STATIC AND DYNAMIC CONTAINER RELOCATION PROBLEMS: FORMULATIONS AND EFFICIENT HEURISTIC PROCEDURES

(STATİK VE DİNAMİK KONTEYNER PROBLEMİLERİ İÇİN FORMÜLASYONLAR VE ETKİN SEZGİSEL YAKLAŞIMLAR)

by

Osman KARPUZOĞLU

Thesis

Submitted in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

in

LOGISTICS AND FINANCIAL MANAGEMENT

in the

GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

of

GALATASARAY UNIVERSITY

May 2016

This is to certify that the thesis entitled

STATIC AND DYNAMIC CONTAINER RELOCATION PROBLEMS: FORMULATIONS AND EFFICIENT HEURISTIC PROCEDURES

prepared by **Osman KARPUZOĞLU** in partial fulfillment of the requirements for the degree of **Master of Science in Logistics and Financial Management** at the **Galatasaray University** is approved by the

Examining Committee:

Yrd. Doç. Dr. M. Hakan AKYÜZ (Supervisor)
Department of Industrial Engineering
Galatasaray University ----------

Prof. Dr. Temel ÖNCAN
Department of Industrial Engineering
Galatasaray University ----------

Doç. Dr. İbrahim MUTER
Department of Industrial Engineering
Bahçeşehir University ----------

Date: ----------

ACKNOWLEDGEMENTS

I would first like to thank my family, especially Mom and Dad, for the continuous support they have given me throughout my time in graduate school; I could not have done it without them. Second, I would like to express my sincere gratitude to my advisors Prof. Temel Öncan and Assist. Prof. M. Hakan Akyüz for the continuous support of my master study and research, for their patience, motivation and immense knowledge. Their guidance helped me in all the time of research and writing of this thesis. I could not have imagined having better advisors and mentors for my master study.

May 2016
Osman Karpuzoğlu

TABLE OF CONTENTS

LIST OF SYMBOLS

CRP	**:** Container relocation problem
DCRP	**:** Dynamic container relocation problem
PI	**:** Path index
TSA	**:** Tabu search algorithm
BS	**:** Beam search
MMD	**:** Min Max heuristic algorithm
MMDJV	**:** Modified MMD heuristic by using Javonovic-Voss heuristic
WLT	**:** Original CRP formulation presented by Wan et al. (2009)
WLT-DCRP	**:** Modified version of WLT-II for DCRP
RI	**:** Reshuffling index heuristic
BRP	**:** Block relocation problem

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

The container relocation problem (CRP) which is known to be NP-hard, tries to empty a single yard-bay which contains *S* containers each having a given retrieval order so as to minimize the total number of relocations performed. The DCRP is an extension of the CRP where containers are both received and retrieved from a single yard-bay and the arrival and departure sequences of containers are assumed to be known in advance. Two enhanced Binary Integer Programming (BIP) formulations for the CRP and a novel BIP formulation for the DCRP are devised. Computational experiments are performed to analyze new formulations by using standart test instances from the literature. Our results show that, new formulations are promising and yield better results in general for both CRP and DCRP. A new heuristic called as Path Index heuristic, is proposed to solve the CRP. Tabu search based heuristic approaches are proposed to solve the DCRP. In addition, two Index Based heuristics are developed and tested for the DCRP. Computational experiments are performed on an extensive set of test instances from the literature. Our results indicate that the proposed algorithms are efficient and yield promising outcomes. Especially, IB heuristics show a superior performance than the ones from the literature on a set of standard test instances for the DCRP.

Keywords: container stacking, integer programming, container relocation, heuristics, container terminals.

ÖZET

Dünya deniz ticaretinin % 50'sinden fazlasını konteyner taşımacılığı oluşturur. Günümüzde teknolojik gelişmeler daha büyük ve yüksek hızlı gemilerin üretimine olanak vermiştir. Bu nedenle konteyner terminallerini öncesine göre daha yüksek miktarlarda konteyner taşıma durumunda kalmıştır. Özellikle mega konteyner gemilerinin ortaya çıkışıyla konteyner terminallerinin öncesine göre daha iyi organize edilmesini bir zorunluluk haline gelmiştir. Bütün bunlar göz önüne alındığında, konteyner terminallerinin etkin yönetimi büyük önem taşımaktadır.

Konteyner terminal alanı rıhtım ve depolama alanı olarak ikiye ayrılmaktadır. Genel olarak, liman işletmecileri rıhtım tahsisi, rıhtım vinci atama ve zamanlama ve depolama planlaması gibi rıhtım alanı operasyonlarına daha fazla önem vermektedir. Depolama alanı işlemleri, rıhtım tarafından konteynerlerin transfer edilmesi, vinç planlaması, konteyner işlemlerinin halledilmesinden oluşur. Depolama alanı işlemlerinin önemi terminal operatörleri tarafından sıklıkla gözardı edilmektedir. Öte yandan depolama alanı operasyonlarının etkin yönetimi iskele alanı işlemlerinin başarısı ile doğrudan bağlantılıdır.

Bir konteyner sırası belirli yükseklik ve ve sütun sayısına sahiptir. Sütundaki konteynerlere erişim sadece yukarıdan yapılmaktadır. Hedef konteynerin üzerindeki konteynerleri, hedef konteyner alınmadan önce başka sütunlara taşınması gerekmektedir. Bu taşınma olayı yer değiştirme olarak tanımlamaktadır. Bu çalışmada depolama alanı problemlerinden olan konteyner yer değiştirme problemine odaklanılmıştır. Konteyner yer değiştirme problemi, konteynerlerin başlangıç düzeninin ve konteynerlerin ayrılış sırasının belli olduğu bir konteyner sırasını boşaltılır yapılan yer değiştirme sayısını en azlamayı hedefler. Konteyner yer değiştirme problemi sadece konteyner sırasından ayrılan konteynerlerin olduğu durağan bir problemdir. Dinamik konteyner yer değiştirme

probleminde ek olarak konteyner sırasına eklenen geliş zamanı bilinen gelen konteyner vardır. Bu çalışmada öncelikle konteyner yer değiştirme problemi literatürdeki gösterimler literatürdeki sınama ortamı kullanılarak karşılaştırılmış ve literatürden seçilen bir gösterim modifiye edilerek konteyner yer değiştirme problemi için daha iyi sonuçlar alınmıştır. Bu yeni gösterim üzerinde değişiklikler yapılarak ikinci bir gösterim elde edilmiştir. Yeni elde edilen iki gösterim literatürdeki gösterimlerle karşılaştırılmış ve önerdiğimiz gösterimlerin daha iyi performans sergilediği gözlenmiştir. Konteyner yer değiştirme problemi için yapılmış gösterim dinamik probleme uyarlanarak ücüncü bir gösterim elde edilmiştir. Sonuçlarda literatürdeki dinamik gösterimlere göre daha fazla işlem hesaplayabildiği gözlenmiştir. Bu gösterimlerden yola çıkarak değişik sezgisel yöntemler konteyner yer değiştirme problemleri için oluşturulmuştur. Sezgiselllerin ilki konteyner yer değiştirme problemi için oluşturulmuştur ve küçük örneklemler üzerinde başarılı olmuştur fakat büyük örneklemlerde aynı performansı sergileyememiştir. Diğer üç sezgisel dinamik konteyner yer değiştirme problemi için tasarlanmıştır. İlk olarak tabu arama sezgiseli ile literatürdeki bir indeks tabanlı sezgiselin karışımı şeklinde üretilen method literatürdeki sezgisellere göre bir gelişme gösterememiştir. Daha sonrasında literatürdeki bir konteyner yer değiştirme problemi sezgiselinin dinamik probleme uyarlanması sonucu iki adet sezgisel oluşturulmuş ve bunlar literatürdeki mevcut sezgisellere göre daha iyi performans göstermişlerdir.

1. INTRODUCTION

More than 50% of the world sea borne trade in terms of dollars are carried with containerized cargo (UNCTAD, 2014). Drastic changes in emerging technologies such as increased speed and size of vessels; enforce, container terminals has to transfer larger amounts of containers than before. In particular, with the introduction of mega container vessels, well organized container terminal operations are needed nowadays. Therefore, efficient management of container terminals is crucial.

The container terminal area can be separated into two: quay side area and yard side area. In general, terminal operators give more priority to quay side area operations which include berth allocation, quay crane assignment and scheduling, and vessel storage planning. Yard side operations include transferring containers from quay side, yard crane scheduling, and storing and handling of containers at the yard storage area. The importance of yard side operations is usually ignored by terminal operators since they mostly charge liner shipping companies according to the number of containers handled with quay cranes. On the other hand, efficiency of yard side highly interrelated with the success of quay side operations.

A yard area includes blocks of containers which is illustrated with Figure 1.1. A yard-bay is served with a yard crane so that containers are received and retrieved at top of the columns. Containers on top of a column are directly accessible for retrieval. However, if a target container (a container that will be retrieved from yard-bay immediately) isn't positioned at top of a column, then all containers above the target container have to be relocated to other columns of the yard-bay. Once blocking containers are cleared, target container can be retrieved. These clearing movements are called as relocations. Relocations are idle operations for yard cranes.

