.60	43177.73	14415.79	37507.38	36143.79	34710.54	8278.77	8074.00	6901.73	526.10	561.57	253.51
Standard error						37.97	61.24	60.12	58.92	28.77	28.41	26.27	7.25	7.49	5.03

LB	370.20	488.41	522.46	511.70	97.93	129.15	121.86	3.14	**-12.95**	**-33.78**
UB	650.40	940.39	966.14	946.50	310.27	338.85	315.74	56.66	**42.35**	**3.38**

Table 27 Comparison of policies for number of reshuffles performed

	Number of Reshuffles Performed					Differences									
Replication	P1	P2	S2	S3	S9	P1 - P2	P1-S2	P1-S3	P1-S9	P2-S2	P2-S3	P2-S9	S2-S3	S2-S9	S3-S9
1	1290	479	481	438	495	811	809	852	795	-2	41	-16	43	-14	-57
2	1086	461	466	523	492	625	620	563	594	-5	-62	-31	-57	-26	31
3	1115	365	484	488	494	750	631	627	621	-119	-123	-129	-4	-10	-6
4	1010	390	427	438	445	620	583	572	565	-37	-48	-55	-11	-18	-7
5	894	283	311	333	322	611	583	561	572	-28	-50	-39	-22	-11	11
6	972	277	336	313	283	695	636	659	689	-59	-36	-6	23	53	30
7	1035	323	320	311	306	712	715	724	729	3	12	17	9	14	5
8	824	229	321	307	316	595	503	517	508	-92	-78	-87	14	5	-9
9	1069	476	482	434	432	593	587	635	637	-6	42	44	48	50	2
10	1087	399	382	424	422	688	705	663	665	17	-25	-23	-42	-40	2
Sample mean	1038.20	368.20	401.00	400.90	400.70	670.00	637.20	637.30	637.50	-32.80	-32.70	-32.50	0.10	0.30	0.20
Sample variance	16308.84	7882.18	5599.78	6234.77	7279.79	5394.89	7433.96	9437.12	7229.83	1984.84	2742.01	2466.72	1176.99	949.57	603.29
Standard error						23.23	27.27	30.72	26.89	14.09	16.56	15.71	10.85	9.74	7.77

LB	584.29	536.59	523.94	538.28	**-84.79**	**-93.80**	**-90.45**	**-39.93**	**-35.66**	**-28.46**
UB	755.71	737.81	750.66	736.72	**19.19**	**28.40**	**25.45**	**40.13**	**36.26**	**28.86**

Table 28 Comparison of policies for number of reshuffle occasions

	Number of Reshuffle Occasions					Differences									
Replication	P1	P2	S2	S3	S9	P1 - P2	P1-S2	P1-S3	P1-S9	P2-S2	P2-S3	P2-S9	S2-S3	S2-S9	S3-S9
1	778	267	248	232	255	511	530	546	523	19	35	12	16	-7	-23
2	664	275	300	327	304	389	364	337	360	-25	-52	-29	-27	-4	23
3	690	218	265	273	275	472	425	417	415	-47	-55	-57	-8	-10	-2
4	640	262	277	291	292	378	363	349	348	-15	-29	-30	-14	-15	-1
5	579	183	214	227	224	396	365	352	355	-31	-44	-41	-13	-10	3
6	599	172	206	201	181	427	393	398	418	-34	-29	-9	5	25	20
7	672	196	185	180	178	476	487	492	494	11	16	18	5	7	2
8	534	164	199	191	200	370	335	343	334	-35	-27	-36	8	-1	-9
9	683	291	286	264	269	392	397	419	414	5	27	22	22	17	-5
10	657	264	228	246	259	393	429	411	398	36	18	5	-18	-31	-13
Sample mean	649.60	229.20	240.80	243.20	243.70	420.40	408.80	406.40	405.90	-11.60	-14.00	-14.50	-2.40	-2.90	-0.50
Sample variance	4555.38	2279.73	1605.51	2169.29	2055.12	2386.93	3703.73	4696.49	3894.54	762.04	1183.33	769.17	253.16	258.99	194.28
Standard error						15.45	19.25	21.67	19.73	8.73	10.88	8.77	5.03	5.09	4.41

LB	363.39	337.79	326.43	333.08	**-43.81**	**-54.14**	**-46.86**	**-20.97**	**-21.68**	**-16.76**
UB	477.41	479.81	486.37	478.72	**20.61**	**26.14**	**17.86**	**16.17**	**15.88**	**15.76**

Table 29 Comparison of policies for ASC workload

	ASC Workload					Differences									
Replication	P1	P2	S2	S3	S9	P1 - P2	P1-S2	P1-S3	P1-S9	P2-S2	P2-S3	P2-S9	S2-S3	S2-S9	S3-S9
1	1296	1026	1026	1012	1031	270	270	284	265	0	14	-5	14	-5	-19
2	1228	1020	1021	1040	1030	208	207	188	198	-1	-20	-10	-19	-9	10
3	1238	988	1027	1029	1031	250	211	209	207	-39	-41	-43	-2	-4	-2
4	1203	996	1008	1012	1014	207	195	191	189	-12	-16	-18	-4	-6	-2
5	1164	960	970	977	973	204	194	187	191	-10	-17	-13	-7	-3	4
6	1190	958	978	970	960	232	212	220	230	-20	-12	-2	8	18	10
7	1211	974	973	970	968	237	238	241	243	1	4	6	3	5	2
8	1141	942	973	968	971	199	168	173	170	-31	-26	-29	5	2	-3
9	1222	996	1027	1011	1010	226	195	211	212	-31	-15	-14	16	17	1
10	1228	999	993	1007	1007	229	235	221	221	6	-8	-8	-14	-14	0
Sample mean	1212.10	985.90	999.60	999.60	999.50	226.20	212.50	212.50	212.60	-13.70	-13.70	-13.60	0.00	0.10	0.10
Sample variance	1806.10	740.99	614.27	694.49	817.61	508.40	825.61	1037.83	791.82	249.79	230.01	195.38	130.67	111.66	66.54
Standard error						7.13	9.09	10.19	8.90	5.00	4.80	4.42	3.61	3.34	2.58
LB						199.89	178.97	174.91	179.76	**-32.14**	**-31.40**	**-29.91**	**-13.34**	**-12.23**	**-9.42**
UB						252.51	246.03	250.09	245.44	**4.74**	**4.00**	**2.71**	**13.34**	**12.43**	**9.62**