IMPROVED REHANDLING STRATEGIES FOR CONTAINER RETRIEVAL PROCESS

by
CENK AYDIN

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science

Sabancı University
August 2006

IMPROVED REHANDLING STRATEGIES FOR CONTAINER RETRIEVAL PROCESS

APPROVED BY

Assist. Prof. Dr. Tonguç Ünlüyurt …….
(Thesis Supervisor)

Assist. Prof. Dr. Bülent Çatay …….

Assoc. Prof. Dr. Meltem Denizel …….

Assist. Prof. Dr. Özgür Erçetin …….

Assist. Prof. Dr. Kemal Kılıç …….

DATE OF APPROVAL: …….

IMPROVED REHANDLING STRATEGIES FOR CONTAINER RETRIEVAL PROCESS

Cenk Aydın

Industrial Engineering, MS Thesis, 2006

Thesis Supervisor: Asst. Prof. Tonguç Ünlüyurt

Keywords:
Container terminal operation, Rehandling strategies

Abstract

Breakdown of trade barriers among countries exploded the volume of international trade. Consequently, amount of bulk cargo carried in containers and transported over seas exploded due to flexibility, reliability and easy handling. Containerization started in mid-fifties and spread all around the world. Number and capacities of containerships as well as container terminals have increased considerably. Only in the 90's usage of containers has increased 2.5 times. So efficiency in container terminals has become a major problem. Today container terminals serve ships that can carry 5000-6000 containers. Number of ships to be served and containers to be handled increase day by day. So efficiency during operations has become the key point, thus container terminals become perfect places for Operations Research applications.

In this work we deal with a low level operational problem in container terminals. Basically efficient retrieval of containers from their stacks is considered. We try to minimize number of container relocations and total distance traveled by the crane. Problem is solved optimally using branch and bound based procedure and alternative heuristics that give near optimal solutions are proposed. In addition, a new concept to further optimize the retrieval operation is introduced, formulated and tested.

KONTEYNIR ALIM SÜRECİNDE GELİŞMİŞ ELEÇLEME STRATEJİLERİ

Cenk Aydın

Endüstri Mühendisliği, Master Tezi, 2006

Tez Danışmanı: Yrd. Doç. Tonguç Ünlüyurt

Anahtar Kelimeler:
Konteynır terminal operasyonu, Eleçleme stratejisi

Özet

Ülkeler arası ticari engellerin kalkmasıyla uluslar arası ticaretin hacmi büyük ölçüde arttı. Buna parallel olarak dayanıklı olmalası ve taşıma ile eleçleme süreçlerinde esneği sayesinde deniz yoluyla taşımacılıkta konteynır kullanımı da önemli miktarda arttı.

Konteynırların kullanımı 50'lilerin ortalarında başlayıp dünyaya hızlıca yayıldı. Konteynır gemileri ile konteynır terminallerinin sayı ve kapasitelerinde de hızlı bir artış olmuştur. Sadece 90'larda konteynır kullanımında 2.5 katlık bir artış gözlemlenmiştir. Bugün konteynır terminallerinde 5000-6000 konteynır kapasiteli gemilere hizmet verilebiliyor. Bu gemilerinin sayıları ile büyüklükleri her geçen gün artması terminal operasyonlarında etkinlik ve verimliliği ön planda tutuyor. Bunedenle terminaller yöneylem araştırmaları için mükemmel bir ortam oluşturuyor.

Bu çalışmada alt derecede opereasyonel bir problem olan depolanan konteynırların alımlarında eleçleme sürecini ele aldık. Ve temel performans kriteri olarak eleçlenen konteynır sayısı ve vincin kat ettiği mesafenin minimize edilmesine odaklandık. Problemin optimalini dallan sınırla algoritmasıyla çözüldük ve alternatif sezgisel yöntemler önerdik. Bunun dışında problemin çözümü için yeni bir fikir olan temizleme hareketlerini tanımlayıp formulize ettik.

To my family

Acknowledgements

First and foremost, I would like to thank my supervisor Dr. Tonguç Ünlüyurt for his help with this work as well as my graduate study. He has always been understanding and supportive and given very good advice on any matter. I consider myself very lucky for working with him.

I owe many thanks to Dr. Bülent Çatay, Dr. Meltem Denizel, Dr. Özgür Erçetin and Dr. Kemal Kılıç for their helpful comments.

I also want to thank my fellows, Ahmet, Aydın, Ayhan, Bahar, Can, Emre, Ersin, Esat, Gürhan, Hatice, Ilkan, Mustafa, Tamer, Taner, Tevfik for their company and support namely. Also I would like to thank my dear friends, Ali, Bahadır, Canan, Mahmut, Onur, Şahbey, Volkan from the dormitory with whom I enjoyed a campus far away from Istanbul.

I thank Güler for her moral sport and help.

Although I don't know how to express my gratitude, I'll try anyway. Thanks to my family for their love and support. I know that they believe in my success by heart, that's why I'm so confident and hopeful about the future.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 Motivation

Containers are basically large boxes that are used for carrying goods and assess properties like; easy handling, hard structure for less damaging and most importantly globally standardized. As a result, containers become today's main unit in cargo transportation and usage is spread all around the world. Containerization Institute defines, containerization as "the utilization, grouping or consolidating of multiple units into a larger container for more efficient movement" [9]. Today two types of containers are used; 20 and 40 feet, and a 20 feet container is universally known as 1 TEU and 40 feet is 2 TEU.

First usage of containers was in mid-fifties, and through the years cargo handling with containers has rapidly increased. Only between 1990 and 2002 the growth of containerization increased by %250 percent throughout the world [11]. According to [21] the number of containers around the world will reach 491 million by 2012. This enormous increase in world's cargo brings needs for modern and highly efficient container terminals.

Today %25 percent of world's container traffic goes in Mediterranean, between Southeast Asia and Europe and main ports in the Mediterranean are Malta, Piraeus, Limasol, Larnaka, Alexandria, Damietta, Port Said, Haifa, Valetta, Ravenna, Gioia Tauro and Algeciras. Even though %91.4 of Turkey's foreign trade is conducted via sea transportation, none of the Turkish ports can play a major role in the Mediterranean container traffic [21]. As a consequence, in the last five years the ports have been privatized to increase Turkey's share in the sector.

Table 1.1 Port throughputs in 1000 TEU's.

	Turkey					Hong Kong
	Haydarpaşa	Mersin	Bandırma	İzmir	Marport	Kwai Tsing
2000	207.417	339.063	0.018	561.197	0	0
2001	226.471	384.146	0	547.218	0	11285
2002	130.38	191.916	0	282.169	469.505	11892
2003	193.894	0.374	328.621	1109.108	553.95	12070
2004	189.076	0.02	347.275	1158.207	769.656	13425
2005	236.016	0.019	399.908	604.768	791.029	14284

Table 1.1 [37], [29], [14] gives a summary of throughputs of main Turkish ports as well as the busiest port of the world, Hong Kong. Haydarpaşa, Mersin, Bandırma, İzmir ports are operated by the state and Marport is private. One can see that even total container throughput of Turkey is much smaller than Hong Kong's main terminal Kwai Tsing. Although the numbers are very pessimistic, the increase in Turkey's role in Mediterranean trade and privatization of the state ports are increasing attention to operations at container ports.

1.2 Container Terminals and Operations

Container can be transported over the seas or on land, and container terminals are facilities where the mode of transportation of containers is changed. Container terminals can be classified into two: automated and non-automated. Automated terminals are located where man power is costly like Western Europe and non-automated terminals operate in Southeast Asia, where labor is less expensive. Terminals, either automated or not, function similarly. Figure 1.1 represents the basic structure of the container terminals [35].

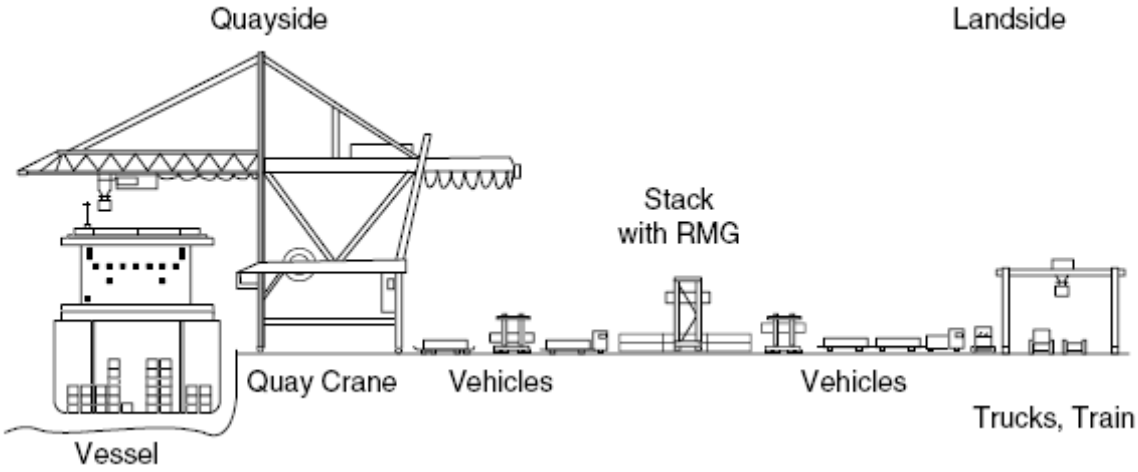


Figure 1.1 Schematic view of a container terminal

Three type of containers arrive at container terminals: export, import and transit. An import container comes in a vessel and stored in the terminal until it is transported by land. An export container comes by land and is stored until it is loaded on a ship. Transit containers are the ones which are discharged from one ship and uploaded on another. The order of operations differ according to the type of the container.

Operations for an import container start with the arrival of a ship. When a ship arrives at a port, a berth is assigned for unloading. After the ship is positioned on the berth, a necessary number of quay cranes is allocated to unload the ship (see Figure 1.2). When a container is unloaded by the quay crane it is loaded to a vehicle and transported to yard area for storage. At the storage area containers are stacked into blocks either with yard cranes or straddle carriers (see Figure 1.3 and Figure 1.4). Containers are stored in blocks until they are claimed by the importer. When a container is claimed it is loaded to a truck with a yard crane. Operations for the container end with its departure.

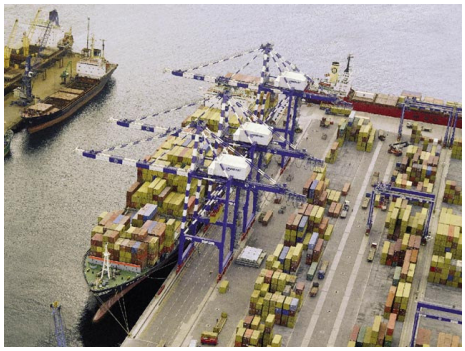


Figure 1.2 View of a berthed ship with 3 quay cranes assigned Marport

Figure 1.3 Straddle Carrier

Figure 1.4 A yard crane assigned to a block Marport

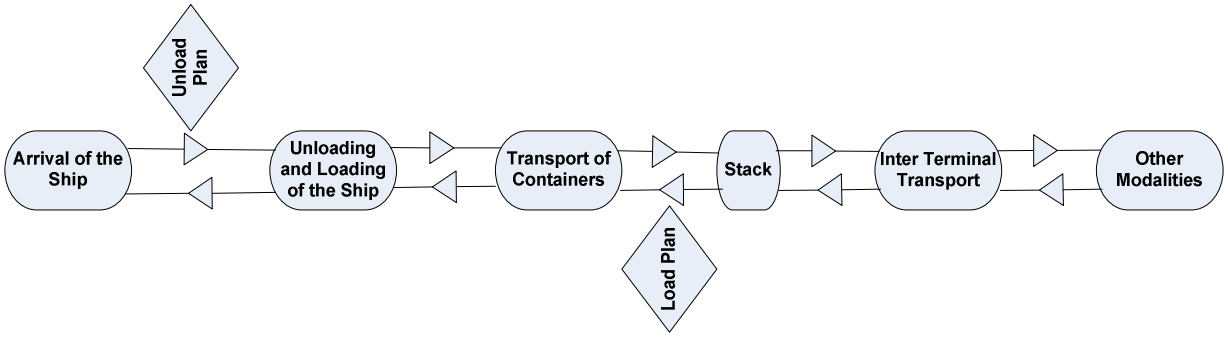


Figure 1.5 Sequence of operations in a container terminal [35]

Similar operations for export and transit containers are applied. The diagram in Figure 1.5 is illustrates a summary of the main operations in a container terminal. Every process is highly dependent on the previous one. Except for arrival and departure of external vehicles all the operations are under the control of port's personnel. A large number of highly dependent operations needs continuous coordination and high efficiency. As a result Operations Research techniques become very handy in planning and control.

1.3 General Approach

Modeling a terminal, including all different processes and problems, is far behind today's technical capabilities. Therefore, decisions on terminal operations are differentiated based on their levels and consequences. These decisions can be grouped in two: strategic

decisions like location and equipment selection and operational decisions such as berth allocation for ships, equipment scheduling, space allocation for containers or even finding exact locations for containers to be stored. Strategic decisions are made very infrequently, whereas operational decisions can be made monthly or as frequently as hourly.

Zhang et al in his work [45], propose a hierarchical model for the operational decisions as depicted in Figure 1.6. This work deals with the location assignment problems which are very low level and need to be applied continuously.

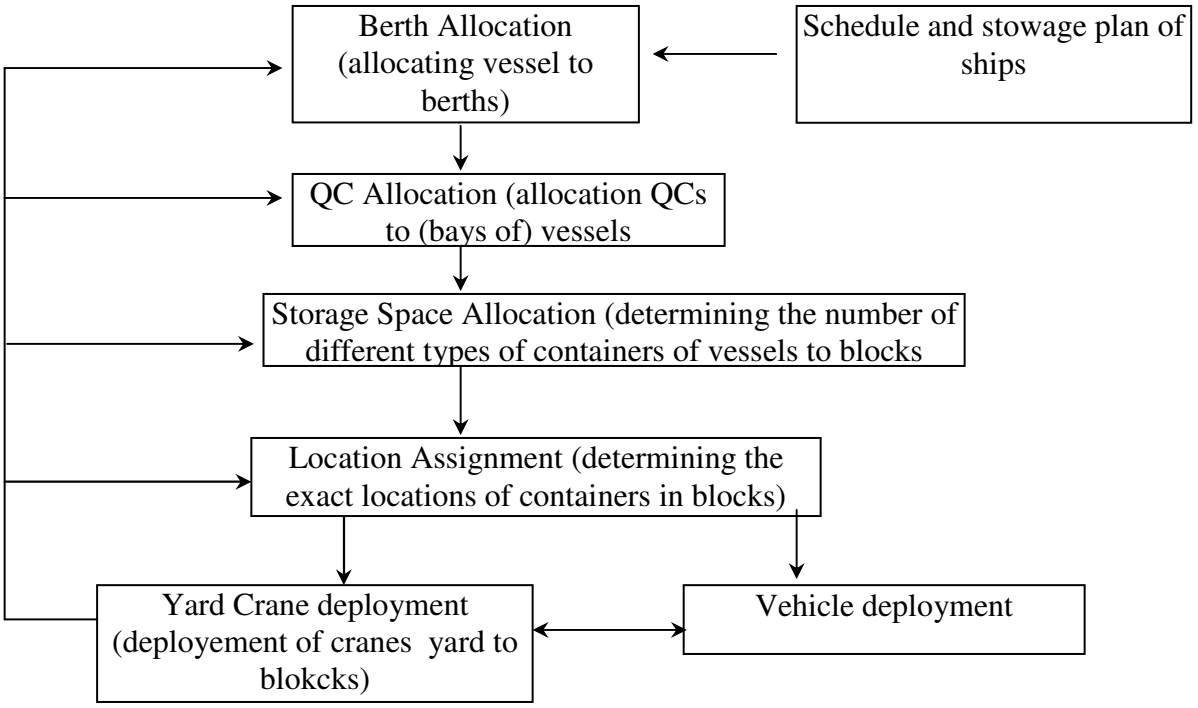


Figure 1.6 Decision Hierarchy in operations

The problem we address is the optimization of the retrieval process of the containers from a bay. In particular, we develop algorithms to determine near optimal moves for a crane to retrieve a set of containers in a sequence to minimize the total number of rehandles and distance traveled. Our contributions can be listed as:

- We find exact branch and bound solutions,
- In addition to number of rehandles, we introduce the total distance traveled by the crane as a criterion,
- We introduce the notion of cleaning moves and present an algorithm based on cleaning moves.

The Remainder of this thesis is organized as follows. A literature review, especially on stacking logistics, is presented in Chapter 2. Chapter 3 provides a reformulation of the container retrieving problem. In this chapter, the problem is solved to optimality and alternative heuristics are developed. Chapter 4 discusses the concept and application of cleaning moves. The thesis concludes with final remarks and future research directions.

2 RELATED WORK

In this chapter, a review of published works will be presented. Optimization in the container terminals is very popular both for industrial researchers and academicians. A huge number of studies is conducted and plenty more is going on. We will be mainly focusing on works based on operations research techniques.