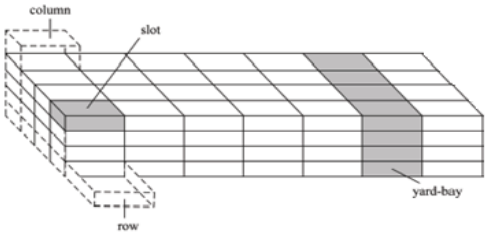

Figure 1.1: A block of containers. Source Akyüz and lee (2014)

We refer to the works by Steenken et al. (2004) and Stahlbock and Voβ (2008) as excellent surveys on container terminal operations. Two problems, which arise in the yard side of the container terminals, is focused in this work. We first address the Container Relocation Problem (CRP) which aims to minimize the total number of relocations accomplished to empty out a single yard-bay with a capacity of *C* columns (stacks) having a height of *P* rows (tiers) where *S* containers, whose retrieval sequences is given a priori, are initially located within the yard-bay. The CRP is a static problem in the sense that it only considers departure of containers from the yard-bay. We also consider a dynamic extension of the CRP where containers arrive and depart at the yard-bay, namely the Dynamic Container Relocation Problem (DCRP).

The CRP is an intensively studied problem which is introduced by Kim and Hong (2006). It is synonymously referred to as the "*Blocks Relocation Problem*" in the works by Caserta et al. (2012, 2011). In the CRP relocations can only occur when a container has to be taken out from the yard-bay. Thus, pre-marshalling is not allowed to reduce the number of future relocations henceforth. The CRP is also called as "restricted" CRP when this assumption is made. On the other hand, "unrestricted" CRP allows pre-marshalling by relaxing this assumption. Here, we follow the same framework described by Kim and Hong (2006) and limit ourselves with the restricted CRP.

DCRP is more realistic extension of the CRP when containers also arrive at the yard-bay. Given arrival and retrieval sequences of *S* containers, the DCRP tries to minimize the total number of relocations in a yard-bay. The DCRP inherits and extends the assumptions of the CRP such that it also considers the case of container arrivals.

We suggest two mathematical programming formulations for the CRP. Our formulations enhance the one originally proposed by Wan et al. (2009) and yields promising outcomes. Existing formulations and the ones presented in thesis are compared on standard benchmark instances from the literature. A novel mathematical programming formulation is developed which enhances the formulation previously offered by Akyüz and Lee (2014) for the DCRP. In light of these information we propose efficient heuristics for both CRP and DCRP.

In the reminder of this work is organized as follows. Section 2 gives a brief review of the literature for the CRP and DCRP. Section 3 presents definition of CRP problem and our solution methods for it. Section 4 contains DCRP problem definition and our brand new formulation for DCRP. Furthermore we present new heuristic approaches for DCRP in section 4. Numerical experiments given in section 5. Lastly, section 6 present our conclusions and a discussion for future research directions.

2. LITERATUR REVIEW

Kim et al. (2000) propose a dynamic programming model to minimize the total number of relocations where containers are grouped based on their weights. Kim and Hong (2006) propose a branch and bound (BB) algorithm and offer a rule of thumb heuristic procedure for the CRP. Since the seminal work by Kim and Hong (2006), there exist several studies addressing the CRP. The first mathematical programming formulation of the CRP is presented in the work by Wan et al. (2009) in which efficient heuristics are also presented. Caserta et al. (2011) develop a metaheuristic algorithm which employs a dynamic programming scheme for the CRP. An efficient tree search procedure for the CRP is offered by Forster and Bortfeldt (2012). Caserta et al. (2012) show that the CRP is NP-hard and suggest two formulations for the CRP. These formulations solve unrestricted CRP and restricted CRP formulation, respectively. Ünlüyurt and Aydin (2012) minimize the total time to empty a single yard-bay by a BB algorithm and propose heuristic procedures. Petering and Hussein (2013) introduce a new look-ahead algorithm that yields better solutions than other algorithms presented by Kim and Hong (2006), Lee and Lee (2010), Caserta et al. (2011), Ünlüyurt and Aydin (2012) for the CRP. Jovanovic and Voss (2014) implement a chain heuristic based on the "Max-Min" (MM) algorithm of Caserta et al. (2012) and offer an improvement on the MM algorithm for the CRP. For an in-depth discussion on container terminal operations and on stacking problems in storage areas, we refer to excellent surveys by Stahlbock and Voss (2008) and Lehnfeld and Knust (2014), respectively. Lehnfeld and Knust (2014) offer a classification scheme which covers other variants of the stacking problems as well as their complexity results that exist in the literature. Recently, Jin et al. (2015) develop a greedy look-ahead solution procedure which is employed for both restricted and unrestricted variants of the CRP as well as grouped and individual containers. Their tree search based approach yields the best results in shorter running times than the previous heuristic procedures in the literature. A modification on the formulation of Caserta et al. (2012) is offered by

Expòsito-Izquierdo et al. (2015) and they implement a BB algorithm for the CRP. Zehendner et al. (2015) made correction on the second formulation presented by Caserta et al. (2012) and offer an improved alternative CRP formulation. A preprocessing strategy is applied to improve the performance of the alternative formulation. Recently, Ku and Arthanari (2016) follow a new perspective and offer an abstraction method. This method significantly reduces the search space of the CRP and exactly solves small to medium size test instances from the literature in reasonable computing times.

In contrast to CRP, the researchers did not give much attention to the DCRP. Wan et al. (2009) is the first work which introduces the DCRP. Rei and Pedroso (2013) work with the DCRP denominated as Stacking Problem (SP), which is shown to be NP-hard, where containers arrive and depart at a single yard-bay. Akyüz and Lee (2014) propose a mathematical programming formulation including a Beam Search (BS) heuristic for the DCRP. Konig et al. (2007) address a closely related problem to the DCRP where they deal with the stacking of steel slabs. Tang et al. (2012) also work on similar relocation problems in steel plants. Casey and Kozan (2012) focus on a variant of the DCRP where the total processing time of the straddle carrier serving a single yard-bay is minimized. Borjian et al. (2015) work on a variant of the DCRP considering a class of exible service policies that permit minor changes in the order of container retrievals. Recently, Zhang et al. (2016) study a connected problem with the CRP where blocks (or containers) can be relocated and/or retrieved in batches. This problem arises in steel plants to remove multiple steel slabs which can be handled by special cranes.

3. CONTAINER RELOCATION PROBLEM

In this section definition of CRP and two formulation for CRP is presented. Next, a index based heuristic developed for CRP.

3.1 Container Relocation Problem Definition

Container Relocation Problem (CRP) which aims to minimize the total number of relocations accomplished to empty out a single yard-bay with a capacity of *C* columns (stacks) having a height of *P* rows (tiers) where *S* containers, whose retrieval sequences is given a priori, are initially located within the yard-bay. The CRP is a static problem in the sense that it only considers departure of containers from the yard-bay. The CRP has the following assumptions introduced by Kim and Hong (2006):

Assumption (A1): The yard-bay is served by a single yard crane which can handle one container at a time
Assumption (A2): The retrieval sequence of containers are known a priori
Assumption (A3): Relocations can only occur to take out a retrieval container from the yard-bay.
Assumption (A4): Relocations can only be made among the columns of the yard-bay
Assumption (A5): Containers have the same type in the yard-bay.

3.2 Container Relocation Problem Formulations

Our formulations, which satisfy assumptions A1-A5, are based on the one developed by Wan et al. (2009) and. We use the original formulation by Wan et al. (2009) and named it as WLT-I.

3.2.1 WLT-II Formulation

The notation employed for the CRP formulations is as follows. Let S be the number of time stages where S containers are retrieved with an order of time stages $s = 1,\ldots, S$. Here, stage s also refers to the retrieval rank of container s from the yard bay. c presents the number of columns for $c = 1,\ldots,C$ and p for the number of rows for $p = 1,\ldots, P$ in a yard-bay. To elaborate better, an example of the CRP is illustrated with figure 3.1 The yard-bay consists of $C = 3$ columns and $P = 3$ rows and contains $S = 6$ containers at the initial stage. Each container is numbered according to the order of their retrieval from the yard-bay. In the first stage (i.e., $s = 1$), the retrieval container 1 should be taken out from the yard-bay. In the second stage (i.e., $s = 2$), the retrieval container 2 should be removed from the yard-bay, and so on.

The container number of a retrieval container (e.g., $i = 1$) is equal to the stage number (e.g., $s = 1$). To satisfy this, two reshuffling containers 3 and 6, which block container 1, needs to relocate. These are indicated above the initial stage of the yard-bay. In the second stage ($s = 2$), the retrieval container 2 departs from the yard-bay. Two reshuffling containers 6 and 5 are placed to other columns. As the remaining yard-bay configuration does not have any blocking container, no relocation is required until the retrieval of container 6 at stage $s = 6$. Let $S(i)$ stands for the smallest ranked container to be retrieved under a container i including container i in the initial yard-bay configuration. Then, for containers 6, 3 and 1 in the first column $S(i)$ values are $S(6) = 1$, $S(3) = 1$ and $S(1) = 1$, respectively.