There are two well known and accepted overviews on the subject ([35], [39]). Steenken et al. [35] present a detailed review including; the history of containers, terminal structure, handling equipment types and optimization methods for terminal logistics covering all the processes. Vis and Koster [39] is focused more on process optimization in the terminals. The main difference between the two studies is the way they classify the literature. Steenken et al. [35] classified the literature as follows:

- Overviews
- Container Terminal Systems
- Terminal Logistics and Optimization Methods
    - Ship Planning
        - Berth Allocation
        - Stowage Planning
        - Crane Split
    - Storage and Stacking Logistics
    - Transport Optimization
        - Quayside Transport
        - Landside Transport
        - Crane Transport Optimization
    - Simulation Systems.

On the other hand Vis and Koster's [39] classification is as follows:

- Arrival of the Ship
- Unloading and Loading of the Ship
- Transportation of Containers Between Ships and Stacks
- Stacking of Containers
- Inter Terminal Transport and Other Modes of Transportation
- Complete Container Terminals
- Remaining Literature

While the second classification is based on the sequence of operations in the container terminals, the first one is based on optimization hierarchy. We will be using first approach because it is more compatible to operations research subjects. In the remainder of this chapter, storage and stacking logistics are discussed in detail but the literature related to other subjects is reviewed briefly.

2.1 Container Terminal Systems

Some researchers work on terminal systems as a whole. These studies are mostly for strategic decisions or descriptive simulations. Choi et al. [6] suggest integration of ERP to the container terminal systems. By doing so, the integration of the facilities followed by the increase in efficiency is achieved, especially for terminals operated by more then one company. Hartmann [13] proposes a scenario generator, which is to be used for generating proper data for testing optimization models. Architectural design for a software that will control and dispatch jobs in an automated terminal is given by Kim et al. [26]. Bielli et al. [3] give a simulation model of a terminal and its components. The aim of this study is to provide strategies for increased port efficiency. Murty et al. [30] work on development of a decision support system to deal with the daily decisions at terminals. Mathematical models and algorithms are presented.

2.2 Ship Planning

In terminals, berths are locations where sea vessels are unloaded and loaded. Figure 1.2 shows a ship being unloaded at a berth. The number of ships being served simultaneously in a berth depends on the length of the berth as well as the ship sizes. Berth allocation concerns determining berthing times and the locations of container ships, which is most likely to be decided on a daily basis. Kim and Mon [25] use simulated annealing technique to schedule container ships. Imai et al. [17] try to maximize berth utilization with a mixed integer programming model and define a related heuristic. Imai et al. [18] deals with the problem by assigning service priorities to container ships. Priorities are based on the service times of the vessels. Imai et al. [20] propose a continuous berth allocation problem rather than discrete assignments as in [17], [18]. The latter method [20] tries to achieve higher port efficiency than the previous ones.

Stowage planning is assigning positions in the ship hangars to containers. This plan is first done by the shipping line, according to route and stability of the ship. The objective of the lines is to maximize ship utilization and/or to minimize the container shifting within the ship. Before the loading operation, the ship line delivers its own plan to terminal operators, who plan their operations accordingly. Terminal operators' objective is to maximize quay cranes efficiency and/or to minimize relocation movements at the yard considering ships stability. The plan of the ships, allocates spaces at the vessel on container groups and the plan of the terminal determines exact locations for each container within the group. Wilson and Roach [42] deal with the problem using a tabu search meta-heuristic, which is applied on the objective function to reduce searching time. Ambrosino et al. [1] define master bay planning problem and model a binary linear program. A heuristic as well as some pre-stowage rules are derived. Imai et al. [19] formulate the problem as a multi-objective integer programming model and use weighting procedure to obtain non-inferior solutions. Chung and Vairaktarakis [7] provides an optimal algorithm and develop a heuristic for loading and unloading operations for a single quay crane.

The assignment of quay cranes to berthed ships is called as crane split. Daganzo [10] formulates a mixed integer program for static split of cranes to already berthed ships with no incoming of ships. Peterkofsky and Daganzo [33] present a branch and bound method to minimize delay cost of the ships. Gamberdella et al. [12] split the problem into two sub-problems and solve them hierarchically. Park and Kim [32] propose a two phase solution hierarchical approach. In the first phase a near optimal solution is found by sub-gradient optimization, which is then used to schedule cranes in detail. Kim and Park [27] present a mixed integer programming model, solve it with branch and bound method and then propose a heuristic called 'Greedy Randomized Adaptive Search Procedure'.

2.3 Transport Optimization

Transport optimization for the container terminals is a very wide subject and there are huge amount of studies conducted. Details and classification of the subjects can be found in [35] and [39]. Here we just present the main problem types found in the literature.

Managing vehicles that carry containers between ships and storage area is referred to as quay side transportation. The number of vehicles, their sequencing, scheduling and control are the main aspects of the problem. Mostly automated guided vehicles are assumed.

In terminals which are connected to railroads or terminals having extra depots, assignment and scheduling of resources is called as the land side transportation.

Another problem is crane transportation. In the storage area there are more stacks than the number of yard cranes. This situation causes cranes to be moved between blocks, which is a very time consuming operation. So, scheduling the cranes for the blocks with minimum number of shifts is important in crane transportation problem.

2.4 Simulation Systems

In the last decade, simulation systems became popular because of the significant increase in the computational powers of computers. Studies on terminal simulations became more frequent and they are generally used for analysis of decisions or parameters. Sgouris et al. [34] use simulation to evaluate handling of import containers in a medium-sized terminal. Their aim is to use simulation as a tool for short term planning and process improvement. Howard et al. [15] describe a commercial discrete event simulation model, 'Portsim-5'. Another work belongs to Yang et al. [43], where the authors evaluate the transportation performance within the automated container terminals. Their model suggests the usage of automated lifting vehicles instead of classical automated guided vehicles.

2.5 Storage and Stacking Logistics

When containers arrive into terminals, they are stored until ships or vehicles come and claim them. For example, a container brought by a ship is stored in the terminal until, either it is claimed by a truck or loaded to another ship. Storage time depends on the type of the container, import, export or transit, and can roughly take from a few days to few a weeks. During this time, containers stored in the yard area in stacks are called as blocks (Figure 3.1 shows stacking of containers). During the storing operation problems such as; determining necessary space, assigning containers to specific locations, scheduling containers to be retrieved and rehandling/relocating containers may arise.

Chung et al. [8] model a simulation system and analyze the effect of buffer area for containers to be retrieved. If sufficient number of yard trucks exists, then rehandling for buffer and sweeping operations can be applied. In both cases having buffering space reduces the cranes unproductive movements and increases efficiency significantly.

Watanabi [40] introduced the notion of selectivity index (SI). In his approach each container in the bay is given a value inversely proportional to the number of containers placed above it and the average value gives the SI of the bay. SI is ranged between 0

(exclusive) and 1 (inclusive). As SI increases selectivity, reaching containers, becomes easier. SI's for several block configurations are analyzed and for denser stacking and higher selectivity the use of larger yard cranes is suggested to reduce the density of containers. Using straddle carriers rather than yard cranes is found to be more efficient in terms of both selectivity and ground occupancy. For ground occupancy yard cranes could be preferred for high stacks, especially higher than four levels.

Ashar [2] opposes Watanabi's SI idea, in his work. According to [2] such an index should asses two factors: storage density and handling convenience as storage effectiveness. First instead of the term selectivity, accessibility, which represents the nature of the problem better is suggested. Secondly, instead of SI, which only gives handling convenience, the accessibility index (AI) based on the average number of shuffles per container is suggested. AI captures the trade off between storage density and the number of unproductive moves. For an example; increasing the stack height from 4 to 5 increases the storage capacity 25 percent but it also increases the unproductive moves by %33 according to AI. Ashar [2] concludes with suggesting detailed operation simulations for evaluating different yard systems, rather than SI and AI only.

In Castilho and Daganza [4] two basic strategies for storing import containers are discussed. First one is similar to [2] and is based on expected number of moves per container. For this strategy, best case, equal retrieval probabilities per container, versus worst case, lower containers having higher retrieval probabilities, is analyzed. An average of %33 difference between the best (ideal) case and the worst case is found. The expected number of shuffles is minimized when all stacks are balanced and problem is referred to as "length-biased sampling, expected delay for passengers waiting a bus" in [4]. Second strategy is based on segregating containers according to arrival times. For this, clearing moves are needed when space is not available for newly arrived containers. Then old containers are relocated on top of containers arrived at different times. In this strategy the expected clearing moves are calculated. Strategies are compared and the second strategy is found to be insensitive to height of the stacks but very sensitive to the number of ship

arrivals. The opposite is true for former strategy. Non-segregating strategy should be used in shorter stacks and segregating strategy should be used for higher stacking.

Taleb-Ibrahimi et al. [16] discuss space allocation and storage strategies for export containers. Given the vessel arrival patterns and workloads they find space requirements. Strategies for reducing wasted space are given as using buffer space and/or remarshalling. Dynamic strategies to minimize necessary space or minimizing re-handling are also given. The results highly depend on the ships' arrival patterns.

The analytic evaluation of the expected number of rehandles in the container yards is conducted by Kim [22]. Throughput rate is estimated by the number of rehandles. Assuming random container retrievals and forbidding remarshalling above old containers, their problem is to find the number of rehandles that would occur while emptying a bay of containers. For various tier heights and bay widths, the expected number of container relocations is calculated assuming equal retrieval probabilities for each container. Considering different bay lengths and widths, various alternative combinations are solved using dynamic programming. Based on these expectations, regression analysis is performed and an approximation algorithm for calculating the expected number of rehandles is given. The accuracy of the approximation formula is compared with the selectivity index suggested in [40] and found to be better.

Remarshalling for export containers is discussed in [23]. This operation is defined in [22] as clearing moves. This work of Kim and Bae in a sense, fills the gap in [22] for executing clearing moves. The initial and ideal block layouts are given for the problem and transformation is discussed. The solution is obtained by dividing the problem in to three sub- problems. First is the bay matching problem that is solved by dynamic programming. Second is the move planning problem, which is transformed into the classic transportation problem. Last one is the task sequencing problem for which the traveling salesperson solution procedures are applied. The problem is defined clearly but no computations are done. The suggested solution procedures are computationally intractable and heuristics should be developed.

Chen [5] work on factors causing unproductive moves during storage. The problem is discussed from operational to strategic levels. No model is presented but higher land utilization is discussed a from top to bottom perspective. [5] defines the problem elegantly and provides several perspectives with alternative views.

Kim et al. [24] derive a methodology to locate export containers within a bay. An optimization model based on containers' weight groups is formulated to find the exact location within the bay, minimizing the expected number of rehandles for an arriving export container. Dynamic programming is used to solve this model. The stages are defined by the number of empty slots while the states are given as weight group and empty slots in each row. For practical reasons dynamic programming can only solve small instances, so a complementary decision tree of the problem is formed. The decision tree is pruned based on the classification procedures defined and a fast working heuristic is given. The results of the heuristic are compared with dynamic programming and found to be acceptable.

Space allocation for containers arriving by ships is also modeled by Zhang et al. [45]. A rolling horizon approach is used and the problem is decomposed into two sub-problems. In the first part, the total number of containers to be stored in blocks is decided. This problem is formulated as a max-min problem and transformed to linear integer programming with an objective function for balancing working times of yard cranes. The second sub-problem is assigning the number of containers from different vessels to blocks for which a transportation problem is generated. The output of the first sub-problem is used as demand nodes and vessels become supply nodes. The associated costs are distances between ships and blocks. In the objective function, total distance traveled between ships and blocks is minimized. A numerical study is conducted on data generated according to specifications of Hong Kong container terminal.

Ünlüyurt and Özdemir [38] deal with the space allocation problem as [45]. The problem is decomposed into two networks, one for space allocation and one for location

matching for containers. The model is tested for alternative types of layouts. Efficient solutions, compared to solving location matching problem, are found in polynomial time.

Container positioning problem is defined by Tranberg [36]. The problem is minimizing total handling time of a block in an automated terminal. A linear mixed-integer model with non-polynomial number of variables is formulated. The model consists of container flow and time restrictions. The real backbone of the model is the application of LIFO (last in first out), principle as a set of additional constraints. The model is intractable for applications in real life problems, so heuristics should be developed.

A recent work by Kim et al. [28], which is the starting point of our research, deals with the problem of the retrieval of import containers from a bay. The problem is to find exact locations of relocated containers while retrieving all the containers from a bay according to a predetermined order. The model is formulated and first solved via a branch and bound search with the objective of minimizing number of relocations. Then a heuristic based on expected additional rehandles is proposed. The heuristic runs fast and gives results with about %10 percent optimality gap.

3 METHODS TO RETRIEVE CONTAINERS WITHIN A BAY

Blocks are main storage units for containers in the container yards. Each block consists of a number of bays and each bay consists of a number of stacks/rows. Except for temporary storage all the containers are stacked in blocks. Mainly yard cranes are assigned to blocks for stacking operations. An illustration of how blocks look like and how yard cranes operate is given in Figure 3.1. In the figure, the bay consists of 7 rows/stacks of containers, where one is occupied by a truck, named as truck lane. Normally a bay could have between 2 and 10 rows, each row containing 3-7 tiers. There is no limitation on the number of bays but usually up to 20 bays and yard cranes can move between bays on their wheels.

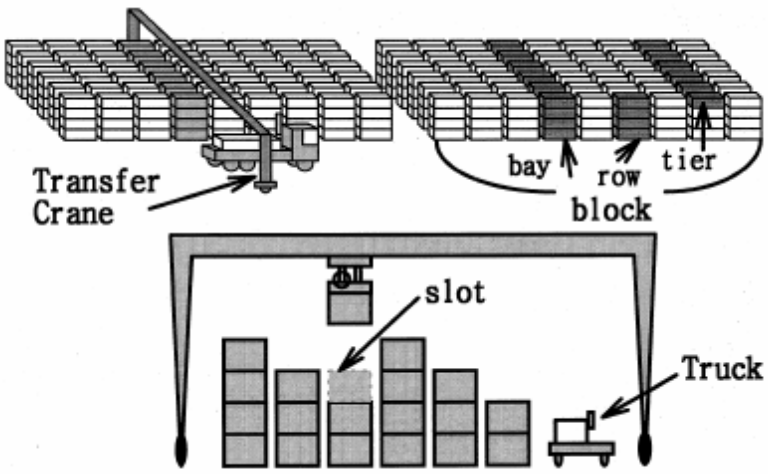


Figure 3.1 Yard Crane and a Container Block [24]

Arrival and retrieval of containers are performed by trucks. When a container arrives on a truck, as in Figure 3.1, the operator moves the crane to the right or left so that it would get on top of the truck. Then, the crane is lowered to pick up the incoming container. Once the container is picked up, the crane is levered up, to move on top of the available position. Then, the crane lowers down to put down the container and is levered up to its usual position. In order for the crane to move horizontally, it has to be levered up.

Retrieval of a container is almost like an arrival. The only difference is that, the container to be retrieved should be accessible by the crane, which means that there should not be any other containers on top of the container to be retrieved. If a container is not accessible, then the containers above it should be rehandled / relocated to other available positions in the bay. Obviously rehandling causes operational inefficiency.

Yard crane operations are low level operations in the terminal and are determined by the crane operators. For example, if a container is being rehandled then crane operator decides where to place it in the bay. In terminals having large throughputs, the yard crane operations may become bottleneck [5]. Thus, there is a potential for improvement and researchers are working on the topic.

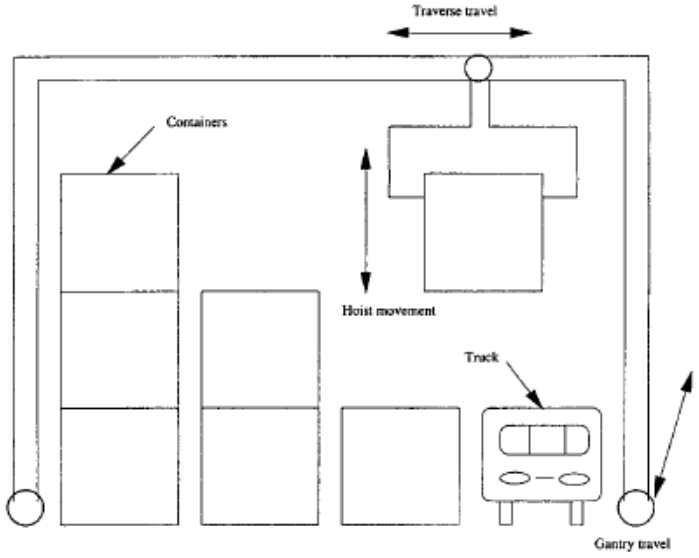


Figure 3.2 Yard Crane Movements [3]

There are four types of movements in yard crane operations (see Figure 3.2). These movements are listed according to their associated costs in time units as follows:

- Bay movement (Gantry travel),
- Handling (Picking up and putting down containers),
- Row movement (Traverse travel) or horizontal movement,
- Tier movement (Hoist movement) or vertical movement.

Because cost of bay movement is too high, when the arrival or retrieval of a container occurs, bays are considered to be independent. Most of the research ([2], [22], [24], [28], [40], assume no intra bay rehandles.