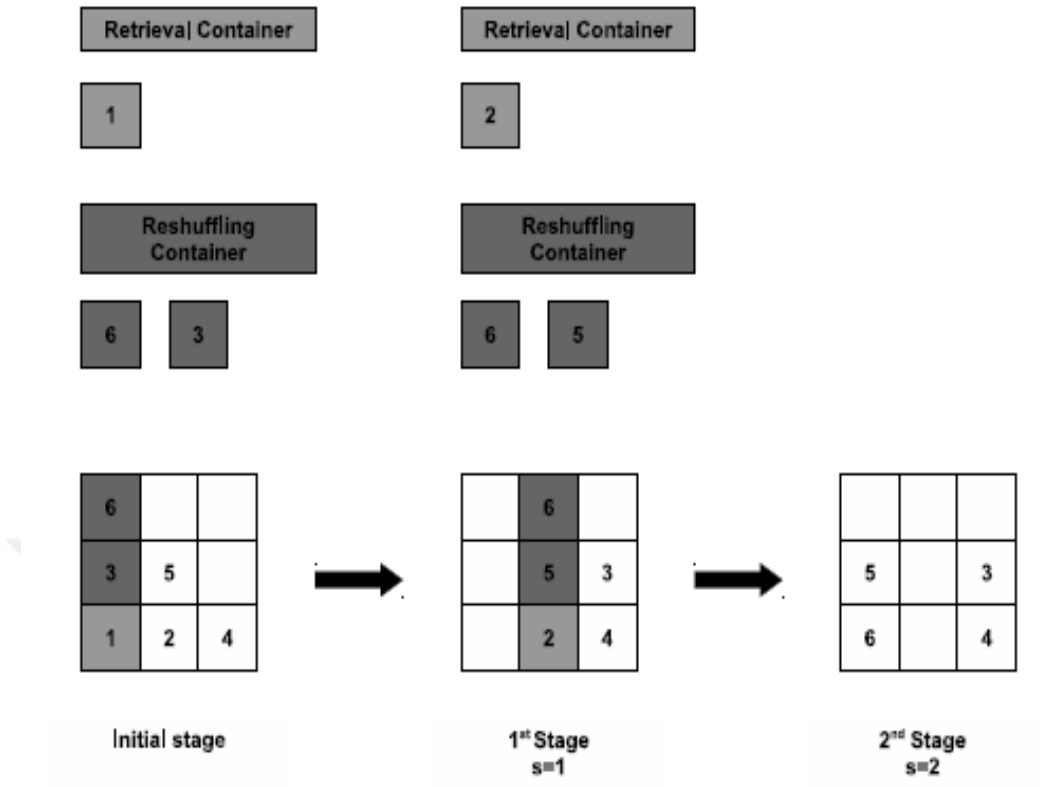

Figure 3.1: Illustrative example for the CRP

Binary variables x_{sicp} are set to 1 if and only if container *i* is located in row *p* of column *c* at stage *s* and zero otherwise. Binary variables z_{si} are equal to 1 if and only if at stage *s* container *i* is situated in a different column than the retrieval container *s*. y_{si} denotes the binary variables taking a value of 1 if and only if container *i* is relocated at stage *s*. w_{sij} states the binary variables having a value of 1 if and only if container *i* and container *j* are relocated, and container *j* is at a higher position than container *i* at stage *s*. Then, our CRP formulation called as WLT-II is as follows.

WLT-II:

$$z^*_{WLT-II} = \sum_{s=1}^{S-1} \sum_{i=s+1}^{S} y_{si} \quad (1)$$

s.t.

$$-z_{si} \leq \left(\sum_{c=1}^{C} \sum_{p=1}^{P} cx_{sicp} - \sum_{c=1}^{C} \sum_{p=1}^{P} cx_{sscp} \right) / C \qquad 1 \leq s < i \leq S \ (2)$$

$$z_{si} \geq \left(\sum_{c=1}^{C} \sum_{p=1}^{P} cx_{sicp} - \sum_{c=1}^{C} \sum_{p=1}^{P} cx_{sscp} \right) / C \qquad 1 \leq s < i \leq S \ (3)$$

$$y_{si} \geq -z_{si} + \left(\sum_{c=1}^{C} \sum_{p=1}^{P} cx_{sicp} - \sum_{c=1}^{C} \sum_{p=1}^{P} cx_{sscp} \right) / P \qquad 1 \leq s < i \leq S \ (4)$$

$$y_{si} \geq -z_{si} + 1 \qquad 1 \leq s < i \leq S (5)$$

$$\left(\sum_{c=1}^{C} \sum_{p=1}^{P} px_{sscp} - \sum_{c=1}^{C} \sum_{p=1}^{P} px_{sicp} \right) / P \leq 1 - y_{si} \qquad 1 \leq s < i \leq S \ (6)$$

$$\sum_{c=1}^{C} \sum_{p=1}^{P} x_{sicp} = 1 \qquad 1 \leq s \leq i \leq S \ (7)$$

$$\sum_{i=s}^{S} x_{sicp} \leq 1 \qquad 1 \leq s \leq S, 1 \leq c \leq C, 1 \leq p \leq P \ (8)$$

$$\sum_{i=s}^{S} x_{sicp} \leq \sum_{i=s}^{S} x_{sic,p-1} \qquad 1 \leq s \leq S, 1 \leq c \leq C, 1 \leq p \leq P \ (9)$$

$$\sum_{p=1}^{P} x_{s+1,icp} \leq 2 - y_{si} - \sum_{p=1}^{P} x_{sscp} \qquad 1 \leq s < i \leq S, 1 \leq c \leq C \ (10)$$

$$2 - y_{si} - y_{sj} - w_{sij} \geq \left(\sum_{c=1}^{C} \sum_{p=1}^{P} p x_{sjcp} - \sum_{c=1}^{C} \sum_{p=1}^{P} p x_{sicp} \right) / P$$

$$1 \leq s < i \leq S, 1 \leq s < j \leq S, i \neq j \ (11)$$

$$y_{si} + y_{sj} + w_{sij} \leq \left(\sum_{c=1}^{C} \sum_{p=1}^{P} p x_{sjcp} - \sum_{c=1}^{C} \sum_{p=1}^{P} p x_{sicp} \right) / P$$

$$1 \leq s < i \leq S, 1 \leq s < j \leq S, i \neq j \ (12)$$

$$w_{sij} \leq y_{si} \qquad 1 \leq s < i \leq S, 1 \leq s < j \leq S, i \neq j \ (13)$$

$$w_{sij} \leq y_{sj} \qquad 1 \leq s < i \leq S, 1 \leq s < j \leq S, i \neq j \ (14)$$

$$\sum_{p=1}^{P} p \left(x_{s+1,icp} - x_{s+1,jcp} \right)$$

$$\geq -P \left[\left(1 - w_{sij}\right) + \left(1 - y_{si}\right) + \left(1 - y_{sj}\right) + \left(1 - \sum_{p=1}^{P} x_{s+1,icp} \right) \right]$$

$$1 \leq s < i \leq S, 1 \leq s < j \leq S, i \neq j \ (15)$$

$$x_{s+1,icp} - x_{sicp} \geq -y_{si} \qquad 1 \leq s < i \leq S, 1 \ \leq c \leq C, 1 \leq p \leq P \ (16)$$

$$x_{sicp} - x_{s+1,icp} \geq -y_{si} \qquad 1 \leq s < i \leq S, 1 \ \leq c \leq C, 1 \leq p \leq P \ (17)$$

$$x_{1icp} = X_{icp} \qquad 1 \le i \le S, 1 \le c \le C, 1 \le p \le P \ (18)$$

$$x_{sicp} = X_{icp} \qquad 2 \le s \le S(i), 1 \le i \le S, 1 \le c \le C, 1 \le p \le P \ (19)$$

$$y_{si}, z_{si} \in \{0,1\} \qquad 1 \le s < i \le S (20)$$

$$w_{sij} \in \{0,1\} \qquad 1 \le s < i \le S, 1 \le s < j \le S, i \ne j (21)$$