The related decision problem can be described as follows: Given an initial configuration of a bay along with the sequence that the containers will be retrieved, we would like to decide how to relocate containers (when necessary) to minimize an appropriate objective function in terms of the costs defined above. In this chapter, we assume that a container will be relocated only when another container beneath that container is to be retrieved. In other words, we will try to devise a strategy that will retrieve the containers in a bay one by one in a predetermined sequence. When there are other containers on top of the container to be retrieved, the strategy will indicate where to relocate those containers.

Kim et al. [28] have worked on the same problem, illustrated in Figure 3.3. Their objective is to minimize the number of rehandles. The truck lane where containers enter and leave the system is on the left hand side of the first row, <6, 5, 14, 15 >. The numbers on the containers indicate the order that they will be retrieved.

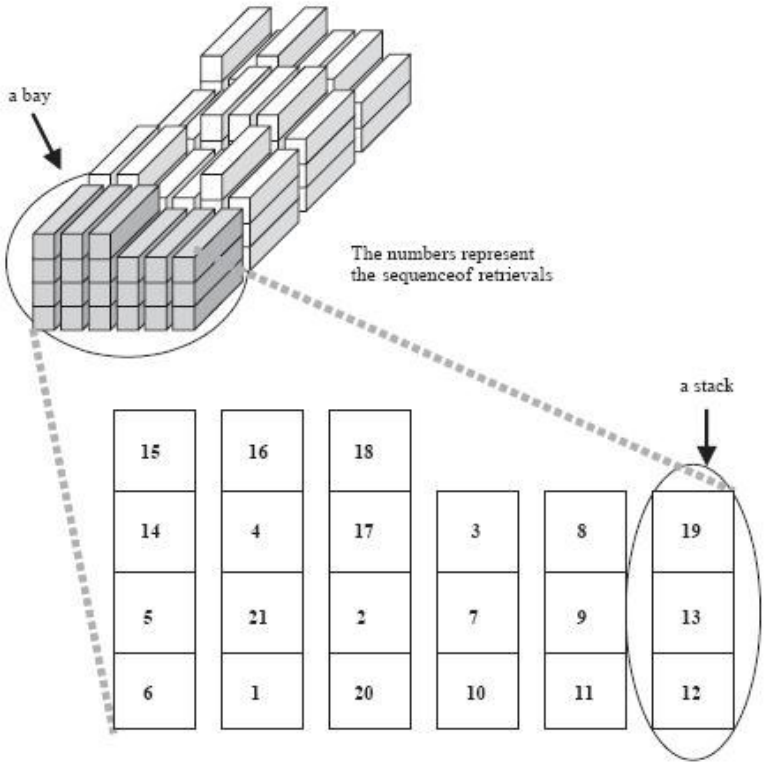


Figure 3.3 Visual Representation of the Problem

Let us define a general cost function for the retrieval of container *k*. Given the following constants, the cost of retrieving container *k* is given in equation (3-1).

A = Handling constant (pick up or put down)
B = Horizontal movement constant
C = Vertical movement constant

$$Cost_k = A * NumberofHandles + B * HorizontalDistance + C * VerticalDistance \quad (3\text{-}1)$$

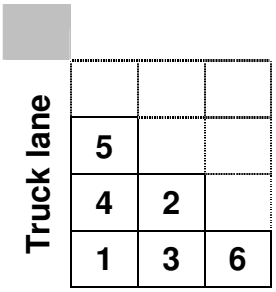


Figure 3.4 Example Configuration and Crane Is Represented as Grey Box

For the example configuration given in Figure 3.4 assuming that, A=5, B=2 and C=1, the cost of retrieving container 5 from the initial configuration is one handle, two horizontal movements and four vertical movements which is equal to 13 (5*(1) + 2*(2*1) + 1*(2*2)). On the other hand the cost of retrieving container 1 depends on where the containers 4 and 5 are relocated. Relocation of containers 4 and 5 also affect future number of relocations when retrieving container 2.

It can be easily seen that problem is dynamic in nature and in fact NP-Hard [24]. In the literature, (see for instance [24], [28]), constants B and C are taken as zero to simplify the problem. This makes the associated cost function for container *k* as in equation (3-2).

$$Cost_k = A * NumberofHandles \tag{3-2}$$

Then objective becomes as in equation (3-3).

$$\min \sum_k \left(NumberofHandles_k \right) \tag{3-3}$$

It can easily be shown that objective in equation (3-3) is same as objective in equation (3-4). The number of handles is sum of the number of rehandles and the number of retrievals and in fact number of retrievals is fixed for a problem.

$$\min \sum_k \left(Numberof\ \mathrm{Re}\, Handles_k \right) \tag{3-4}$$

Minimizing total number of re-handles is accepted in the literature because it simplifies the problem a lot and also A>>B>>C. From now on, for convenience we will refer this objective as Case 1.

In this part of our work, we will drop the assumption of the constant B being equal to zero. Then the cost function of retrieving container k will be as equation (3-5) and the objective function will be as equation (3-6). We will refer to this objective as Case 2.

$$Cost_k = A * NumberofHandles_k + B * HorizontalDistance_k \tag{3-5}$$

$$\min \sum_k \left(A * NumberofHandles_k + B * HorizontalDistance_k \right) \tag{3-6}$$

3.1 Branch and Bound Search

We first propose a branch and bound procedure to solve the problem exactly. Our main goal is to compare our proposed heuristic's results with the optimal solutions when possible. We will basically use the following notation introduced in [2] and define new terms as need arises.

N	:The number of containers in the initial bay.
r	:The number of stacks in the bay.
k	:Container to be retrieved
i	:Stack number from 1 to r
S^k	:The state of the bay after k containers are picked up from the initial bay.
a^k	:Action taken at the removal of k^{th} container.
$h\left(a^k \middle\| S^{k-1}\right)$	:The number of relocations experienced during action a^k on the bay of state S^{k-1} .
$F(S^k)$	:The minimum total number of relocations to pick up remaining containers from the bay at state S^k.

Problem with Case 1 is formulated in equation (3-7) [2]:

$$F(S^0) = \min_{a^1, a^2 \ldots, a^k} \left\{ \sum_{c=1}^{k} h(a^c \mid S^{c-1}) + F(S^k) \right\} \quad \text{(3-7)}$$

where $S^{c-1} \xrightarrow{a^c} S^c$ for $c=1,2,\ldots,k$

For Case 2, we introduce the following notation:

$handle\left(a^k \mid S^{k-1}\right)$	:The number of crane pick ups experienced during action a^k on the bay of state S^{k-1} .
$horizontal\left(a^k \mid S^{k-1}\right)$	:Horizontal distance travelled by crane during action a^k on the bay of state S^{k-1} .
$V(s^k)$	:The minimum value to pick up remaining containers from the bay at state S^k.
A	:Handling coefficient (handling time)
B	:Horizontal movement coefficient (Horizontal movement time)

We formulated Case 2 as in equation (3-8):

$$V(S^0) = \min_{a^1 \ldots a^k} \left\{ \sum_{c=1}^{k} A * handle(a^c \mid S^{c-1}) + B * horizontal(a^c \mid S^{c-1}) + F(S^k) \right\} \quad \text{(3-8)}$$

where $S^{c-1} \xrightarrow{a^c} S^c$ for $c=1,2,\ldots,k$

For solving Case 1 and Case 2 optimally, a branch and bound search, with depth-first and backtracking strategies in [28] is coded in C++ using Microsoft Visual Studio 6.0. Alternative retrieval of containers are enumerated. Using consistent enumeration for branching forced us to use depth-first search rather than other possible tree searching algorithms, which may have been less time consuming. Alternative is to generate all possible bay configurations for each possible retrieval scenario; but would limit our problem size due to computer memory restriction.

3.1.1 Branching Process

In the implementation, our code differs from [28] when branching is executed. In [28], each container movement is used as a branching strategy (each action a^c may be expressed in several consecutive branches). On the other hand, we used each retrieval as a branching strategy (each action, a^c , is a branch). So with the initial configuration, [<1, 4, 5><3, 2><6>], in Figure 3.4. Kim et al. [28] branch the root into two children with configurations [<1, 4><3, 2, 5><6>] and [<1, 4><3, 2><6, 5>]. We branched it into four children with configurations [<1><3, 2, 5, 4><6>], [<1><3, 2, 5><6, 4>], [<1><3, 2, 4><6, 5>] and [<1><3, 2><6, 5, 4>]. This makes our search tree smaller in number of nodes and much shallower, thus, leading our program to search faster. The following statement would explain why our algorithm is more efficient: Searching a not sorted tree is proportional to internal path length of the tree which is $O\left(N \log N\right)$ where N is number of nodes and log N is depth of the tree. Proof and details of the statement is out of the scope of this work but can be found in [41].

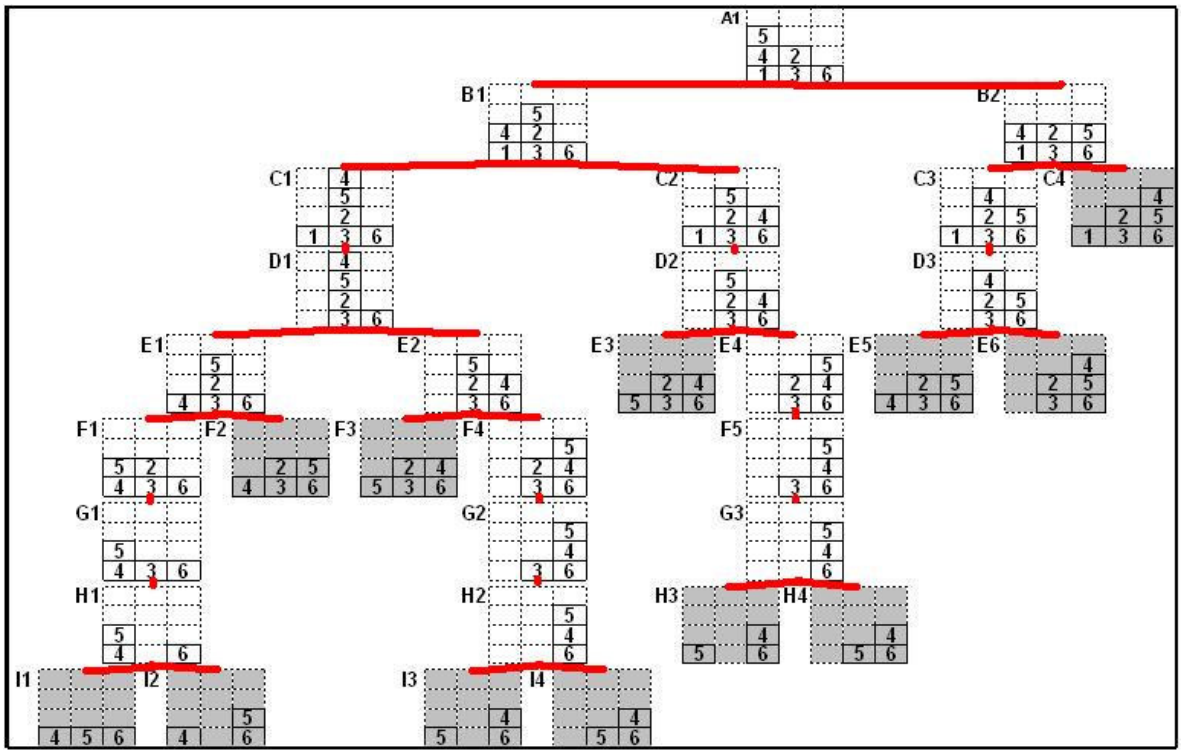


Figure 3.5 Container Movement as Branching Factor

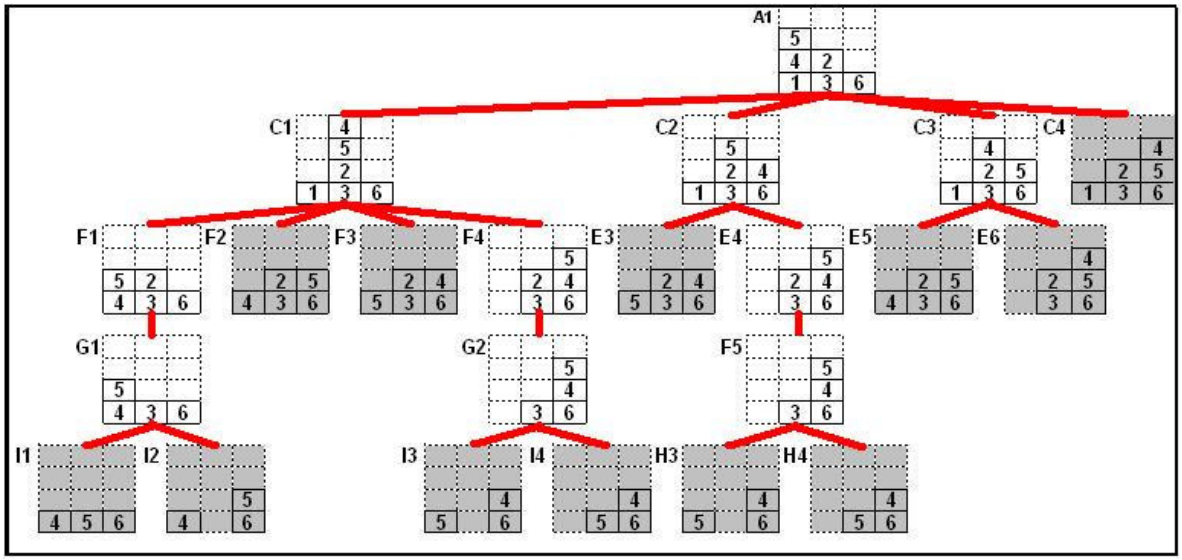


Figure 3.6 Container Retrieval as Branching Factor

Figure 3.5 and Figure 3.6 illustrate the two trees with different branching strategies. The number of nodes and the depth of the first tree are 32 and 9, respectively. The second one only has a depth of 5 with a total 22 nodes.

3.1.2 Bounding Process

For the bounding process we need the following definitions. A realized action is an action which has already occurred and a recognized action is an action which is confirmed to take place. So in Case 1, the following terms are used and their numeric examples are given w.r.t. Figure 3.5.

- Realized rehandle: number of rehandles that is actually acknowledged. For node D1 there are 2 realized rehandles which are rehandling of containers 4 and 5.
- Recognized rehandle: number of rehandles that has to take place. For node D1, container 4 and container 5 should be relocated so recognized rehandles is 2.
- Bound value for node: minimum number of rehandles to retrieve all the containers, which is the number of realized rehandles plus the number of recognized rehandles. For node D1 it is 2+2=4.

Figure 3.7 illustrates rehandlings during a retrieval operation. Initially there are two recognized rehandles, this means that bound on the number of minimum rehandles to retrieve all of the containers is two. When container 5 is placed on top of container 6, realized relocations increase by one and container 4 remains to be a recognized rehandle, bound on the minimum is still two. When container 4 is relocated on top of container 2 realized number of rehandles increase by one and become two. Container 4 is again recognized to be rehandled and thus makes the bound on the minimum as 2+1=3.

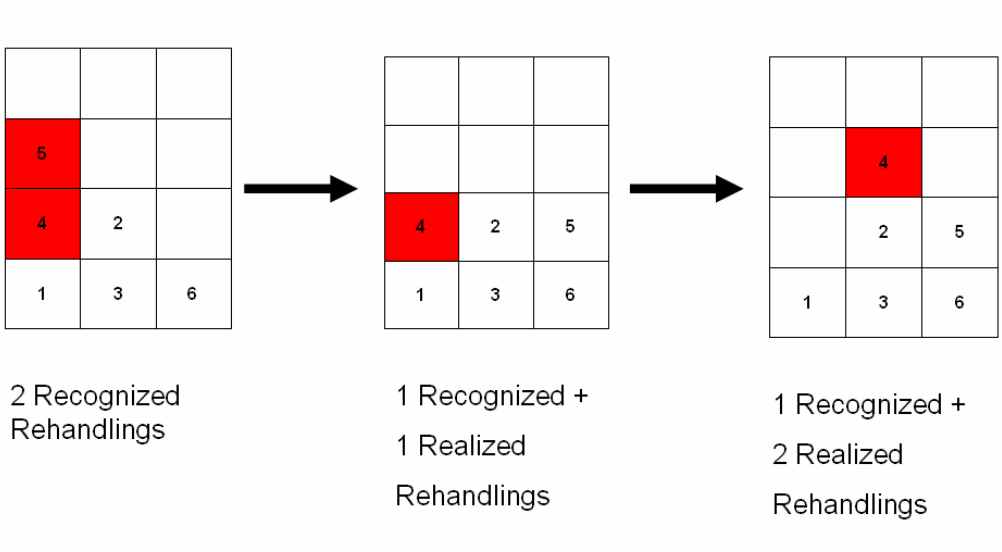


Figure 3.7 Rehandling In a Retrieval Operation

Bounding criteria for Case 2 is more complicated and needs additional definitions. Numerical examples are given considering node D1 in Figure 3.5.

- Realized handle: number of pick-ups and put downs executed by the crane, (1 retrieval + 2 rehandles) = 3.
- Realized horizontal distance: horizontal distance already traveled by the crane, 2*(1 for container 1+1 for container 4+ 1 for container 5) = 6.
- Recognized handle: number of recognized rehandles plus remaining number of containers, (2 for rehandles+5 for retrievals) = 7.
- Recognized horizontal distance: total minimum distance that should be traveled to retrieve remaining containers, 2*(4 for number of containers in stack 2* 2 for horizontal distance from stack 2 to truck lane+1 for number of containers in stack 3* 3 for horizontal distance from stack 3 to truck lane) = 2*(4*2+1*3) = 22.
- Bound value for node: given as in equation (3-9) minimum possible value for a node, A*(3+7) + B*(6+22) = 10A + 28B.

$$Cost_k = A*(numberofRealizedHandles + numberofRecognizedHandles) + B*(realizedHorizontalDistance + recognizedHorizontalDistance) \quad (3\text{-}9)$$

Bounding process in Case 2 is very loose compared to Case 1 because recognized horizontal distance is not affected by the number of rehandles. This situation can be viewed in node E1 in Figure 3.5, second stack is not ordered and container 5 is recognized to be rehandled, thus total recognized horizontal distance is 2*3 = 6 for the second stack. If second stack was ordered, there was no recognized rehandles, then total recognized horizontal distance would be again be 2*3 = 6.

3.1.3 Data Generation

Data used in our experiments are generated with C++ compiled at Microsoft Visual Studio 6.0 using built in random classes. A total of 8000 problems are generated for different initial layouts, different width and height of bays with different load percents. Properties of differences can be given as:

- 2 layouts: balanced; meaning equal initial stack heights and unbalanced; meaning random initial stack heights,
- 5 bay widths: 3 to 7 stacks,
- 4 bay heights: 4 to 7 which is maximum allowed containers on top of each others,
- 5 loading percentages: %55-%60-%65-%70-%75 of the bay is initially occupied,
- 40 cases for each possible combination.

For each case, the layout is determined first and then containers are distributed uniformly random according to their order of retrieval.

3.1.4 Comparison of Case 1 and Case 2:

We have optimally solved 7286 of the 8000 instances in Case 1 and 6455 of the 8000 instances in Case 2. For Case 2, the parameters are assumed as $A = 5, B = 1$. The

number of problems solved by the branch and bound algorithm and average computation times for each bay configuration are given in Appendices A, B and C.

The difference between our branching strategy and that of [28] was previously stated. Even though an exact comparison cannot be done because of lack in the details of problem instances in [28], Figure 3.8 shows the difference between computation times. In the figure computation times for randomly generated problems are given. Computer we used is a Celeron (R) with 2.8 GHz processor and 256 MB of RAM. Computer used in [28] is a Pentium III-800 with 128 MB RAM. Both programs our and [28]'s, are constructed by Visual Studio C++ 6.0. It is clear that our algorithm performs much more efficiently. Small and large instances are neglected in the figure but complete list for average computation times and number instances solved is given in the Appendix D.

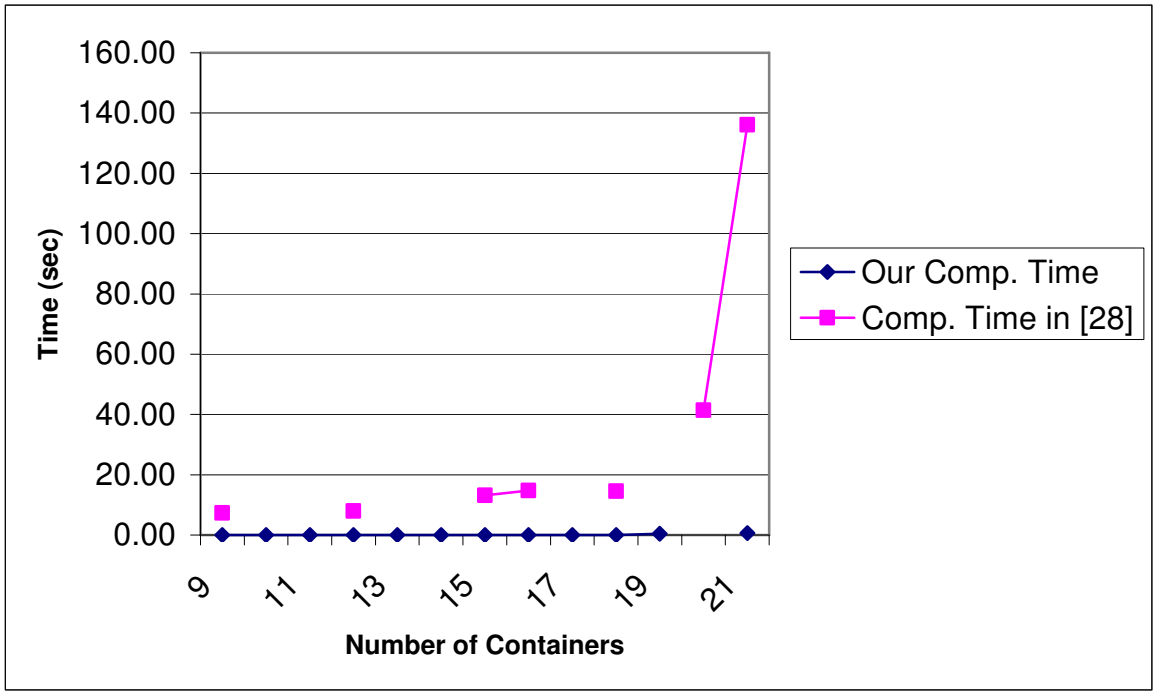
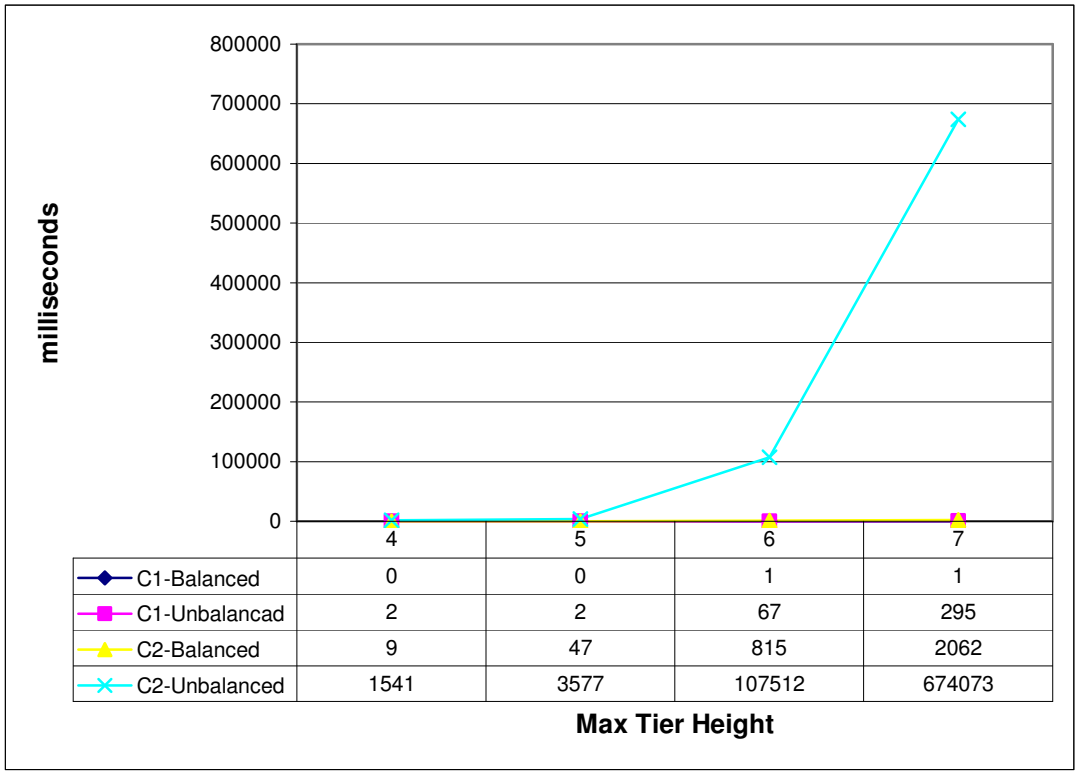


Figure 3.8 Computation Times of Branch and Bound Algorithm

Figure 3.9 captures computational differences between Case 1 and Case 2. The average computation times are for bays either initially balanced or unbalanced, with a width of 6 rows which are initially %60 filled and height is the varying parameter. It is obvious

that average computational time increases from Case 1 to Case 2 and also from balanced to unbalanced.



	4	5	6	7
C1-Balanced	0	0	1	1
C1-Unbalancad	2	2	67	295
C2-Balanced	9	47	815	2062
C2-Unbalanced	1541	3577	107512	674073

Figure 3.9 Average Computational Times for Various Tier Heights

On the other hand Figure 3.10 and Figure 3.11 show how the gap between Case 1 and Case 2 increases for varying number of stacks when maximum tier height is kept fixed. Values of Case 1 are recalculated according to equation (3-5) so as to be compatible with values for Case 2. Data points in the graphics are ratio of the difference of Case 1 and Case 2's objective values and calculated as (Case1 – Case 2)/Case 2.

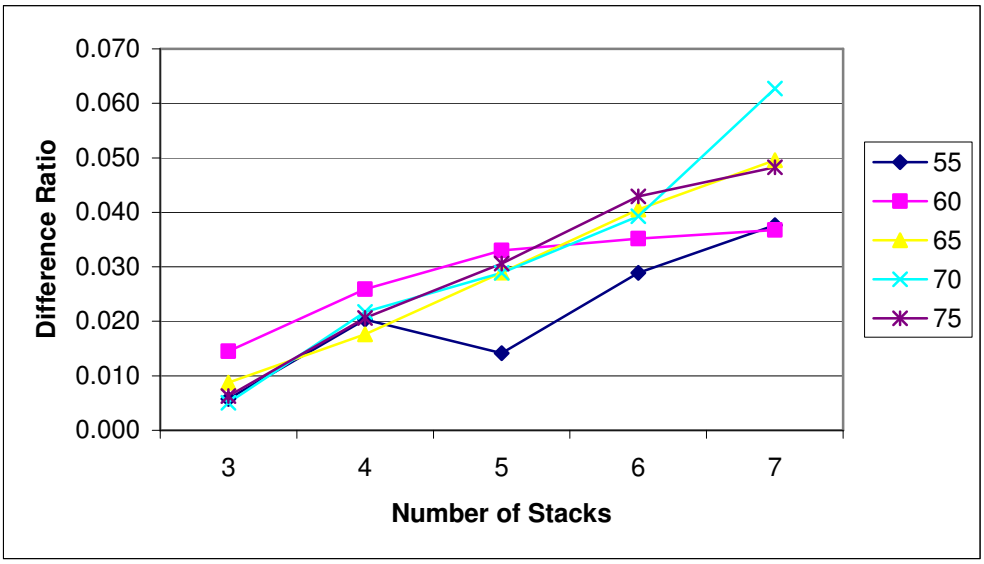


Figure 3.10 Difference in Objective Values of Case 1 and Case 2 for Different Percent of Initial Container Densities in a Balanced Bay.

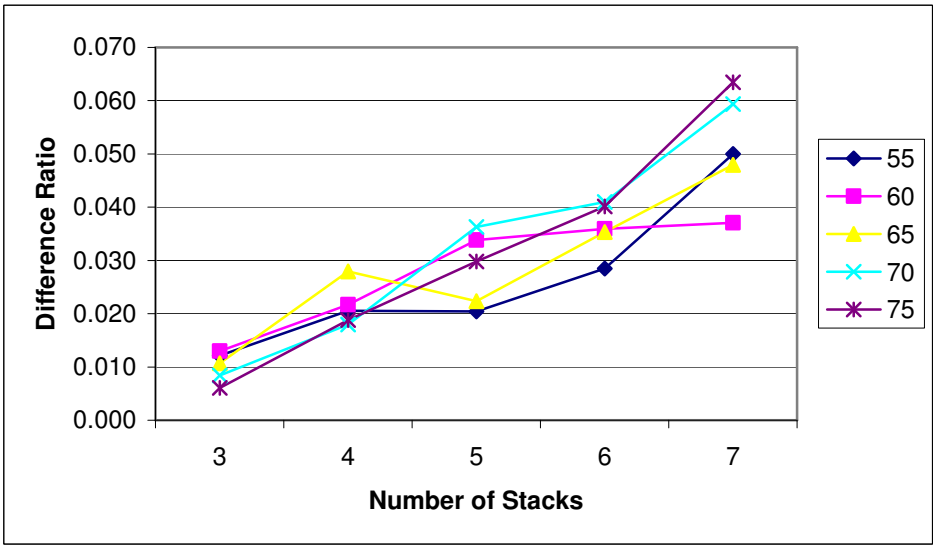


Figure 3.11 Difference in Objective Values of Case 1 and Case 2 for Different Percent of Initial Container Densities in an Unbalanced Bay.

Complete data for differences between values of cases is given in Appendix E. We observed that as the size of the problem increases, the solution gap between the two cases of the same instance of the problem increases. Consequently we were convinced that the

distinction between Case 1 and Case 2 should be made even though solution to the problem becomes harder.

3.2 Heuristic with Expected Additional Relocations

Since solving the problem for larger size instances requires excessive computational effort, we propose heuristic algorithms. We compare the algorithms among themselves and with the optimal solution, when possible, for Case 1 and Case 2.

According to our knowledge the only proposed heuristic for the solution of the problem other than branch and bound search is suggested in [28], where a pessimistic approach is considered assuming random container movements. Basically, the idea is to check each alternative location for container X that is being rehandled and estimate the expected additional relocations that would result by placing container X to the alternative location. Since our first heuristic uses a similar idea we explain this heuristic in detail.

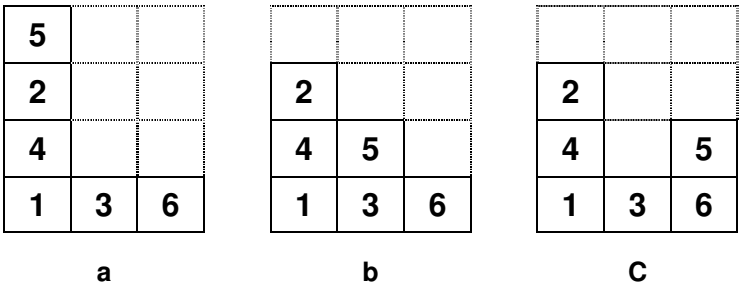


Figure 3.12 Example bay configuration

Let us assume that the initial configuration of a bay is as in Figure 3.12-a. Then, containers 4, 2, 5 have to be rehandled to retrieve container 1. First container to be moved is 5 and there are two options as demonstrated in Figure 3.12-b and Figure 3.12-c.

For the first option assuming Case 1, Figure 3.12-b, cost of moving container 5 is at least two additional rehandles. First rehandle is the movement from Figure 3.12-a to Figure 3.12-b and the second one is the new recognized rehandle, container 5. On the other hand this movement of container 5 on top of 3 may cause more additional rehandles if the two

slots above container 5 are filled with other containers like 6. So there is possibility for other containers to be located on top of container 5, where two empty locations are available. In the best case those two empty locations would remain empty and associated cost with moving container 5 to middle stack would be two rehandles. In the worst case, both of the empty locations could be filled, but assume that only one of them will be filled. If container 2 is placed on top of 5, no additional rehandles would be added because container 2 is earlier than 3, smallest container in the stack. If rather than container 2, container 4 is placed above container 5 then a recognized rehandle should be added to the cost because container 4 is later than 2 and should be relocated to retrieve container 2. In scenario with container 2, movement of container 5 costs 2 rehandles, in the other scenario if container 4 is placed above container 5 cost would be 3 rehandles. Assuming both scenarios are equally likely then expected additional rehandle for moving container 5 to middle stack would be 0.5*2+0.5*3=2.5.

For the second option, Figure 3.12-c, the expected additional rehandle is only 1, which is rehandling of container 5. Container 5 is earlier than stack minimum, container 6, and either moving container 4 or container 2 on top of container 5 would not add any more additional movement. In the first option expected additional re-handles was 2.5 and in the second option it is 1. Between two options choosing second one is better in the short term and this is strategy in [28] for placement of rehandled containers. It is seen that this heuristic greedily chooses between movements of containers with the best short term additional cost. In this strategy we assume that only next movement is known and possible movements after the next movement is assumed to be random.

The following notation is defined for the algorithm:

n = Remaining number of containers in the bay.
r = Maximum number of stacks in the bay.
f_i = Number of occupied slots in stack i.
a_i = Accessible container of stack i.
m_i = Container having minimum order number in the stack i.
e_i = Number of empty slots in stack i.
p_{ji} = Probability of container of order j being located to stack i.
$E(e_i, j)$ = Expected additional relocations that would be caused by relocating future containers to stack with minimum order j and e_i number of empty slots.

$$e_i = \begin{cases} \lceil n/r \rceil - f_i & \text{if } f_i < \lceil n/r \rceil, \\ 1 & \text{if } f_i > \lceil n/r \rceil, \\ 0 & \text{if } f_i = \text{maximum possible height.} \end{cases} \tag{3-10}$$

Let z be the order of next container that will be stacked on top of stack i. Then, finding the expected additional relocations is a recursive function given as:

$$E(e_i, j) = \sum_{z=1}^{j} p_{zi} * E(e_i - 1, z) + \sum_{z=j+1}^{n} p_{zi} * (1 + E(e_i - 1, j)) \tag{3-11}$$

Equation (3-11) is a recursive function. To limit the number of recursions, e_i is limited with 1 if stack i contains more containers than bay's average and $\lceil n/r \rceil - f_i$ if less than the bay's average, as shown equation (3-10). For Case 1, the algorithm is presented in Figure 3.13.