$$x_{si} \in \{0,1\} \qquad 1 \le s \le S, 1 \le c \le C, 1 \le p \le P (22)$$

The objective function (1) minimizes of the total number of the relocations performed until the last container leaves the yard-bay. Constraints (2) and (3) determine the value of binary variables z_{si}. Constraint (2) enforces z_{si} to be equal to 1 if containers *s* and *i* are in different columns at stage *s* (i.e., in the retrieval stage of container *s*). For the case, namely, when containers *s* and *i* are in the same column at stage *s*, z_{si} is set to 0. Constraints (4) and (5) impose that a container *i* is relocated at stage *s*, i.e., y_{si}= 1, when the retrieval container *s* and container *i* are in the same column and container *i* stands at a higher position than container *s* which leaves the yard-bay earlier than container *i*, i.e., $s < i$. Constraint (6) makes sure that y_{si}= 0 when retrieval container *s* is situated above container *i* at stage *s*. Note that, in that case, constraints (4) and (5) become redundant. Constraint (7) states that container *i* occupies exactly one slot at stage *s* for $1 \le s < i \le S$. Constraint (8) implies that a slot of the yard-bay can contain at most one container among the containers remaining in the yard-bay at each stage *s*. Constraint (9) ensures that given a column *c* to locate a container in row *p* there a must be a container underneath, that is in row $p - 1$. At the retrieval of container *s* from its column *c*, if there exists another container *i* in the same column that has to be relocated, then constraint (10) guarantees that container *i* is not placed in column *c* again. Constraints (11) and (12) are used to define the value of binary variable w_{sij}. Constraint (11) states that w_{sij} takes a value of 1 when containers *i* and *j* are relocated at stage *s* and container *i* is at a lower slot than container *j*. When container *j* stays at a lower slot than container *i* constraint (12) implies w_{sij} = 0. Constraints (13) and (14) give the dependency of w_{sij} on relocation

variables y_{si} and y_{sj} of containers *i* and *j*. w_{sij} = 0 when either container *i* or *j* is not relocated at stage *s* and constraints (11) and (12) become redundant. Assuming two containers *i* and *j* are relocated at stage *s* then new locations of containers *i* and *j* should be decided in the reverse order prior to their locations before relocations. For example, when they are located in the same column both before and after their relocations, the container at a higher row before the relocation should be placed below the lower row container after relocations. Constraint (15) achieves this property. When a container *i* is not relocated at stage *s* constraints (16) and (17) impose that container *i* maintains the same slot at the next stage *s* + 1.

Constraint (18) introduces the initial locations of containers in the yard-bay. Notice that a container *i* preserves its position until stage *s* = *i* as long as it does not block another container which should be retrieved before stage *s* = *i*. Constraint (19) aims to satisfy this property. Constraints (20), (21) and (22) state binary restrictions. Our enhancements on the WLT-I formulation can be summarized as follows. First, in the original WLT-I formulation, there are two extra binary decision variables to keep record of the position of a container *i* with respect to container *s* such that *s* < *i*. One of the binary variables states that container *i* is on the left side of the retrieval container *s* at stage *s*. The other binary variable checks whether container *i* is located on the right side of the retrieval container *s* at stage *s* or not. Notice that, x_{sicp} contains the slot, i.e., the column and row, information for container *i* at stage *s*. Therefore, we replaced the constraints originally labelled as (7)-(11) in the WLT-I formulation with our new constraints (3) and (4). To achieve this, the definition of binary variables z_{si} is reversed as explained in our WLT-II formulation. Shortly, binary variable z_{si} in the WLT-I formulation is substituted with -z_{si} in our WLT-II formulation and its interpretation is modified accordingly. Constraints (5) and (6) are also changed in the WLT-II formulation with respect to our modification on z_{si}. The remaining parts of the WLT-II formulation are the same as in the WLT-I formulation.

3.2.2 WLT-III Formulation

The WLT-II formulation is tightened by adding the following lower and upper bounding constraints

$$\sum_{s=1}^{S-1} \sum_{i=s+1}^{S} y_{si} \leq Z_{UB} \tag{23}$$

$$Z_{LB} \leq \sum_{s=1}^{S-1} \sum_{i=s+1}^{S} y_{si} \tag{24}$$

where Z_{LB} and Z_{UB} are lower and upper bounds on the optimal objective function value Z_{WLTII}. To calculate lower bound Z_{LB}, we benefit from the following definition. Container i is named as a "singly-blocking" container when it is located above of an earlier departing container within the same column. Clearly, the total number of singly-blocking containers constitutes a lower bound on Z_{WLTII} . Besides, let i be a singly-blocking container that will be relocated at stage s from its column, say column c, in the initial yard-bay configuration. When there exists a container j which will be retrieved between time stages s and i, e.g., $s < j < i$, for all remaining columns $c' = 1, \ldots, C$ and $c' \neq c$, then container i will continue to be a blocking container. This implies that container I will increase the number of relocations by one for a second time. Let us define such a container as "doubly-blocking" container. Note that, doubly-blocking containers consist of singly-blocking containers. Now, summing the total number of doubly-blocking containers with total number of singly-blocking containers in the initial yard-bay configuration yields Z_{LB}, that is used in constraints (24). The calculation of the Z_{LB}, is illustrated with Figure 3.2. for an initial yard-bay of 3 columns and 4 rows. Containers 2, 6 and 7 are singly-blocking containers at the initial yard-bay configuration. At the retrieval of container 1, container 7 is relocated to column $c = 2$ or $c = 3$. However, container 7 will continue to be a blocking container in both columns since it leaves the yard-bay last. Therefore, container 7 is a doubly-blocking container. Consequently, the

sum of singly-blocking containers and doubly-blocking containers yields the lower bound as Z_{LB}, = 3 + 1 = 4.

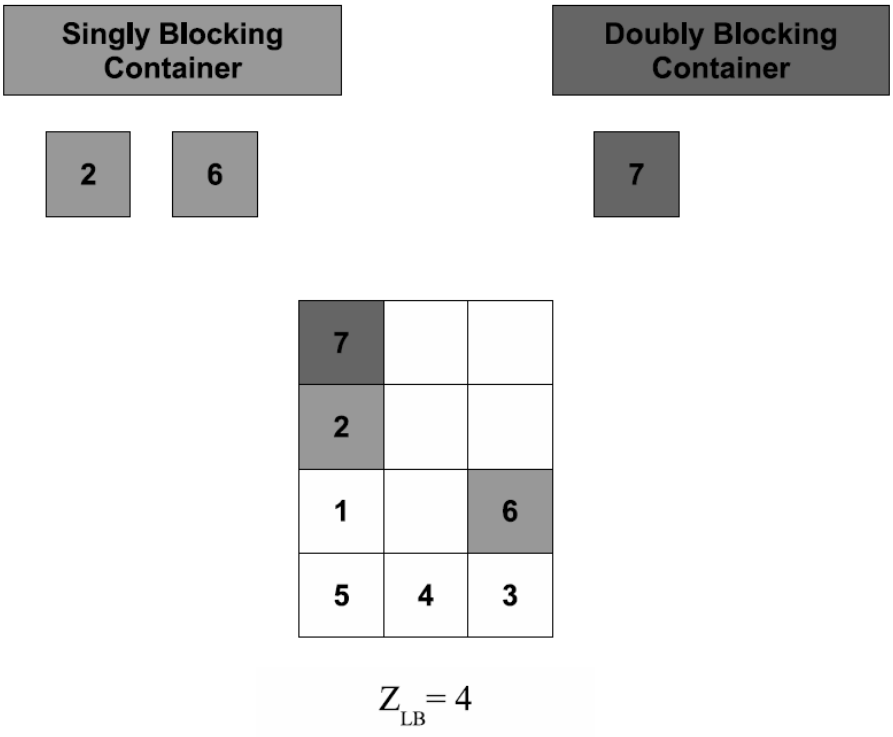

Figure 3.2: Illustrative example for Z_{LB} calculation

A heuristic procedure initially proposed by Murty et al for determine upper bound Z_{UB}. (2005) based on the Reshuffling Index (RI) of the columns of the yard-bay. We call their heuristic as the RI heuristic in the sequel. RI of a column is show us the number of singly-blocking containers at that column. Next, RI heuristic selects the column with the lowest RI to relocate a blocking container. In case of a tie, the column with largest number of containers (i.e., the highest column) is chosen among the columns with lowest RI. When a further tie arises, then the column is arbitrarily selected from qualifying columns. The RI heuristic is very efficient and we employ the best outcome of 100 runs as the Z_{UB} value in constraint (23). Lastly, to make a fair comparison with WLT-I and WLT-II formulations, we separately introduce our second CRP formulation named as WLT-III. We present the WLT-III formulation as follows.

WLT-III:

$$z^{*}_{WLT-III} = \sum_{s=1}^{S-1} \sum_{i=s+1}^{S} y_{si} \quad (25)$$

s.t. (2)-(24) 26)

3.3 A New Heuristic Approach for Container Relocation Problem

We designee Path Index (PI) heuristic by inspiring from possible shortest paths for a container before its retrieval. Container path stands for a stage s and all paths resets at the end of stage s when target container taken out. In this path, each path step is indicated by a container retrieval of the earliest container in column c. For example in Figure 3.3 if container *i=22* is reshuffling container we start our path by container *i=1*. Next, if there is a possibility of container *i=22* makes a movement to column c=1; path of container *i=22* is updated as $path_{i=22,\ c=1}$=1, 13. If we continue reshuffle container *i=22* and relocate it above container *i=14* then $path_{i=22,\ c=1}$= (1, 13, 14). Finally we will move our reshuffling container to empty or above a earliest container which has lower priority than container *i=22.* That means our path become $path_{i=22,\ c=1}$= (1, 13, 14, 0). At the end of the movements we made three relocation for $path_{i=22,\ c=1}$= (1, 13, 14, 0).