While (There is container to be retrieved)

If (Container to be retrieved is accessible by the crane)

Remove the container from the bay

Else

y= Accessible container on top of container to be retrieved

$stack = \arg\min_i E(e_i, j)$ for next container y.

Relocate container y to *stack*.

Figure 3.13 Expected Additional Relocation Algorithm for Case 1 (EAR1).

3.2.1 Modifying Expected Additional Relocation Idea for Case 2

We modify the recursive function for Case 2 so that it would contain crane's horizontal distance. Lets;

A :Handling coefficient
B :Horizontal movement coefficient
y :Container that is being rehandled
l :Stack origin of y.

Thus recursive function becomes as in equation (3-12) and the modified algorithm EAR2 is given in Figure 3.14.

$$E_{\text{mod}}(e_i, j) = A * \left(\sum_{y=1}^{j} p_{yi} * E(e_i - 1, y) + \sum_{y=j+1}^{n} p_{yi} * \left(E(e_i - 1, j) + 1 \right) \right) + 2 * B * |l - i| \tag{3-12}$$

While (There is container to be retrieved)

If (Container to be retrieved is accessible by the crane)

Remove the container from the bay

Else

y= Accessible container on top of container to be retrieved

$stack = \arg\min_{i} E_{\text{mod}}(e_i, j)$ for next container *y*.

Relocate container *y* to *stack*.

Figure 3.14 Expected Additional Relocation Algorithm for Case 2 (EAR2).

3.3 Greedy Heuristic

The idea of expected additional rehandles relies on the assumption of random container movement. As a matter of fact, this will not typically be true. Therefore we propose another algorithm which does not rely on such assumptions.

In branch and bound search, we use an effective branching strategy and generate well structured trees for the optimal solution. In this algorithm, like greedily minimizing next movement's expected additional moves, we choose a branching action to minimize bounding value of one step ahead configuration. The variables used for the following algorithms are as defined previously in section 3.2.

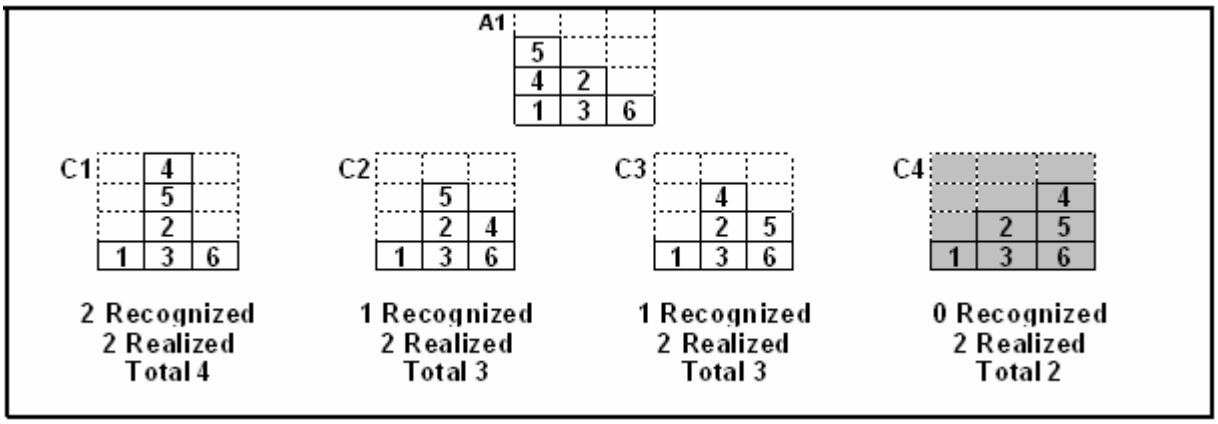


Figure 3.15 A Greedy Algorithm Iteration

Figure 3.15 is an illustration for the algorithm's iteration. In the figure A1 is the initial configuration and C1 to C4 are the possible configurations for retrieving container 1. Among these possibilities C4 has the minimum possible value so that the greedy algorithm for Case 1 iterates to C4 from A1. These iterations continue until all the stacks are ordered. Algorithm for Case 1 is given in Figure 3.16 and the modified algorithm for Case 2 is in Figure 3.17

While (There is container to be retrieved)

At state S^k choose action a^k s.t.

$$\arg\min_{a^k}\left\{F(S^{k+1})\right\} \quad \text{where } S^{i-1} \xrightarrow{a^i} S^i \quad \text{for } i=1,2,...,k \tag{3-13}$$

Figure 3.16 Greedy Algorithm for Case 1 (GA1)

While (There is container to be retrieved)

At state S^k choose action a^k s.t.

$$\arg\min_{a^k}\left\{V(S^{k+1})\right\} \quad \text{where } S^{i-1} \xrightarrow{a^i} S^i \quad \text{for } i=1,2,...,k \tag{3-14}$$

Figure 3.17 Greedy Algorithm for Case 2 (GA2)

3.4 Difference Heuristic

Both of the heuristics given above need many calculations. This may be a burden even for a middle sized problem. Alternatively we propose another algorithm, which is very straightforward and can even be applied manually. This is an algorithm for crane operators' usage. The idea behind the algorithm is same with the expected additional relocation algorithm. When a container X is to be relocated, a stack with container Y is chosen such that:

- A container Y that is the container smallest with the order number in the stack and bigger than that of container X is searched. This way we don't add any cost, just a recognized rehandle becomes a realized rehandle. If multiple stacks satisfying this condition exist then stack containing smallest Y is chosen. By minimizing the difference we minimize the number of containers ordered between X and Y, which will be rehandled again in the case of being relocated on X.
- If container Y satisfying above condition is not found then a container Z that is accessible by the crane and with an order number smaller than X is searched. In this way we stack containers which will be relocated in a reverse order so that they may become ordered when they are relocated. Again the difference between X and Z is minimized due to the same reasoning.
- If either Y or Z cannot be found, we simply minimize the difference between the order numbers of container X and the container which X will be located on.

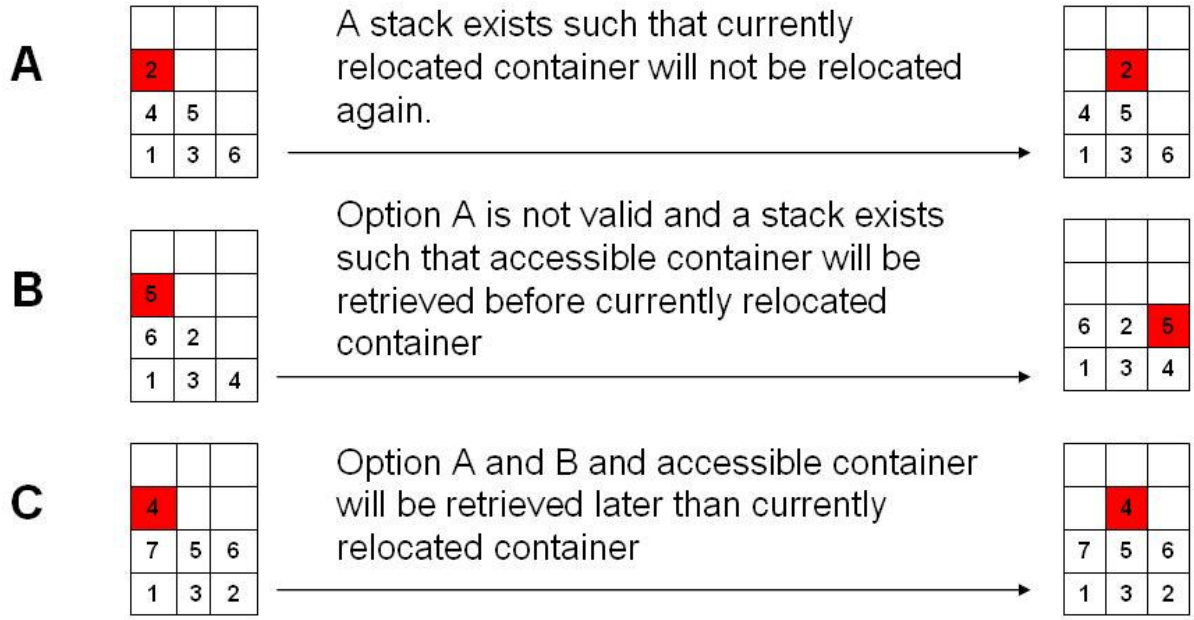


Figure 3.18 Possible Bay Configurations for Difference Heuristic

So in each case we try to minimize the difference in the orders of containers to minimize the number of containers that would potentially be rehandled in the future. While doing this, empty stacks are assumed to contain highest ordered container. Figure 3.18 illustrates how the algorithm iterates at each step. The following notation will be used to define the algorithm and in Figure 3.19 difference algorithm for Case 1 is given.

$i =$ stack of container to be retrieved

$j =$ alternative stack number j for container a_i

$mDist_{ij} = m_j - a_i$

$aDist_{ij} = a_i - a_j$

$m_j = a_j = N + 1$ if stack j is empty

$j < i$ if stack j is closer to truck lane than i

While (There is container to be retrieved)	D1-1
If (Container to be retrieved is accessible by the crane)	D1-2
Remove the container from the bay	D1-3
Else	D1-4
i = stack of container to be retrieved	D1-5
If $\min_{j, j \neq i} \left(mDist_{ij} \middle\| mDist_{ij} > 0 \right)$ exists	D1-6
move a_i to j	D1-7
Else if $\min_{j, j \neq i} \left(aDist_{ij} \middle\| aDist_{ij} > 0 \right)$ exists	D1-8
move a_i to j	D1-9
Else	D1-10
$\min_{j, j \neq i} \left(-aDist_{ij} \middle\| aDist_{ij} < 0 \right)$	D1-11
move a_i to j	D1-12

Figure 3.19 Difference Algorithm for Case 1 (DA1)

Lines 6, 8 and 11 of DA1 should be clarified. When the container to be retrieved is not accessible, the container on top of stack *i*, say container X, should be rehandled. In line 6 we try to find a stack for container X which has a minimum order higher than X, so that it should not be rehandled again. If such a stack could not be found, then the algorithm moves to line 8.

In line 8 stacks with an accessible container Y, which has an order higher than X is searched. The reason is as follows: When X is relocated if Y would also be relocated; then Y could be positioned above X. If that is also not the case the remaining stacks are searched in line 11. In every scenario, the difference in the orders of containers is minimized to minimize the probability of unnecessary rehandlings for the containers ordered within that difference.

While (There is container to be retrieved)	D2-1
If (Container to be retrieved is accessible by the crane)	D2-2
Remove the container from the bay	D2-3
Else	D2-4
i = stack of container to be retrieved	D2-5
If $\min_{j,j<i}\left(mDist_{ij}\,\middle|\,mDist_{ij}>0\right)$ exists	D2-6
move a_i to j	D2-7
else if $\min_{j,j>i}\left(mDist_{ij}\,\middle|\,mDist_{ij}>0\right)$ exists	D2-8
move a_i to j	D2-9
else if $\min_{j,j<i}\left(aDist_{ij}\,\middle|\,aDist_{ij}>0\right)$ exists	D2-10
move a_i to j	D2-11
else if $\min_{j,j<i}\left(-aDist_{ij}\,\middle|\,aDist_{ij}<0\right)$ exist	D2-12
move a_i to j	D2-13
else if $\min_{j,j>i}\left(aDist_{ij}\,\middle|\,aDist_{ij}>0\right)$ exists	D2-14
move a_i to j	D2-15
else	D2-16
$\min_{j,j>i}\left(-aDist_{ij}\,\middle|\,aDist_{ij}<0\right)$	D2-17
move a_i to j	D2-18

Figure 3.20 Difference Algorithm for Case 2 (DA2)

Modification for Case 2 is given in Figure 3.20 and distinction between two types of stacks is done. First kind is the stacks between stack of container to be rehandled and truck lane and it has higher priority than the other kind. If available position is not found in first kind of stacks then the stacks away from the truck lane, are searched. This algorithm for Case 2, is unlike other algorithms for Case 2, and does not consider constants the A and B, but just the priority of handling is kept higher than the horizontal distance traveled.

3.5 Comparison of Heuristics

All of the 8000 instances are tested for;

1. Random container movements,
2. Expected Additional Rehandle Algorithms Case 1 & Case 2,
3. Greedy Algorithms Case 1 & Case 2,
4. Difference Algorithms Case 1 & Case 2.

Head-to-head comparisons of algorithms are presented in Table 3.1, Table 3.2, Table 3.3, Table 3.4. For the tables each number in the cell corresponds to number of instances that heuristic in the row head gives better solution than the heuristic in the column head.

Table 3.1 One to One Comparison of Heuristic Values with the Optimal Values for Case 1

	Opt.	Greedy	Diff	Exp.
Optimal	0	3263	2002	5077
Greedy	0	0	690	3990
Difference	0	2429	0	4613
Expected	0	704	249	0
		Out of	7286	cases

Table 3.2 One to One Comparison of Heuristic Values with the Optimal Values for Case 2

	Opt.	Greedy	Diff	Exp.
Optimal	0	4598	5344	5493
Greedy	0	0	3670	3847
Difference	0	1779	0	3017
Expected	0	1568	2547	0
		Out of	6455	cases

In Table 3.1 and Table 3.2 we compare the proposed heuristics with the optimal solutions. As given in section 3.1.4, for Case 1 out of 8000 total instances 7286 of them are solved through optimality and for Case 2 number of instances optimally solved are 6455. In these two tables heuristics are compared for each instance's optimal value and heuristic value. So branch and bound search gives better solutions than: GA1 for 3263 instances, DA1 for 2002 instances and EAR1 out of 7268 instances. This gap in the number of non optimal solutions increases for Case 2: 4598 for GA2, 5344 for DA2 and 5493 for EAR2.

So from Table 3.1 we can conclude that the difference heuristic is better than the greedy and expected heuristics for Case 1. However for Case 2 the greedy heuristic is better the difference and the expected heuristics.

Table 3.3 One to One Comparison of Heuristic Values with the Random Movement Values for Case 1

One to One Comparison for Case 1				
	Random	Greedy	Diff	Exp.
Random	0	160	55	631
Greedy	7124	0	771	4599
Difference	7307	2951	0	253
Expected	6369	773	5307	0

Table 3.4 One to One Comparison of Heuristic Values with the Random Movement Values for Case 2

One to One Comparison for Case 2				
	Random	Greedy	Diff	Exp.
Random	0	258	448	602
Greedy	7532	0	4525	4926
Difference	7254	2450	0	4003
Expected	7117	2019	3088	0

Table 3.3 and Table 3.4 exclude optimal values and compare heuristics with the solutions gathered by random container movements for each instance. For most of the instances the heuristics do significantly better than random acting but none of the heuristics seems superior to another one.

Performance of heuristics on finding the optimal solutions and optimality gaps of the heuristics are given in Table 3.5. For Case 1 difference heuristic is superior and for %73 of the instances difference algorithm finds the optimal. When Case 2 is considered the greedy heuristic is better, however performance of the heuristics decrease significantly compared to Case 1.

Table 3.5 Overall Comparison of Heuristics with the Optimal

	Case 1		Case 2	
	% Optimal	% Gap	% Optimal	% Gap
Greedy	55	03	29	05
Difference	73	02	17	08
Expected	30	08	15	08

The effect of initial container density in the bay on the average optimality gaps are given in Figure 3.21, Figure 3.22, Figure 3.23, Figure 3.24. Distinction between balanced and unbalanced initial bay configurations is considered in the figures. Gaps are calculated for Case 1 as (heuristic value-optimal value) and for Case 2 as (heuristic value/optimal value)-1. The data points represent the average of differences not the difference of averages. The heuristics are represented by their initials.

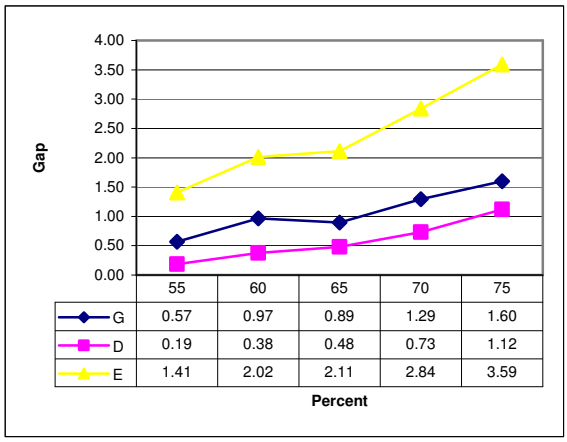


Figure 3.21 Optimality Gap of Heuristics For Case 1 and Initially Balanced Bay

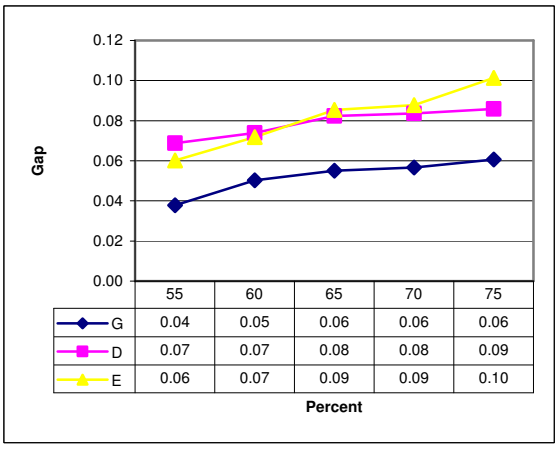


Figure 3.22 Optimality Gap of Heuristics For Case 2 and Initially Balanced Bay

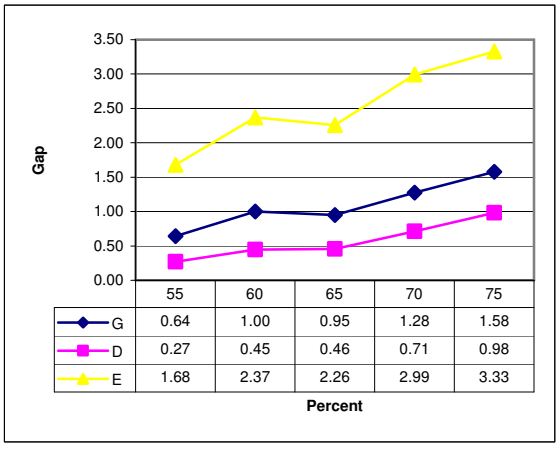


Figure 3.23 Optimality Gap of Heuristics For Case 1 and Initially Unbalanced Bay

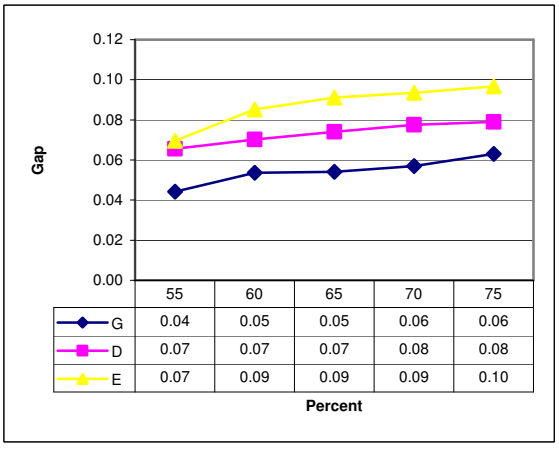


Figure 3.24 Optimality Gap of Heuristics For Case 2 and Initially Unbalanced Bay

We can see that the optimality gap for balanced bays is less than that of unbalanced bays. Expected additional rehandle heuristic has the highest optimality gap in all cases except for the two data points in Figure 3.22. The difference heuristic seems superior for Case 1 and the greedy heuristic for Case 2. For Case 1 distinction within the heuristics is more significant than Case 2. Figures confirm the results given in Table 3.5.

In Table 3.6, heuristic solutions are compared to solutions provided by random relocation movements. The numbers in the tables are average ratios of the difference in the heuristic solutions and random solutions. On average, all of the heuristics give better results. As before the difference heuristic is better in Case 1 while the greedy heuristic is better in Case 2. The gap between random solutions and heuristics is not effected by bays initially being balanced or unbalanced.

Table 3.6 The Heuristic Solutions Compared with Random Container Movement Solutions

	Case 1			Case 2		
	Balanced	Unbalanced	Average	Balanced	Unbalanced	Average
Greedy	13	13	13	14	14	14
Difference	15	15	15	12	13	13
Expected	09	09	09	12	11	12

Computation times of heuristics are omitted here because the majority of the computations finished in less than 0.5 milliseconds on a personal computer Celeron 2.80 GHz with 256 MB RAM.

Following remarks for the algorithms can be done:

- The greedy algorithm and branch and bound search are action based, the expected algorithm and the difference algorithm are movement based, where actions consist of several rehandling movements followed by a retrieval movement.
- Computation time will increase linearly as the number of rows in bays increases but computation times for the greedy heuristic and the expected heuristic increases exponentially as height of the bay increases.

- The performance of expected heuristic is weak and basic reason for this is the result of assuming random container movements.
- Difference heuristic is easy and gives best results for Case 1 but modification for Case 2 is weak and should be developed.
- Greedy heuristic is very robust and gives acceptable solutions for both cases but as the problem gets detailed, the optimality gap of solutions increases.
- All of the suggested heuristics are easy to implement and require very short computation times. So, a combination of these heuristics can give superior heuristics.

4 INTRODUCING CLEANING MOVES

In the previous chapter, relocation movement of a container is executed when the container beneath will be retrieved. This is an assumption and in fact it may increase yard crane's workload by increasing number of relocations. To clarify consider the example in Figure 4.1. According to the assumption in Chapter 3, container 20 will be rehandled when container 5 will be retrieved. Relocating container 20 on 23, before any container ordered smaller than 20, may reduce future number of relocations of container 20. In this chapter, to minimize total handling time of the cranes, we propose a strategy that would allow containers to be rehandled earlier than their turn. These movements will be referred to as Cleaning Moves.

In the literature, the need and possibility of cleaning moves were suggested for different problems [5], [46]. The basic suggestion is the reorganization of a container during the yard crane's idle time. By the reorganization, when the time comes, retrieval process of the containers will be improved. Although the concept was mentioned, exactly when and how cleaning moves shall be performed were not defined.

	20			
	10			
8	11		17	19
9	21		16	24
6	14		12	4
15	5		22	13
3	18	23	7	25
i	ii	iii	iv	v

Figure 4.1 Example Configuration (Recognized Rehandles are Shaded)

We can illustrate how cleaning moves may be useful by the following example. In Figure 4.1, configuration for an unbalanced bay is given and the container that will be retrieved next is container number 3. When the problem is solved with the branch and bound search described in the previous chapter, the next container movement is to move container 8 from stack i to stack iii. This move is the optimal move under the assumptions stated in Chapter 3, but obviously this move will cause extra rehandles later, if containers ordered between 8 and 23 are placed on top of container 8. On the other hand, there are containers which are already recognized to be rehandled and ordered within the range 8 to 23. So, there is a potential gain by placing one of containers 20, 17 and 19 on top of 23 before container 8. Among containers 20, 17 and 19 placing 20 on top of 23 is better because 17 or 19 can be placed on top of 20 without being rehandled again.

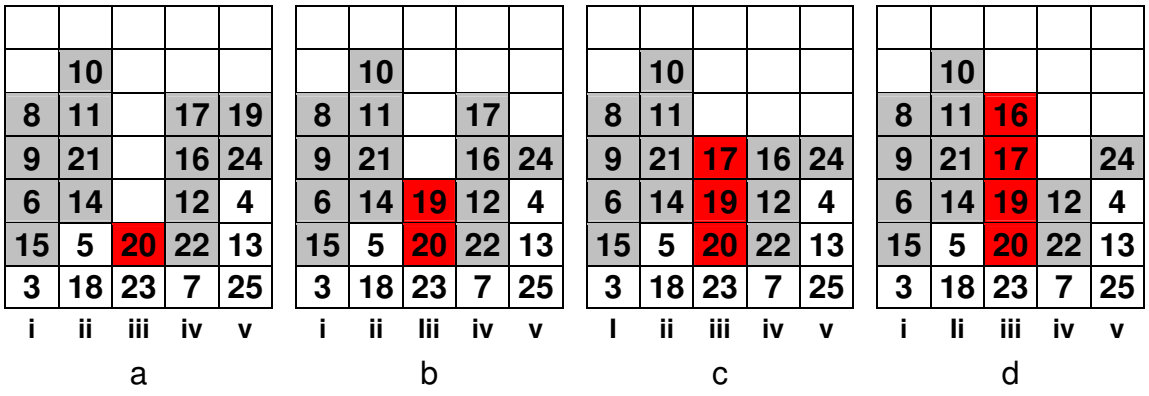


Figure 4.2 Iterative Cleaning Moves

When container 20 is relocated on 23 as a cleaning move, the bound value (defined in Chapter 3.1.2) of the bay assuming Case 1, has not been changed. Before the relocation there were 15 recognized rehandles with in the bay and by moving container 20 on 23, number of recognized rehandles is reduced to 14 and number of realized rehandles increase by 1. Even though current values of bay configurations in Figure 4.1 and Figure 4.2-a are same, when the problems are solved by the branch and bound algorithm the former has an objective value of 46 and the latter has 44. In the latter case moving container 20 on top of 23 should be added to objective, which would make latter case's objective value 45. So, this cleaning move decreases the total working time of the crane by reducing the number of rehandles by 1 (for Case 1). When Figure 4.2-a is analyzed we can see that instead of just

relocating container 20 in stack iii, all or some of containers 19, 17 and 16 can also be relocated to the same stack earlier than their relocation time.

In Figure 4.3, the effect of consecutive cleaning moves (as in Figure 4.2) on the optimal objective function is given for the above example. Figure 4.4 shows the corresponding solution times of the instances. Clearly, except for the last cleaning move, Figure 4.2-d, all of the moves resulted improvement in the objective value.

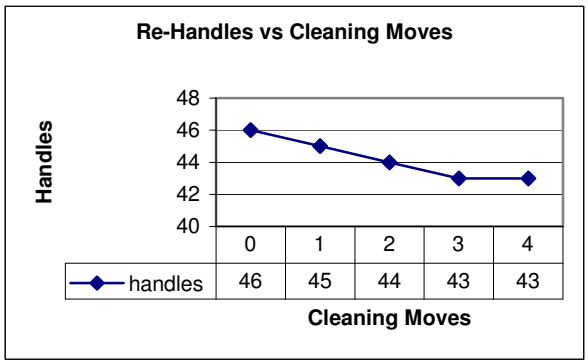


Figure 4.3 Number of Handles for Iterative Cleaning Moves

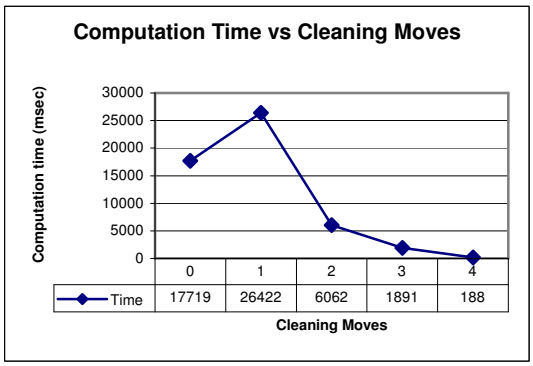


Figure 4.4 Computation Time for Iterative Cleaning Moves

Change in the objective function value is the difference between the number of future relocations of cleaned container (assume 20), if it was not moved and the number of additional relocations of containers 21 and 22 caused by avoiding them to be placed on top of container 23. The new objective function value might be superior as in Figure 4.2-a, Figure 4.2-b and Figure 4.2-c, same as in Figure 4.2-d or be worse. So, if cleaning moves will be executed they should improve the solution compared to the case of not doing the cleaning move. In this context two main points for cleaning moves become important: recognizing a cleaning move and justifying the cleaning move. The latter is referred as deciding if the cleaning will improve the objective value or not. The idea of cleaning move suggested here is a simple method towards finding the real optima of the problem.

4.1 Recognition of Cleaning Moves

We assume that a cleaning move should not increase a bay's configuration value. So, whenever a cleaning move is performed, a container recognized to be rehandled should be moved to a stack such that it would not be rehandled again. With this strategy, our aim is to find cleaning moves, if exist, that would potentially improve the bay configuration by reducing the future number of rehandles. Essentially, recognizing a cleaning move means, finding out whether there is a potential cleaning move that will reduce future number of rehandles.

Define the following notation:

a_i = The accessible container in stack i

m_j = The minimum container in stack j

$m_j = a_j = N+1$ if stack j is empty

$$mDist_{ij} = \begin{cases} m_j - a_i & \text{if } m_j > a_i \\ \infty & \text{else} \end{cases}$$

In Figure 4.5 we propose an algorithm to recognize cleaning moves. The algorithm searches stacks for finding a position for containers that are already recognized to be rehandled but won't be rehandled in the position that is found

$mDist_{\cdot\cdot} = \infty$
For all i
$j = \arg\min_{j, j \neq i} \left(mDist_{ij} \middle| mDist_{ij} < mDist_{\cdot\cdot} \right)$
If $mDist_{ij} < \infty$
    A potential cleaning move i to j exists
Else
    There is no cleaning move

Figure 4.5 Algorithm for Recognizing Cleaning Moves

4.2 Justification of a Cleaning Move for Case 1

4.2.1 Idea of Mobilized Containers

Mobilized containers, within a specific configuration, are those containers that have to be relocated before a given container. For example, in Figure 4.6 mobilized containers for container 21 are shaded. These containers will be relocated before 21 and one cannot know where they are placed when 21 is being rehandled.

	20			
	10			
8	11		17	19
9	21		16	24
6	14		12	4
15	5		22	13
3	18	23	7	25
i	ii	iii	iv	v

Figure 4.6 Mobilized containers for 21

Finding mobilized containers in Figure 4.6 is easy. For instance, container 21 will be rehandled to retrieve 5, so all of the containers located above containers smaller than 5 and containers above 21 will be relocated on some stack other than ii. Containers 3 and 4 will already be retrieved so they should not be considered as a part of mobilized containers for 21.

Let container Y, be the container to be retrieved and let container X be a container currently positioned on Y. Set of mobilized containers for X includes: containers placed above X and containers placed above containers smaller than Y. Any container that is ordered earlier than Y is excluded from the set. An additional information is that mobilized containers for X can be in any stack other than stack of X.

When we exclude mobilized containers for X and containers retrieved before Y, there will be untouched containers left in the bay. Because the locations of these containers will be known when X is being rehandled, they will be referred as known configuration for X.

4.2.2 Justification

Suppose a cleaning move from *i* to *j* is recognized. Then a_i is the container to be moved and m_j is the minimum container in the stack that a_i will be moved to. This move will avoid any further rehandles of a_i, but may cause any container Z, ordered between a_i and m_j to be rehandled unnecessarily. So if one can know either;

a. Z will not be rehandled if cleaning move is performed (Figure 4.2-b, c, d), or
b. Z will be rehandled even if cleaning move is not performed.
   Then cleaning move can be justified.

We can know whether the condition in (a) holds a by the idea of mobilized containers, but it is not easy to predict whether the condition imposed in (b) will occur. So, the condition b will be predicted by the expected number of additional rehandles of Z, again based on the idea of mobilized containers.

4.2.3 Probability Estimation

For container X, if known configuration and mobilized containers are given, we can estimate worst case probability of X to be rehandled, assuming random relocation movements. This probability can be found by equation (4-1).

For example in Figure 4.6 there are 9 mobilized containers for 21 and there are 4 available stacks for these containers to be placed (i, iii, iv, v). Container 21 will be rehandled if it is placed on top of 8 out of 9 mobilized containers and 21 will not be rehandled if it is placed in an empty stack (stack i will be empty when container 3 is retrieved) or directly on top of container 23 in stack iii. So, if none of those 8 containers is placed either in stack i or iii, then 21 may not be rehandled. This probability will be randomly placing 8 containers to available 4 stacks in known configuration of container 21 (i, iii, iv, v) and positioning none of the containers either in stack i or iii.

Let:

$mobilized_i$ = Set of mobilized containers for container *i*.
$mobilizedNum_i$ = Number of mobilized containers for i that will be retrieved earlier than i.
e_i = Number of stacks j where $a_i < m_j$ and $i \neq j$.
$P_{re-handle}(i)$ = Probability of container i to be rehandled.
f_i = Available number of stacks in a known configuration for container *i*.
$P(f_i)$ = Transition probability matrix of moving a container randomly in bay having f_i stacks available.
$P_{sm}^{n}(f_i)$ = Probability of having $m = e_i$ after n random container movements, starting from $s = e_i$ in a bay having f_i stacks available.

$$P_{re-handle}(i) \geq P_{e_i 0}^{mobilizedNum_i}(f_i) \tag{4-1}$$

We justify a cleaning move from *i* to *j* if either of equations (4-2) or (4-3) holds. (4-2) holds if $mobilizedNum_i = 0$.

$$\sum_{X=a_i}^{m_j} P_{re-handle}(X) = 0 \tag{4-2}$$

$$\sum_{X=a_i}^{m_j} P_{re-handle}(X) > 1 \tag{4-3}$$

For the cases when equation (4-2) holds, the number of mobilized containers would be 0. Thus when the cleaning movement is executed there would not be any container ordered between a_i and m_j to rehandled. Equation (4-3) states that if the cleaning move is not done, at least 1 container ordered between a_i and m_j is expected to be in stack *j* before a_i. This will make a_i to be rehandled if it is positioned to stack *j*. In a sense by cleaning a_i may avoid future relocations of a_i when equation (4-3) holds.

4.2.4 Transition Probability Matrix

Randomly locating containers in stacks can be modeled as a Discrete Time Markov Chain Process, assuming available stacks in the bay will not be full even after all containers are positioned and every placement of a container corresponds to a time period. At every random container placement, e_j will either decrease by one or will not change. So, current e_j only depends on previous e_j.