Path index is calculated one by one for each blocking container.

0	12	22	6	0	0
0	7	8	19	21	24
0	20	10	18	15	23
0	9	17	2	5	16
13	4	1	11	14	3

Figure 3.3 Initial yard-bay configuration for PI

Calculation of path index is given below in Algorithm 3.1
For each blocking container *i* do the steps which presented below.

Step 0: (Initialization) Set $pathindex_{ics}$=0 for *i*= 1,…,*S* , *c*=1,…,*C* and *s*=1,…,*S*

Step 1: Set $pathindex_{ics}$=M for blocking container *i* which is above target container *s* at column *c*. For the rest of the columns update $path_{ics,k=0} = path_{ics,k=0}$, target container.

Step 2: Define the earliest leaving container *i** in these columns and add the *i** in to the path for each column update $path_{ics,k+1} = path_{ics,k=0}$, *i**. If columns are empty or have lower priority than *i*, update $path_{ics,k+1} = path_{ics,k=0}$, 0.

Step 4: If there is any $path_{icsk}$ which has 0 as last element then stop and update, $pathindex_{ic}$=the number of elements in $path_{icsk}$ for all paths which has 0 as last element choose that column *c* for relocate container *i* or there is multiple paths fulfills this requirement, randomly select one of the paths for relocate the blocking container. Relocate the blocking container then stop the algorithm. Otherwise continue to Step 5.

Step 5: If the *i** in a path not equal 0 then write a yard-bay configuration $\varphi_{i^*}^{path_{icsk}}$ (Figure 3.4) by removing earliest containers in columns which has lower priority than *i** the in path and you should also remove containers above the earliest removed container in columns.

0					
0					
0					
0					
13				14	

Figure 3.4 Configuration image after removal for PI

Step 6: Create new $path_{icsk}$ and add the earliest leaving containers *i** to the path for all available columns. If these columns are not empty or don't have lower priority than *i*, add the *i** to the path for each column $path_{icsk(new)} = path_{icsk(old)}$, *i** and go to Step 4. Otherwise update $path_{ics,k(new)} = path_{ics,k(old)}$, 0. Go to Step 4.

Yard-bay configuration after taken out of target container s=1 is given in Figure 3.5 Blocking containers are 22,8,10 and 17 respectively for stage s=1. Path indexes of each column for container *i*=22, calculated as 2,M,M,M,2,2, respectively. Blocking container is relocated to the column which has the smallest Path index. In case of a further tie, the container is randomly assigned to a column.

0	12	0	6	22	17
0	7	0	19	21	24
10	20	0	18	15	23
8	9	0	2	5	16
13	4	0	11	14	3

Figure 3.5: Configuration after retrieval of target container for PI

4. DYNAMIC CONTAINER RELOCATION PROBLEM

4.1 Dynamic Container Relocation Problem Definition

The DCRP is a dynamic problem in a sense that containers both depart and arrive at the yard-bay. The difficulty of solving the DCRP can be seen better when a new container arrives at the yard-bay. Observe that, the DCRP reduces to solving a CRP as long as containers depart from the yard-bay until the next container arrival. However, an incoming container is likely to change the retrieval sequence of the existing containers within the yard-bay. Therefore, incoming containers change plans repetitively at each arrival. Now, not only the relocation of the containers but also finding the best location for incoming containers gains importance in order to increase the efficiency of yard cranes in yard-bay planning

In this study, we employ the following assumptions used by Akyüz and Lee (2014). Assumptions A1,A3,A4 and A5 are the same for both problems CRP and DCRP. The assumption A2 of the CRP is replaced with the following assumption for the DCRP:

- Assumption (A2'): The arrival and retrieval sequences of containers are known a priori.

4.2 Dynamic Container Relocation Problem Formulation

In the CRP formulations (WLT-I, WLT-II, WLT-III) each stage *s* stands for the retrieval of container *s*. We preserve a similar conditions for the DCRP formulation and match the arrival sequence of containers with the retrieval of container *s*. It is assumed that arrival containers located right after the retrieval of container *s*. Notice that, this does not cause any change on generality for the problem. As the arrival and departure order of containers

are known, arrival pattern of containers can be adjusted for any instance accordingly. Let parameter $K(i)$ denote a retrieval container s after which container i arrives at the yard-bay and i starts to occupy a slot at stage s+1. Notice that $K(i)$ also states the latest time stage before container i joins the yard-bay. There can be multiple containers arriving at the yard-bay which have the same $K(i)$ value. They are distinguished with respect to their arrival order which is represented as $O(i)$. For example, if container i arrives before container j after retrieval container s at stage s, then $K(i) = K(j) = s$ and $O(i) < O(j)$. U_{si} is a parameter that is equal to 1 if and only if container i arrives immediately after the retrieval of container s, i.e., $K(i) = s$. A DCRP example is given in figure 4.1. The yard-bay consist of $C = 3$ columns and $P = 3$ rows containing $S = 5$ containers at the initial stage. Each container is numbered in the order of their retrieval from the yard-bay. In the first stage ($s = 1$), the retrieval container 1 should be taken out from the yard-bay. Reshuffling container 3 is placed to column $c = 3$ to enable access to container 1. Then, container 1 is retrieved from the yard-bay. Next, two containers 7 and 6 consecutively arrive to the yard-bay right after the retrieval of container 1 at stage $s = 1$. Here, the arrival order of containers 7 and 6 are $O(7) = 1 < O(6) = 2$ respectively. The remaining stages are similar to the CRP example as there are no new container arrival. Binary variables a_{sij} are equal to 1 if and only if there exist a blocking container i above the retrieval container s and an arrival container j, i.e., $y_{si} = 1$ and $K(j) = s$, respectively. Then, the DCRP formulation is stated as follows.

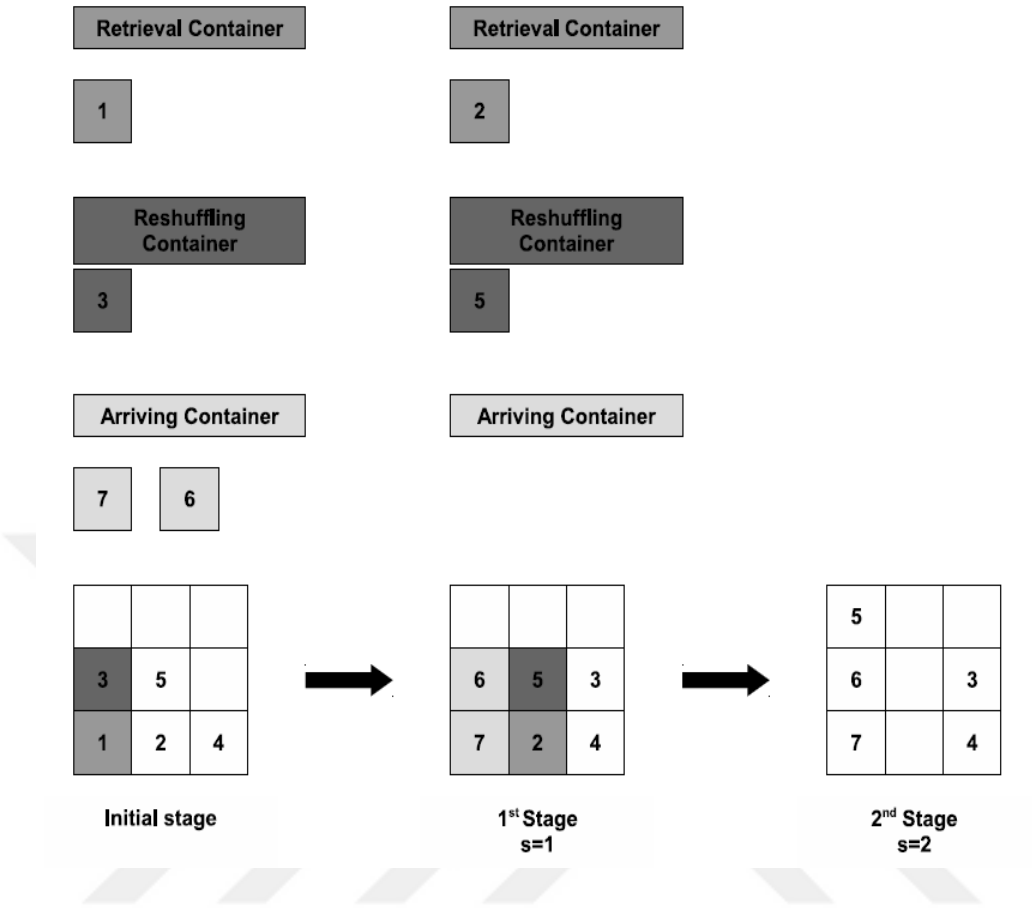

Figure 4.1 DCRP example

WLT-DCRP:

$$z^*_{DCRP} = \sum_{s=1}^{S-1} \sum_{i=s+1}^{S} y_{si} \tag{27}$$

s.t.