Transition probability matrix can be used to estimate equation (4-1). If the states of the Markov Chain model is defined as e_j, where e_j is an integer in the range $[0, f_i]$, then the transition probability matrix of the model will be formed as (4-4).

$$P^1(f_i) = \begin{pmatrix} 1 & 0 & 0 & . & . & . & 0 \\ \frac{f_i - 1}{f_i} & \frac{1}{f_i} & & & & & 0 \\ 0 & \frac{f_i - 2}{f_i} & \frac{2}{f_i} & & & & 0 \\ . & & . & . & & & . \\ . & & & . & . & & . \\ . & & & & \frac{1}{f_i} & \frac{f_i - 1}{f_i} & . \\ 0 & . & . & . & 0 & 1 & 0 \end{pmatrix} \qquad 4\text{-}4$$

For Figure 4.6, P will be:

$$P^1(4) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1/4 & 3/4 & 0 & 0 & 0 \\ 0 & 2/4 & 2/4 & 0 & 0 \\ 0 & 0 & 3/4 & 1/4 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

The justification of Figure 4.2-a will be done as:

$$P_{20}^{8}(4) + P_{20}^{12}(4) \;\; ?\; 1$$
$$(.80) + (.94) > 1 \quad .$$

4.3 Combining Cleaning Moves with Difference Heuristic

In the previous section we see that cleaning moves could improve the bound on the optimal solution as defined in Chapter 3. One could try to find optimal cleaning moves by branch and bound procedure but this will not be a viable alternative due to the following reasons:

- Branch and bound search is non-polynomial. Including cleaning moves into the search would both increase depth and branching of the search tree (compared to Chapter 3). Thus increase computation time of the algorithm which already is excessive.
- Secondly, our branch and bound search relies on branching according to actions (container movements ending with a retrieval), but cleaning movements may change with each container movement. This makes combining cleaning moves with the action based branching very challenging.

Because we cannot integrate the branch and bound method with the cleaning moves idea, we choose to combine the difference heuristic with cleaning moves; to which we refer as clean difference heuristic. The clean difference algorithm is given in Figure 4.7.

```
While (There is container to be retrieved)
        If (There is a cleaning move from i to j and it is justified)
                Move a_i to j
        Else
                Move container according to difference heuristic
```

Figure 4.7 Clean Difference Algorithm for Case 1

4.4 Modifying Justification for Case 2

For Case 2, our objective is not just to minimize the number of rehandles but to also include distances. To account for this we incorporated the cleaning move distance with a justification threshold. Equation (4-2) does not change but equation (4-3) is modified as equation (4-5). As defined in the previous chapter, A is the rehandling constant and B is the horizontal distance constant.

$$\sum_{X=a_i}^{m_j} P_{rehandle}(X) > \left(1 + \frac{B}{A} * |i-j|\right) \qquad 4\text{-}5$$

The algorithm does not really change except two small modifications: cleaning move is justified according to Case 2 and the difference heuristic for Case 2 is used (Figure 4.8).

```
While (There is container to be retrieved)
        If (There is a cleaning move from i to j and it is justified for Case 2)
                Move a_i to j
        Else
                Move container according to difference heuristic for Case 2
```

Figure 4.8 Clean Difference Algorithm for Case 2

4.5 Results

All of the 8000 instances are solved with clean difference heuristic for both Case 1 and Case 2. When the data is analyzed over all the instances, as seen in Table 4.1, cleaning do improves the difference heuristic for some instances, but generally the algorithm seems to work better without cleaning. This is true for especially Case 1.

If the average of the instances are analyzed, on the average, cleaning improves the algorithm for %0.3 in Case 1 but worsens by %8.0 in Case 2. Cleaning moves do not improve the solutions on average but for individual cases up to % 30.0 and %75.9 cost reduction can be gained, Table 4.2.

Table 4.1 Overall comparison for including cleaning in to difference heuristic

	Head to Head Comparison for Cleaning			
	Case 1		Case 2	
	Difference	Clean Difference	Difference	Clean Difference
Difference	0	1024	0	3967
Clean Difference	548	0	3939	0

Table 4.2 Individual instance comparisons in % change

Change (%)	Case 1	Case 2
Maximum	22.7	221.5
Minimum	-30.0	-75.9
Average	-0.3	8.0

We have calculated averages for each combination of parameters: initial container density, balanced vs. unbalanced layout, width and height of the bay. For Case 1 results are not interesting and nearly all combinations look alike except that the number of rehandles increases as initial container density, width and tier increases as well as unbalanced bays have higher number of rehandles. In the following figures: Figure 4.9, Figure 4.10, Figure 4.11, Figure 4.12, graphical analysis of the instances where bays are initially %55 filled and has width of 7 rows are given. Heuristics are represented with their initial letters.

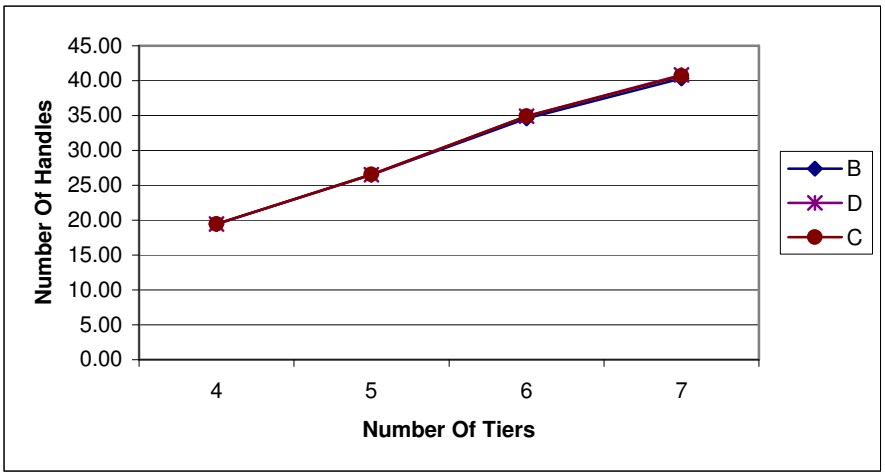


Figure 4.9 Effect of Increasing Bay Height on Number of Handles for Balanced Bays and Case 1

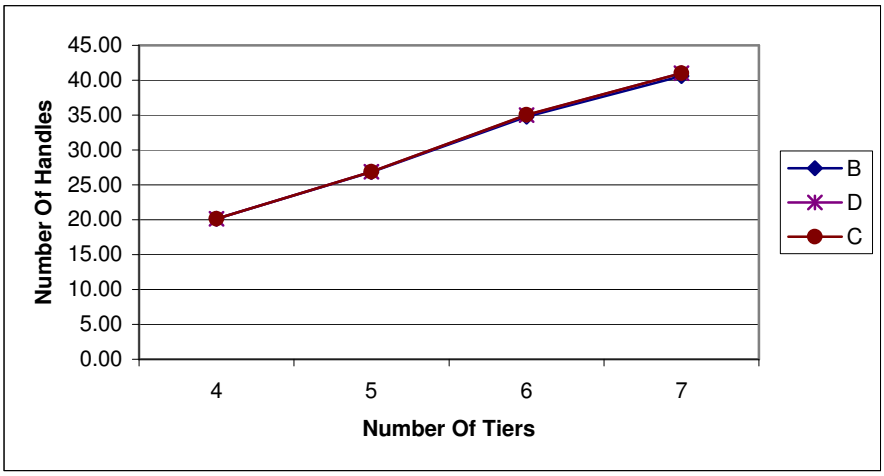


Figure 4.10 Effect of Increasing Bay Height on Number of Handles for Unbalanced Bays and Case 1

In Figure 4.9 and Figure 4.10 average of the solutions of branch and bound search and the difference heuristics proposed in Chapter 3 is compared with the clean difference heuristic for Case 1. Averages look nearly the same and heuristics cannot be really distinguished but branch and bound search values are slightly better than clean difference heuristic. For Case 2, following figures are given with the same parameters as in the figures above.

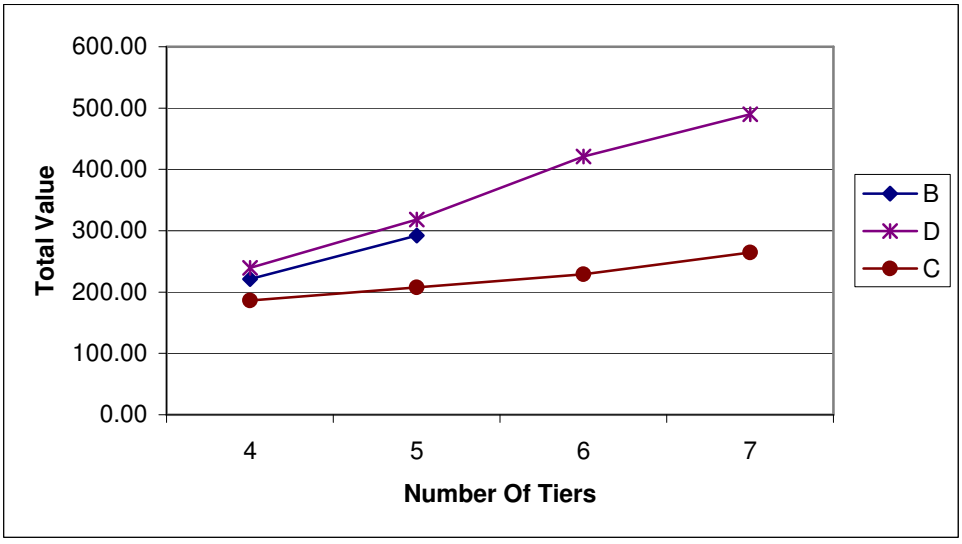


Figure 4.11 Effect of Increasing Bay Height on Number of Handles for Balanced Bays and Case 2

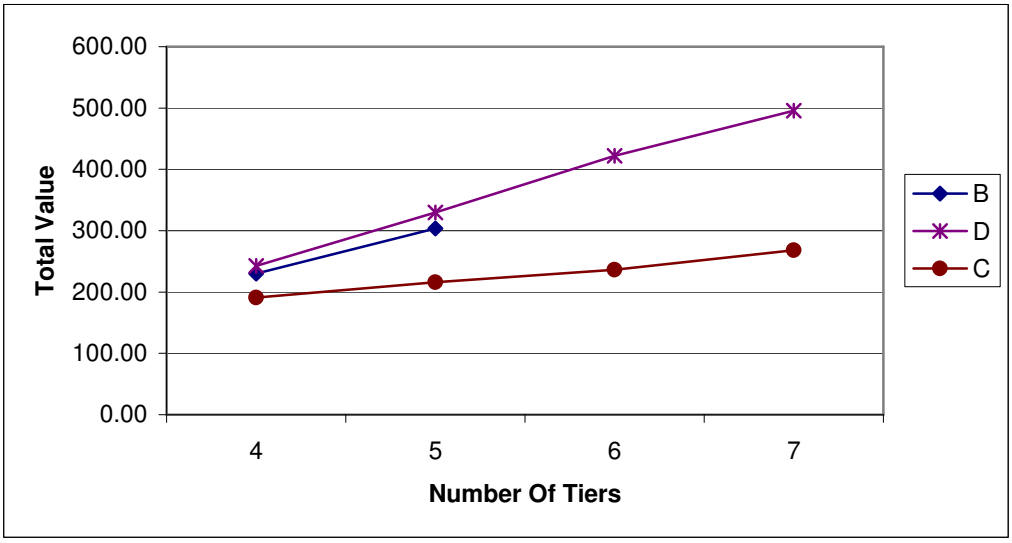


Figure 4.12 Effect of Increasing Bay Height on Number of Handles for Unbalanced Bays and Case 2

In Figure 4.11 and Figure 4.12 we can clearly see that cleaning improves the results significantly. Results are even much better than the branch and bound search proposed in Chapter 3. Also, improvement becomes more significant as the number of tiers increase. These graphics contradicts with the results of over all analysis given in

Table 4.2. Reason for this may be clarified with the help of Figure 4.13 where graphic of the solutions of instances with bays are 7 rows width, initially %70 filled and unbalanced.

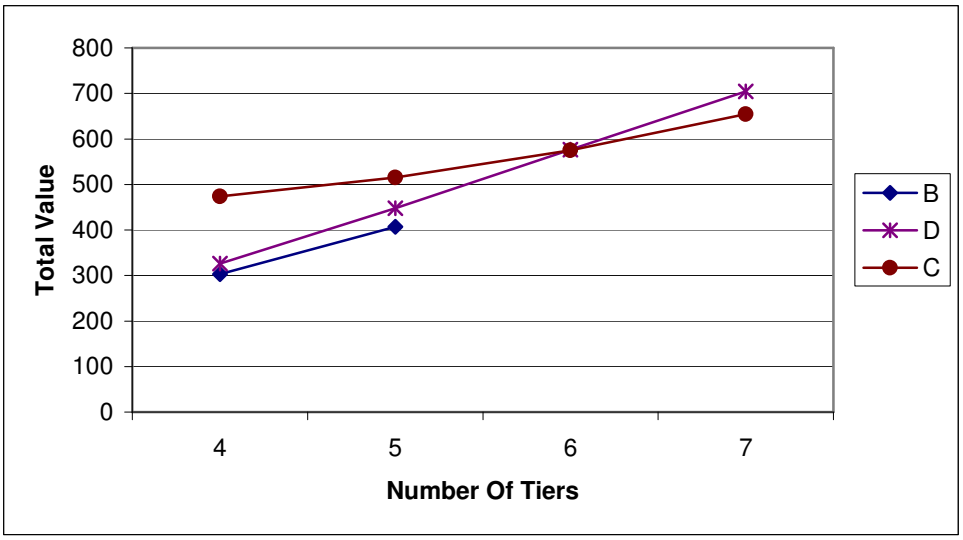


Figure 4.13 Effect of Increasing Bay Height on Number of Handles for Bays Initially %70 Occupied and Unbalanced, Case 2

One can see that for less number of tiers in the bay, cleaning moves are inefficient, compared the difference heuristic solutions without the cleaning moves. But as the tier height of the bay increases cleaning works better.

As a conclusion, we see that;

- Cleaning works and even improves the bound on the optimal solution.
- Justification of cleaning moves should be improved. Integrating tier height in the process may be useful.
- Random container movements is a loose assumption, which we have already seen in Chapter 3 and is reminded again in this chapter.
- Cleaning does not seem to work for Case 1, because even if moves act as cost reducing, difference heuristics increase the cost and improvement could not be recognized. If the problem size gets bigger, by increasing width and height of the bay, the improvement could be significant and easy to recognize.

5 CONCLUSION AND FUTURE WORK

In this thesis, we study problem of the retrieval of containers from their stacks. The problem is addressed in literature with the objective of minimizing number of rehandling movements. We also incorporated the distance traveled by the crane and minimized total number of rehandles plus the distance traveled. First objective is referred to as Case 1 and the latter as Case2. Branch and bound method is implemented to find optimal values in both cases. The optimal values indicated that distinguishing Case 2 from Case 1 may be useful. Also three heuristics namely, Expected Additional Relocations Heuristic (based on [28]), Greedy Heuristic and Difference Heuristics are developed for both of the cases. 8000 bay configurations in 200 different combinations are generated and heuristics are applied. Heuristic results compared with both the optimal solutions and the solutions obtained from the random container movements are analyzed. Cross comparisons of heuristics are performed.

W also present a new idea, referred as the cleaning move, which is basically altering the sequence of relocation movements to further reduce the workload of the crane. This idea is implemented with the difference heuristic. On average introducing cleaning moves does not improve the solutions, but huge amount of cost reduction is found for some instances.

Future research directions in this area may be as follows:

- Cleaning move concept can be discussed more deeply and the effectiveness of the idea can be increased.
- Finding optimal cleaning moves and development of heuristics can be studied.
- The assumption that each container has different retrieval times. This can be generalized to container groups having different retrieval times.
- Relocations on two dimensions (within the block) can be considered.

6 REFERENCES

[1] D. Ambrosino, A. Sciomachen, E. Tanfani, "Stowing a Containership: The Master Bay Plan Problem", Transportation Research Part A: Policy and Practice, 38(2), 81-99, February 2004.

[2] A. Ashar, "On selectivity and accessibility", Cargo Systems, 44-45, June 1991.

[3] M. Bielli, A. Boulmakoul, M. Rida, "Object Oriented Model for Container Terminal Distributed Simulation", European Journal of Operational Research, Available Online, 2005.

[4] B. De Castilho, C. F Daganzo, "Handling Strategies for Import Containers at Marine Terminals", Transportation Research B, 27(2), 151-166, 1993.

[5] T. Chen, "Yard operations in the container terminal: A Study in the Unproductive Moves", Maritime Policy & Management, 26(1), 27-38, 1999.

[6] H. R. Choi, H. S. Kim, B. J. Park, N. K. Park, S. W. Lee, "An ERP Approach for Container Terminal Operating Systems", Maritime Policy & Management, 30(3), 197-211, July 2003.

[7] L. L. Chung, G. L. Vairaktarakis, "Loading and Unloading Operations in Container Terminals", IIE Transactions, 36(4), 287- 298, April 2004.

[8] Y. G. Chung, S.U. Randhawa, E. D. McDowell, "A Simulation Analysis for a Transtainer-Based Container Handling Facility", Computers & Industrial Engineering, 14(2), 113-125, 1998.