$$-z_{si} \leq \left(\sum_{c=1}^{C} \sum_{p=1}^{P} cx_{sicp} - \sum_{c=1}^{C} \sum_{p=1}^{P} cx_{sscp} \right) / C \qquad 1 \leq s < i \leq S, K(i) < s \tag{28}$$

$$z_{si} \geq \left(\sum_{c=1}^{C} \sum_{p=1}^{P} cx_{sicp} - \sum_{c=1}^{C} \sum_{p=1}^{P} cx_{sscp} \right) / C \qquad 1 \leq s < i \leq S, K(i) < s \ (29)$$

$$y_{si} \geq -z_{si} + \left(\sum_{c=1}^{C} \sum_{p=1}^{P} cx_{sicp} - \sum_{c=1}^{C} \sum_{p=1}^{P} cx_{sscp} \right) / P \qquad 1 \leq s < i \leq S, K(i) < s \ (30)$$

$$y_{si} \geq -z_{si} + 1 \qquad 1 \leq s < i \leq S, K(i) < s \ (31)$$

$$\left(\sum_{c=1}^{C} \sum_{p=1}^{P} px_{sscp} - \sum_{c=1}^{C} \sum_{p=1}^{P} px_{sicp} \right) / P \leq 1 - y_{si} \qquad 1 \leq s < i \leq S, K(i) < s \ (32)$$

$$\sum_{c=1}^{C} \sum_{p=1}^{P} x_{sicp} = 1 \qquad 1 \leq s \leq i \leq S, K(i) < s \ (33)$$

$$\sum_{i=s}^{S} x_{sicp} \leq 1 \qquad 1 \leq s \leq S, 1 \leq c \leq C, 1 \leq p \leq P \ (34)$$

$$\sum_{i=s}^{S} x_{sicp} \leq \sum_{i=s}^{S} x_{sic,p-1} \qquad 1 \leq s \leq S, 1 \leq c \leq C, 1 \leq p \leq P (35)$$

$$\sum_{p=1}^{P} x_{s+1,icp} \leq 2 - y_{si} - \sum_{p=1}^{P} x_{sscp} \qquad 1 \leq s < i \leq S, 1 \leq c \leq C \ , K(i) < s \ (36)$$

$$2 - y_{si} - y_{sj} - w_{sij} \geq \left(\sum_{c=1}^{C} \sum_{p=1}^{P} px_{sjcp} - \sum_{c=1}^{C} \sum_{p=1}^{P} px_{sicp} \right) / P$$

$$1 \leq s < i \leq S, 1 \leq s < j \leq S, i \neq j, K(i) < s \ (37)$$

$$y_{si} + y_{sj} + w_{sij} \leq \left(\sum_{c=1}^{C} \sum_{p=1}^{P} px_{sjcp} - \sum_{c=1}^{C} \sum_{p=1}^{P} px_{sicp} \right) / P$$

$$1 \leq s < i \leq S, 1 \leq s < j \leq S, i \neq j, K(i) < s \quad (38)$$

$$w_{sij} \leq y_{si} \qquad 1 \leq s < i \leq S, 1 \leq s < j \leq S, i \neq j, K(i) < s \quad (39)$$

$$w_{sij} \leq y_{sj} \qquad 1 \leq s < i \leq S, 1 \leq s < j \leq S, i \neq j, K(i) < s \quad (40)$$

$$\sum_{p=1}^{P} p\left(x_{s+1,icp} - x_{s+1,jcp}\right) \geq -P\left[\left(1 - w_{sij}\right) + \left(1 - y_{si}\right) + \left(1 - y_{sj}\right) + \left(1 - \sum_{p=1}^{P} x_{s+1,icp}\right)\right]$$

$$1 \leq s < i \leq S, 1 \leq s < j \leq S, i \neq j, K(i) < s \quad (41)$$

$$x_{s+1,icp} - x_{sicp} \geq -y_{si} \qquad 1 \leq s < i \leq S, 1 \leq c \leq C, 1 \leq p \leq P, K(i) < s \quad (42)$$

$$x_{sicp} - x_{s+1,icp} \geq -y_{si} \qquad 1 \leq s < i \leq S, 1 \leq c \leq C, 1 \leq p \leq P, K(i) < s \quad (43)$$

$$x_{1icp} = X_{icp} \qquad 1 \leq i \leq S, 1 \leq c \leq C, 1 \leq p \leq P, K(i) < s \quad (44)$$

$$x_{sicp} = X_{icp} \qquad 2 \leq s \leq S(i), 1 \leq i \leq S, 1 \leq c \leq C, 1 \leq p \leq P, K(i) < s \quad (45)$$

$$y_{si}, z_{si} \in \{0,1\} \qquad 1 \leq s < i \leq S, K(i) < s \quad (46)$$

$$w_{sij} \in \{0,1\} \qquad 1 \leq s < i \leq S, 1 \leq s < j \leq S, i \neq j \quad (47)$$

$$x_{si} \in \{0,1\} \qquad 1 \leq s \leq S, 1 \leq c \leq C, 1 \leq p \leq P \quad (48)$$

$$\sum_{c=1}^{C} \sum_{p=1}^{P} x_{s+1,icp} \geq U_{si} \qquad 1 \leq s < i \leq S \quad (49)$$

$$-1 + y_{si} + U_{si} \leq a_{sij} \qquad 1 \leq s < i \leq S, 1 \leq s < j \leq S, i \neq j \ (50)$$

$$a_{sij} \leq y_{si} \qquad 1 \leq s < i \leq S, 1 \leq s < j \leq S, i \neq j \ (51)$$

$$a_{sij} \leq U_{si} \qquad 1 \leq s < i \leq S, 1 \leq s < j \leq S, i \neq j \ (52)$$

$$\sum_{p=1}^{P} p\, x_{s+1,jcp-} \sum_{p=1}^{P} p\, x_{s+1,icp} \geq -P(1 - a_{sij}) - P(1 - y_{si}) - P(1 - U_{si}) - P\left(1 - \sum_{p=1}^{P} x_{s+1,jcp}\right)$$
$$1 \leq s < i \leq S, 1 \leq s < j \leq S, i \neq j, s = K(j), \quad 1 \leq c \leq C \ (53)$$

$$\sum_{i=s}^{S} \sum_{i=s}^{S} x_{sicp} = 0 \qquad s \leq k(i) \ (54)$$

$$\sum_{p=1}^{P} p\, x_{s+1,jcp-} \sum_{p=1}^{P} p\, x_{s+1,icp} \geq -P(1 - U_{si}) - P(1 - U_{sj}) - P\left(1 - \sum_{p=1}^{P} p\, x_{s+1,jcp}\right)$$
$$1 \leq s < i \leq S, 1 \leq s < j \leq S, O(i) < O(j), i \neq j, s = K(i), s = K(j), 1 \leq c \leq C \ (55)$$

Constraints (34), (35) and (46)-(48) are the same as the CRP formulations given. Constraints (28)-(33) and (36)-(45) are adapted from CRP formulations for the DCRP. However, these constraint sets are restricted for each arriving container, say container *i* that is handled after time stage *s*, i.e., *K*(*i*) < *s*. Constraint (49) ensures that when a container *i* arrives at the yard-bay after retrieval of container *s* at stage *s*, i.e., U_{si} = 1, then arriving container *i* occupies a slot at the next stage *s*+1. Constraints (50), (51) and (52) define the value of the binary variables a_{sij} . Constraint (50) enforces a_{sij} = 1 when there exist a container *i* that will be relocated, i.e., y_{si} = 1, to discharge the retrieval container *s* and a container *j* arrives immediately after retrieval of container *s*, i.e., U_{sj} = 1. Otherwise constraint (50) is redundant. Constraint (51) sets the value of a_{sij} to 0 when there is no blocking container *i* at the retrieval of container *s*. Similarly, constraint (52)

forces a_{sij} to be 0 when there is no arriving container *j* with U_{sj} = 1 at the retrieval of container *s*. Constraint (53) arranges handling order between arrival containers and reshuffling containers in the yard-bay. When a_{sij}= 1 constraint (53) implies that arrival container *j* is taken to the yard-bay after relocation of reshuffling container *i* and retrieval of container *s* at stage *s*. Therefore, arrival container *j* is placed at a higher slot than the relocated container *i* when they are both located in the same column *c* at next stage *s* + 1.

When blocking container *i* and arrival container *j* are not located in the same column *c*, then this constraint becomes redundant. Constraint (54) ensure that arriving containers do not occupy a slot in yard-bay before their arrival. Constraint (55) is the DCRP equivalent of constraints (15) in the CRP formulations to consider container arrivals. Constraint (55) maintains arriving containers are placed in the yard-bay in their arrival order.

There are at most $S(S-1) + S(S+1)CP/2+2S(S-1)(S-2)/3$ binary variables in the DCRP formulation. However, the total number of binary variables are less than this bound in practice. The number of binary variables significantly reduces by taking into account only the ones satisfying $K(i) < s$. Unfortunately, solving the DCRP formulation to optimality can be demanding. Therefore, heuristic methods are more reasonable to obtain solutions for the DCRP instead of optimum solution.

4.3 Heuristic Approaches for Dynamic Container Relocation Problem

4.3.1 Reshuffling Index Heuristic

Reshuffling index (RI) heuristic is initially used by Murty et al. (2005) and adapted for the DCRP by Wan et al. (2009). The RI heuristic is an efficient upper bounding method which is employed in our TS heuristic approaches. RI defines the number of relocations that should be made to clear blocking containers above the earliest outgoing container in a column. For each column the RI is calculated and incoming (or relocating) containers are located on top of the column with the lowest RI. Clearly, RI is calculated among the

columns which are not full. In case of a tie, the container is placed on top of the highest column. In case of a further tie, the container is randomly assigned to a column.

4.3.2 Tabu Search Algorithms

Broadly speaking, TS algorithms move from one solution to another by changing values of one or more decision variables depending on the structure of the problem considered (Gluver and Laguna,1997). Unfortunately, changing the value of a decision variable, which represents locations of containers within a yard-bay at a time-step, affects all subsequent container movements to be made in the DCRP. Hence and efficient algorithm is required to restore the feasibility. To this end we employ the RI heuristic which is tailored for the tabu search. We present a modification to the RI heuristic which is used for that purpose in the following. Next, we give details of the two suggested TS based algorithms, namely TS algorithm-I (TSA1) and TS algorithm-II (TSA2) for the DCRP. In the tabu search based heuristic methods section, we call an arriving (departing) container at the yard-bay as incoming (outgoing) container.

To avoid the randomness of the RI heuristic, the latter tie breaking rule is modified to assigning the container into the leftmost column. The slot assignment of containers in the yard-bay is called as yard-bay configuration. A feasible solution of the DCRP is obtained by keeping track of yard-bay configurations at each time-step. Initially, the RI heuristic is run to obtain a feasible solution. A Tabu List (TL) records the status of columns of the yard-bay for each container. Once a container, denoted by *n*, is assigned to a column4 *c*, in a feasible solution, then column *c* is declared as tabu for container *n* for at least b iterations. Here, b stands for the number of iterations of the tabu duration (tenure). Then, container *n* can not be positioned at the tabu column *c* for at least b iterations. The TS heuristic is run for a total number of *K* iterations. A tabu iteration consists of finding a feasible solution by following the RI heuristic steps described considering the TL which provides diversification of solutions, and hence, the generation of different solutions at each iteration of the TS algorithm. We use two different strategies to declare a column *c* tabu for a container *n*. In the first strategy, a percentage of the incoming containers are randomly selected. In the second strategy, the container that is

relocated after *w* relocations is selected. In the following we present details of these two TS heuristic approaches.

4.3.2.1 Tabu Search Algorithm-I

Now, we present the notation used and a generic algorithm for it. Let $TList_{n,c}$ and $\mathscr{E}$ denote the TL value of container *n* for column *c* and the set of available columns on which container *n* can be pla4ced, respectively. The number of containers that exist in a column *c* is given by N_c. The total number of time-steps *t* is shown with *T*. A formal outline of the TSA1 is given in Algorithm 1.

4.3.2.2 Tabu search algorithm-II

On the other hand, TSA2 employs the number of relocations made since the last tabu declaration. At every w relocations the last relocated container is selected for tabu. Thus, the random parameter α is replaced with a constant parameter *w* in TSA2. TSA2 differs from the TSA1 in two aspects. The first one is the selection of tabu containers. The TSA1 chooses incoming containers to declare a column as tabu while the TSA2 considers the relocating containers for that purpose. The second one is the parameter used for tabu declarations. The TSA1 uses the parameter α to randomly decide if a container is going to be marked as tabu at Step 4 of the Algorithm 1. All other steps of TSA2 are the same as the ones that of the TSA1. TSA-I pseudo code is presented below.

Step 0. (Initialization): Set $TList_{n,c} = 0$ for *n*= 1,…, *N* and *c*= 1,..., *C* and $\mathscr{E}_n^{\alpha} = \varnothing$ for *n*= 1,…, *N*. Set tabu iteration number *k*=1 and time-step *t*=1.

Step 1.For time-step *t* if container *n* is an arriving container then go to Step 2. Otherwise go to Step 6.

Step 2. (Incoming container): Check $TList_{n,c}$ for *c*= 1,…,*C* and $\mathscr{E}_n^{\alpha}$= {c: $TList_{n,c}$ =0, $N_c < H$}.

Step 3. If the cardinality of the set $\mathscr{E}_n^{\alpha}$, $\mathscr{E}_n^{\alpha} > 1$ then use RI heuristic steps to determine the location of container n for columns $c \in \mathscr{E}_n^{\alpha}$. If $|\ \mathscr{E}_n^{\alpha}\ |$ =1, then place container n to column $c \in \mathscr{E}_n^{\alpha}$. Otherwise, select the column with lowest $TList_{n,c}$ among the columns

satisfying $N_c < H$ to locate container n. Set c^* equals the selected column to place container n.

Step 4. Generate a random number $r \in [0,100]$. If $r < \alpha$ then $TList_{nc^*} = \beta + 1$.

Step 5. If $t < T$, set $t = t + 1$ and go to Step 2. Otherwise, go to Step 7.

Step 6. (Outgoing container): If there is no container above the outgoing container n then remove the container n from the yard-bay configuration and go to Step 5. Otherwise for each container n' above the container n at column c^* repeat the following starting from the highest slot and go to Step 5. Check the $TList_{n`c}$ for all c= 1,…,C, $c \notin c^*$ and set $\mathscr{E}_n^\alpha = \{c: TList_{n,c} = 0, N_c < H, c \notin c^*\}$. If the cardinality of the set $\mathscr{E}_{n'}^\alpha$ $|\ \mathscr{E}_{n'}^\alpha| > 1$ then use RI heuristic steps to determine the location of container n' for columns $c \in \mathscr{E}_n^\alpha$. Otherwise, select the column with lowest $TList_{n,c}$ among the columns satisfying $N_c < H$ to locate container n'.

Step 7. If $k<K$ then set $k = k+1$ a feasible solution is found and update the best upper bound accordingly. Set $TList_{n,c}=TList_{n,c} - 1, t = 1$ and go to Step 1. Otherwise, stop and report the best upper bound.

4.3.3 Min-Max Heuristics for the DCRP

The DCRP formulation becomes intractable for large instances. Heuristic procedures are useful to get approximate solutions within reasonable computational times. In what follows, we suggest an efficient heuristic procedure named as Min-Max DCRP (MMD) heuristic. Next, an extension of the MMD heuristic that is originally proposed by Jovanovic and Voβ (2014) for the CRP is described. The Min-Max (MM) algorithm is initially proposed by Caserta et al. (2012) as an efficient heuristic for the CRP. Ünlüyurt and Aydin (2012) present a similar solution procedure for the CRP with different objective functions. Next, an improved MM algorithm is proposed by Jovanovic and Voβ (2014) for the CRP. The MM algorithm tries to avoid from creating new relocations when a reshuffling container will be relocated to retrieve a target container from the yard-bay. To obtain a solution, when there exist a blocking container over a target container, it is relocate in a column depending on container priorities. The priority of a container is defined with the inverse order of retrieval of containers. That is to say, earlier departing containers have higher priority than the containers leaving the yard-bay later. Next, the

priority of a column is determined as the highest priority container in that column. The priority of an empty column is assumed to be of lowest priority. In case, there are multiple columns with a lower priority than the blocking container, then the column having the highest priority is selected among them to locate the blocking container. When there is no column with a lower priority than the blocking container, it is relocated to the column with the lowest priority. This implies that, the blocking container will continue to be a blocking container after its relocation. Hence, by choosing the column with the lowest priority the relocation of the blocking container has been retarded as much as possible until the corresponding highest priority container is retrieved from that column. We benefit from the ideas behind the MM algorithm and suggest, the so called, MMD heuristic for the DCRP. In addition to the retrieval of containers, the DCRP also considers arrival of containers. Hence, we adapt the MM algorithm so that it can also handle arriving containers for the DCRP. The definition of priority of a container stays the same. However, to obtain priority values of containers, their retrieval times should be sorted first since there are arriving containers to the yard-bay. The MMD heuristic works the same as the MM algorithm for the blocking containers as described. Blocking containers has to be reshuffled to one of other columns than its current column. Unlike the blocking containers, an incoming container can be placed in any column without such a restriction. Therefore, priority calculations is performed for all columns. Figure 4.2 gives an example for the MMD heuristic. Now, we introduce some additional notation used in MMD heuristic. Let C^s be the yard-bay configuration at time stage s where container s is retrieved from the yard-bay. The priority of a container i is represented with $p(i)$ and the priority of a column c is show as $p(c)$. P^s stands for the columns that have a lower priority than a selected blocking or arriving container. We present a formal outline of the MMD heuristic in Algorithm 1. Jovanovic and Voβ (2014) suggest an improvement on the MM algorithm that works as follows. Assume that a blocking container is relocated and it will continue to be a blocking container. Now, when the selected column using the MM algorithm for relocation becomes full, then that column misses the chance to host a higher priority (earlier departing) container on its top. The suggested changes avoids putting a blocking container on the highest row P in such a case. We modified this improvement on the MMD heuristic and call the resulting method as MMD-JV heuristic in the sequel.