[9] Containerization and Intermodal Institute, www.containerization.org

[10] C. F. Daganzo, "The Crane Scheduling Problem", Transportation Research-B, 23(3), 159–175, 1989.

[11] J. N. Ece, "Denizcilik Sektörünün Özelleştirilmesi", Özelleştirme Stratejileri Paneli, www.turkishpilots.org.tr/DOCUMENTS/J_Nur_Ece_Ozellestirme.htm

[12] L. M. Gambardella, M. Mastrolilli, A. E. Rizzoli, M. Zaffalon, "An Optimization Methodology for Intermodal Terminal Management", Journal of Intelligent Manufacturing, 12(5)-(6), 521-534, October 2001.

[13] S. Hartmann, "Generating Scenarios for Simulation and Optimization of Container Terminal Logistics", OR Spectrum, 26(2), 171-193, March 2004.

[14] Hong Kong Port Development Council, www.pdc.gov.hk/eng/statistics/docs/KT-stat.pdf

[15] D. L. Howard, M. J. Bragen, J. F. Burke, R. J. Love, "PORTSIM 5: Modeling From a Seaport Level", Mathematical and Computer Modeling, 39(6)-(8), 715-731, March 2004.

[16] M. T. Ibrahimi, B. De Castilho, C. F. Daganzo, "Storage Space vs. Handling Work in Container Terminals", Transportation Research B, 27, 13-32, 1993.

[17] A. Imai, E. Nishimura, S. Papadimitriou, "The Dynamic Berth Allocation Problem For a Container Port", Transportation Research-B, 35(4), 401–417, 2001.

[18] A. Imai, E. Nishimura, S. Papadimitriou, "Berth Allocation with Service Priority", Transportation Research Part B: Methodological, 37(5), 437- 457, June 2003.

[19] A. Imai, K. Sasaki, E. Nishimura, S. Papadimitriou, "Multi-Objective Simultaneous Stowage and Load Planning for a Container Ship with Container Rehandle in Yard Stacks", European Journal of Operational Research, Available Online, December 2004.

[20] A. Imai, X. Sun, E. Nishimura, S. Papadimitriou, "Berth Allocation in a Container Port: Using a Continuous Location Space Approach", Transportation Research Part B: Methodological, 39(3), 199-221, March 2005.

[21] Japan International Cooperation Agency (JICA), "Master Plan of National Port Development", www.jica.go.jp

[22] K. H. Kim, "Evaluation of the Number of Rehandles in Container Yards", Computers & Industrial Engineering, 32(4), 701-711, September 1997.

[23] K. H. Kim, J. W. Bae, "Re-Marshaling Export Containers in Port Container Terminals", Computers & Industrial Engineering, 35(3)-(4), 655-658, December 1998.

[24] K. H. Kim, Y. M. Park, K. R. Ryu, "Deriving Decision Rules to Locate Export Containers in Container Yards", European Journal of Operational Research, 124, 89-101, 2000.

[25] K. H. Kim, K. C. Moon, "Berth Scheduling by Simulated Annealing", Transportation Research Part B: Methodological, 37(6), 541-560, July 2003.

[26] K. H. Kim, S. H. Won, J. K. Lim, T. Takahashi, "An Architectural Design of Control Software for Automated Container Terminals", Computers & Industrial Engineering, 46(4), 741-754, July 2004.

[27] K. H. Kim, Y. M. Park, "A Crane Scheduling Method for Port Container Terminals", European Journal of Operational Research, 156(3), 752-768, August 2004.

[28] K. H. Kim, G. P. Hong, "A Heuristic Rule for Relocating Blocks", Computers & Operations Research, 33, 940-954, July 2006.

[29] Marport Liman İşletmeleri, www.marport.com.tr

[30] K. G. Murty, J. Liu, Y. Wan, R. Linn, "A Decision Support System for Operations in a Container Terminal", Decision Support Systems, 39(3), 309-332, May 2005.

[31] A. Narasimhan, U. S. Palekar, "Analysis and Algorithms for the Transtainer Routing Problem in Container Port Operations", Transportation Science, 36(1), 63-78, February 2002.

[32] Y. M. Park, K. H. Kim, "A Scheduling Method for Berth and Quay Cranes", OR Spectrum, 25(1), 1-23, February 2003.

[33] R. I. Peterkofsky, C. F. Daganzo, "A Branch and Bound Solution Method for the Crane Scheduling Problem", Transportation Research Part B: Methodological, 24(3), 159-172, June 1990.

[34] S. P. Sgouridis, D. Makris, D. C. Angelides, "Simulation Analysis for Midterm Yard Planning in Container Terminal ", Journal of Waterway, Port, Coastal & Ocean Engineering, 129(4), 178-188, July 2003.

[35] D. Steenken, S. Voß, R. Stahlbock, "Container Terminal Operation and Operations Research: A Classification and Literature Review", OR Spectrum, 26(1), 3-49, January 2004.

[36] L. K. Tranberg, "Optimizing Yard Operations in Port Container Terminals", Proceedings of the 10th EWGT Meeting and 16. Mini Euro Conference, 386-391, 2005.

[37] Türkiye Cumhuriyeti Devlet Demiryolları Liman İşletmeleri, www.tcdd.gov.tr/liman/konteyner.htm

[38] T. Ünlüyurt, H. M. Özdemir, "Space Allocation and Location Matching in Container Terminals", Proceedings of the 10th EWGT Meeting and 16. Mini Euro Conference, 367-372, 2005.

[39] F. A. Vis, R. Koster, "Transshipment of Containers at a Container Terminal: An Overview", European Journal of Operational Research, 147(1), 1-16, May 2003.

[40] I. Watanabi, "Selection Process", Cargo Systems, 35-36, March 1991.

[41] M. A. Weiss, "Data Structures & Algorithm Analysis in C++", Chapter 8, Addison Wesley, 2001

[42] D. Wilson, P. A. Roach, "Principles of Combinatorial Optimization Applied to Container-Ship Stowage Planning", Journal of Heuristics, 5(4), 403-418, December 1999.

[43] H. Yang, Y. S. Choi, T. Y. Ha, “Simulation-Based Performance Evaluation of Transport Vehicles at Automated Container Terminals”, OR Spectrum, 26(2), 149-170, March 2004.

[44] C. Q. Zhang, Y. W. Wan, J. Liu, R. J Linn, “Dynamic Crane Deployment in Container Storage Yards”, Transportation Research Part B, 36, 537-555, 2002.

[45] C. Zhang, J. Liu, Y. Wan, K. G. Murty, R. J. Linn, “Storage Space Allocation in Container Terminals”, Transportation Research Part B: Methodological, 37(10), 883-903, December 2003.

[46] I. Zyngiridis, “Optimizing Container Movements Using One and Two Automated Stacking Cranes”, Master Thesis, Naval Post Graduate School Monterey California, 2005

APPENDIX

A Number of instances solved to optimality using branch and bound method

		CASE 1								CASE 2							
		4		5		6		7		4		5		6		7	
		B	U	B	U	B	U	B	U	B	U	B	U	B	U	B	U
3	55	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40
	60	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40
	65	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40
	70	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40
	75	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40
4	55	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40
	60	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40
	65	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40
	70	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40
	75	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40
5	55	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40
	60	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40
	65	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40
	70	40	40	40	40	40	40	40	40	40	40	40	40	40	40	2	0
	75	40	40	40	40	40	40	40	40	40	40	40	40	40	40	0	0
6	55	40	40	40	40	40	40	40	40	40	40	40	40	40	40	0	0
	60	40	40	40	40	40	40	40	40	40	40	40	40	40	40	0	0
	65	40	40	40	40	40	40	4	0	40	40	40	40	39	4	0	0
	70	40	40	40	40	40	40	0	0	40	40	40	40	0	0	0	0
	75	40	40	40	40	40	40	0	0	40	40	40	40	0	11	0	0
7	55	40	40	40	40	40	40	40	40	40	40	40	40	0	0	0	0
	60	40	40	40	40	40	40	40	40	40	40	40	40	0	0	0	0
	65	40	40	40	40	0	0	0	0	40	40	40	40	0	0	0	0
	70	40	40	40	40	0	0	0	0	40	40	40	40	0	0	0	0
	75	40	40	40	40	0	0	0	0	40	40	40	40	0	0	0	2

B Branch and Bound Solution Times For Case 1 (msec)

		Case 1								Case2							
		Balanced				Unbalanced				Balanced				Unbalanced			
		4	5	6	7	4	5	6	7	4	5	6	7	4	5	6	7
55	3	1	0	1	0	0	1	1	0	0	0	0	1	0	1	1	0
	4	0	0	1	6	0	1	2	21	0	1	3	334	1	2	23	990
	5	1	0	4	4341	1	0	4	1187	1	4	281	19798	1	295	2062	674073
	6	1	2	11	5481	1	0	93	10414	2	12	1342		12	250	38346	
	7	1	2	173	33686	0	13	971	19876	15	480			46	32120		
60	3	0	1	0	0	0	0	1	1	1	0	1	2	0	1	1	0
	4	0	1	2	23	1	0	2	15	0	2	14	2061	1	2	23	990
	5	0	2	9	1541	0	2	47	3577	1	67	815	107512	1	295	2062	674073
	6	1	3	30	101152	2	9	1223	38379	2	554	349566		12	250	38346	
	7	1	3	2615	543072	1	25	5378	1194229	10	5462			46	32120		
65	3	1	0	0	1	0	2	0	3	1	1	0	2	0	2	1	4
	4	0	2	10	66	0	1	2	47	1	3	200	1145	1	4	37	4472
	5	1	1	97	1719	2	2	59	4559	4	31	9147	1211428	1	38	10367	259032
	6	1	5	1342	60239	0	4	1800		3	1602	196258		9	5057	21611907	
	7	0	60			3	76			315	8194			209	207602		
70	3	0	1	1	1	0	1	1	1	0	1	2	5	0	0	0	5
	4	0	1	19	296	0	0	2	51	1	6	161	2292	0	3	44	1944
	5	1	2	206	3482	0	3	545	272953	4	96	28470	3992415	2	121	5500	
	6	2	29	13655		2	17	1389		22	1295			34	5243		
	7	2	160			2	362			43	142433			90	111460		
75	3	0	0	1	1	0	1	1	2	0	1	1	9	1	0	1	3
	4	0	2	7	849	0	1	6	878	0	5	308	60425	1	4	452	94272
	5	1	8	1280	44118	1	8	227	38790	4	120	700249		2	82	42201	
	6	1	448	308002		2	65	24489		54	161657			58	2448	5290603	
	7	2	1839			5	3257		16352	790	2282543			411	632230		

C Branch and Bound Solution Times For Case 2 (msec)

		Case 1								Case2							
		Balanced				Unbalanced				Balanced				Unbalanced			
		4	5	6	7	4	5	6	7	4	5	6	7	4	5	6	7
55	3	3.5	2.5	3.3	2.5	2.4	4.2	3.4	2.5	2.5	0.0	2.4	4.1	4.1	4.2	0.0	3.4
	4	2.5	2.4	5.1	14.8	2.5	3.4	5.2	72.4	2.4	4.2	7.2	1058.1	3.4	6.3	28.3	2178.7
	5	3.4	2.5	7.5	25813.9	3.4	0.0	9.2	5971.9	3.3	14.4	654.2	62677.1	2.5	20.6	9458.4	420229.1
	6	3.5	5.1	30.7	23832.2	3.4	2.5	262.6	35425.4	4.9	34.8	2767.8		7.4	10641.3	33810.8	
	7	3.4	7.0	608.7	97205.8	2.5	61.8	4066.6	55054.6	67.3	1483.7			18.7	2421.5		
60	3	2.5	3.3	0.0	0.0	2.4	2.4	3.4	4.0	3.5	0.0	3.4	4.6	0.0	3.5	3.3	2.4
	4	0.0	3.5	4.7	52.8	3.5	2.5	5.9	38.7	0.0	4.7	29.3	11629.1	4.3	5.7	67.8	3556.9
	5	2.4	6.4	21.5	5068.3	0.0	4.8	121.1	14236.4	3.5	305.9	1788.6	370350.6	3.5	1626.9	8611.5	2224737.8
	6	3.5	12.8	53.4	405475.2	5.2	31.6	5887.3	197820.7	8.3	2356.7	1490365.4		54.3	1064.3	105466.9	
	7	3.3	7.2	8927.0	1873206.9	4.2	126.6	25866.4	4672842.8	28.0	15459.3			148.9	181014.6		
65	3	4.2	2.4	0.0	4.1	0.0	4.8	2.5	6.1	3.5	3.4	2.5	6.3	2.4	4.8	4.1	9.2
	4	2.5	5.2	24.8	175.9	2.5	3.5	5.7	114.7	3.4	12.9	712.0	2652.2	3.4	7.7	127.4	13721.5
	5	3.3	4.1	374.9	4290.8	4.9	5.3	168.6	18296.2	17.7	80.2	41822.8	3883538.4	4.1	92.7	44571.8	570886.6
	6	4.2	16.3	4249.3	69785.2	2.5	17.6	7157.7		6.5	4696.6	478941.0		17.0	24200.0	37425292.4	
	7	0.0	218.8			6.4	388.7			1271.1	23488.7			621.4	893766.0		
70	3	2.4	4.3	4.2	3.3	2.4	3.4	3.4	4.2	0.0	4.1	5.7	13.1	2.5	2.5	2.5	7.6
	4	2.5	4.2	59.0	1163.5	0.0	2.4	4.8	94.4	4.2	15.4	435.5	4747.4	2.5	7.4	78.9	5810.7
	5	4.3	8.2	529.5	11913.4	2.4	9.3	1758.3	1618004.4	7.7	415.4	66792.9	155330.1	4.7	361.8	12594.5	
	6	5.2	49.5	55814.3		7.8	41.4	3331.1		56.3	2713.9			124.6	26931.5		
	7	4.9	484.3			6.1	1234.4			108.2	432867.7			221.9	399907.1		
75	3	2.5	0.0	4.3	3.4	0.0	3.3	4.3	4.8	0.0	3.5	3.5	42.0	4.2	0.0	4.1	7.2
	4	0.0	4.9	15.8	2819.8	2.5	3.3	12.2	4056.5	0.0	7.4	1010.1	267761.1	3.4	7.7	2161.8	552084.9
	5	4.2	30.6	3941.4	115127.1	3.4	17.7	473.5	134118.9	9.9	258.0	4239465.5		4.8	188.5	163855.5	
	6	4.1	2533.8	1702214.7		6.2	328.6	92276.7		173.9	995611.1			248.6	4839.2	10397341.2	
	7	4.8	10165.3			20.1	16730.3		21423.2	3522.8	10091662			1060.0	3302761.2		

D Comparison of Branch and Bound Solution Times (sec)

	Number of Containers	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	29	30
This Work	Number of Problems	400	240	480	400	560	400	640	560	80	560	560		650	320	240	160	240	240	84	80	
	Solution Time	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.02	0.44		0.64	1.05	3.36	69.24	25.34	23.59	161.20	868.65	
[28]	Number of Problems	40			40			40	40		40		80	80			80	40				40
	Solution Time	7.3			8			13.2	14.8		14.5		41.45	136.15			147.35	223.3				2657.5

E Ration of Additional Movements Between Case 1 and Case 2

	Ratio of Additional Movements For the Balanced Bays																			
	4					5					6					7				
	55	60	65	70	75	55	60	65	70	75	55	60	65	70	75	55	60	65	70	75
3	0.001	0.007	0.002	0.002	0.005	0.006	0.014	0.009	0.005	0.006	0.016	0.008	0.008	0.007	0.011	0.013	0.012	0.011	0.006	0.011
4	0.013	0.017	0.015	0.009	0.011	0.020	0.026	0.018	0.022	0.021	0.026	0.019	0.029	0.024	0.027	0.031	0.028	0.030	0.024	0.020
5	0.023	0.026	0.019	0.041	0.023	0.014	0.033	0.029	0.029	0.031	0.042	0.038	0.044	0.038	0.032	0.040	0.039	0.051	0.051	
6	0.016	0.020	0.028	0.042	0.037	0.029	0.035	0.041	0.039	0.043	0.032	0.050	0.052							
7	0.024	0.025	0.039	0.038	0.039	0.038	0.037	0.050	0.063	0.048										

	Ratio of Additional Movements For the Unbalanced Bays																			
	4					5					6					7				
	55	60	65	70	75	55	60	65	70	75	55	60	65	70	75	55	60	65	70	75
3	0.012	0.007	0.006	0.011	0.005	0.012	0.013	0.011	0.008	0.006	0.009	0.013	0.010	0.007	0.011	0.010	0.010	0.009	0.009	0.011
4	0.009	0.016	0.008	0.014	0.012	0.021	0.022	0.028	0.018	0.019	0.022	0.021	0.024	0.018	0.021	0.034	0.027	0.030	0.023	0.027
5	0.015	0.020	0.026	0.022	0.022	0.020	0.034	0.022	0.036	0.030	0.039	0.030	0.048	0.036	0.034	0.043	0.045	0.031		
6	0.023	0.026	0.026	0.033	0.030	0.028	0.036	0.035	0.041	0.040	0.045	0.049	0.074		0.043					
7	0.027	0.041	0.032	0.040	0.042	0.050	0.037	0.048	0.059	0.063										