1. (Initialization) Set time stage $s = 0$ and C^s as the initial configuration of the yard-bay. Sort containers with respect to their retrieval times in ascending order to obtain container priorities *p(i)*. Set upper bound value $Z_{UB} = 0$.
2. Set $s = s + 1$ and $C^s = C^{s-1}$. If container s directly accessible from top of a column, remove s from C^s and go to Step 5.
3. If container s is not directly accessible, let c^s be the column of container s, starting from top to bottom of column c^s for each blocking container I above container s.
   i. Set $Z_{UB} = Z_{UB} + 1$.
   ii. Determine column priorities show as $p^*(c)$ for $c=1,\ldots, C$ and $c \neq c^s$.
   iii. Construct *set* $P^s \neq \{ c : p^*(c) < p(i)$ and $c \neq c^s \}$.
   iv. If $P^s \neq \varnothing$ then select highest priority column c^* as $c^* = argmax_{c \in P*} \{p^*(c)\}$
   v. If $P^s = \varnothing$ then select lowest priority column c^* as $c^* = argmin_{c \in P*} \{p^*(c)\}$
   vi. Locate container i in column c^*. Update C^s accordingly.
4. If there is no arriving container at time stage s go to Step 5. Otherwise in the order of their arrival, for each arriving container I at time stage s.
   i. Determine column priorities show as $p^*(c)$ for $c=1,\ldots, C$.
   ii. Construct *set* $P^s \neq \{ c : p^*(c) < p(i) \}$.
   iii. If $P^s \neq \varnothing$ then select highest priority column c^* as $c^* = argmax_{c \in P*} \{p^*(c)\}$
   iv. If $P^s = \varnothing$ then select lowest priority column c^* as $c^* = argmin_{c \in P*} \{p^*(c)\}$
   v. Locate container i in column c^*. Update C^s accordingly.
5. If $s = S$ stop and report Z_{UB}. Otherwise go to Step 2.

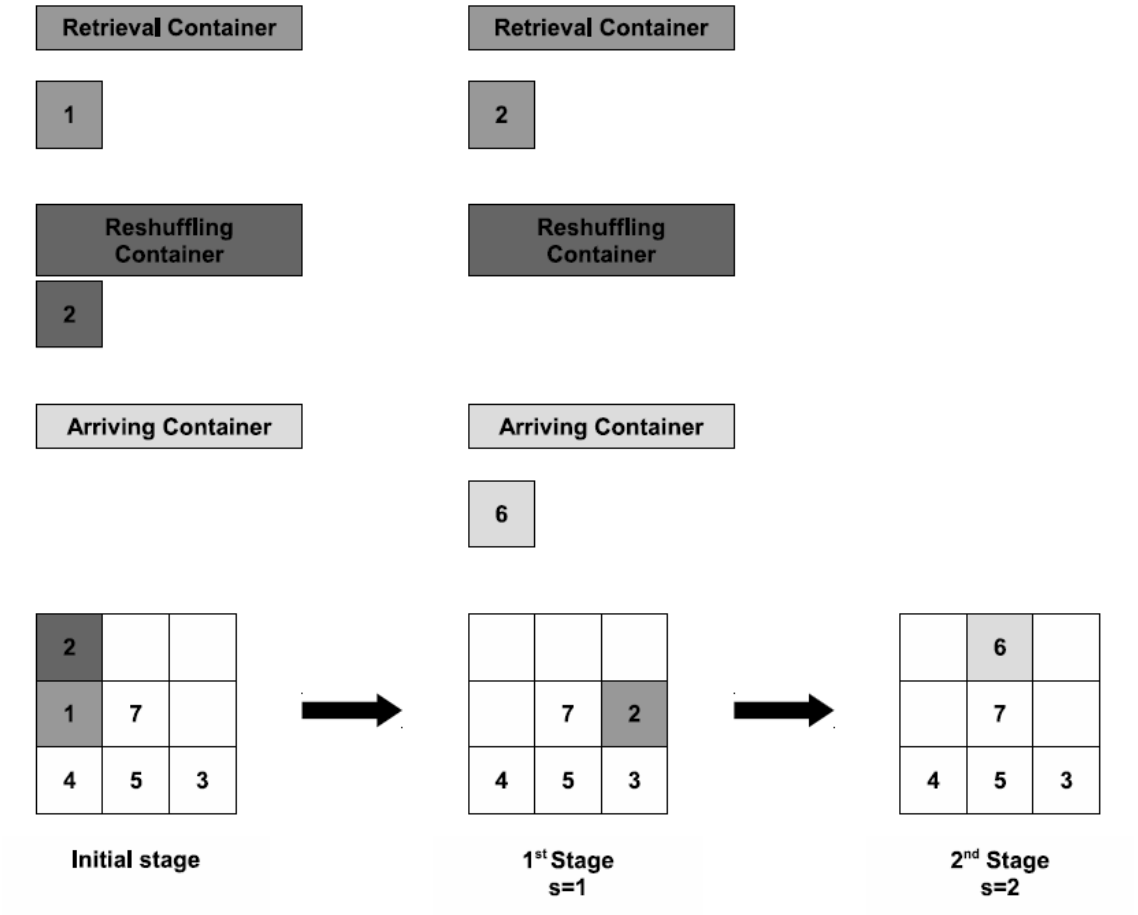

Figure 4.2: Illustrative examples for the MMD heuristic

5. COMPUTATIONAL EXPERIMENTS

In this section, we present our computational experiments for the CRP and DCRP, respectively. The experiments are performed on a computer with a Intel(R) Core(TM) i7-4790 CPU 3.60 GHz and 16 GB RAM operating within Microsoft Windows 10 Pro 64-bit environment. The formulations are solved using Gams 24.1.3 with Gurobi solver and codes are written in *C*++.

5.1 Container Relocation Problem

In this section, we present our computational experiments for the. Then, we show the performance of the CRP formulation on randomly generated test instances. The CRP is tested on a standard test bed given by Caserta et al. (2011) that is employed as benchmark in most (if not all) of the studies offering a CRP formulation. It consists of the first 5 instances randomly generated for 10 different yard-bay size. These test instances are generated so that initially the yard-bay consisting of C columns and H' tiers where the slots are fully loaded by $C*H'$ containers. Then, the maximum height of the corresponding yard-bay is set as $P = H' + 2$ which means the yard-bay contains 2 empty tiers for each instance. Table 5.1 gives a comparison of the CRP formulations. The first and second columns stand for the number of fully loaded rows H' and the number of columns $C{\cdot}H'$ is chosen as 3 and 4 rows, thus, the maximum height P of the yard-bay takes values of 5 and 6 respectively. Total number of columns C of the yard-bay is selected from the set {3, 4, 5, 6, 7, 8}. The third column gives the instance number as shown in the work by Caserta et al. (2012). For each formulation considered, the rows under UB and CPU(s) indicate the total number of relocations and the CPU times in seconds, respectively. BRP-II* states the outcome of the formulation suggested by Expósito-Izquierdo et al. (2015). BRP-II formulation is proposed by Zehendner et al. (2015). The values in columns from 4 to 7 are directly taken from the reference works

mentioned. The columns from 8 to 13 stands for the performance of WLT-I, WLT-II and WLT-III formulations. Note that, WLT-I is the original CRP formulation suggested by Wan et al. (2009). Here, we reproduce their results and test on the standard CRP instances. The average value of 5 test instances belonging to different yard-bay configurations is denoted with bold characters underneath them. For the case that a formulation does not find a solution, the average values are not reported and indicated with *N/A*. Note that, WLT formulations yields an outcome on all test instances, and hence, there is no *N/A* value for them. It is observed that BRP-II*, BRP-II, WLT-I, WLT-II and WLT-III formulations require an overall average of 1681.73, 2498.78, 292.73, 118.36 and 149.16 seconds of running time, respectively. These values are calculated with the instances for which the corresponding formulation yields an outcome. For the WLT-I, WLT-II and WLT-III formulations a CPU time limit of 7200 seconds is imposed. Cells marked with *"-"* indicate that no outcome is produced for that instance by the associated formulation. The cells indicated with *a "*"* imply that the CPU time of 18000 seconds is exceeded by the BRP-II formulation and report the best upper bound found. Outcomes of the test instances with $H' = 4$ and $C = 7$ are not reported for the BRP-II* and BRP-II formulations in the reference works mentioned. As a result, they are shown with empty cells at the bottom of Table 5.1. The WLT-II formulation outperforms other formulations in an overall average of CPU times. A pairwise comparison is performed among the WLT formulations to give a verdict. In Table 5.2 each cell represents the total number of instances for which the formulation stated in the row is superior then the formulation given in the column of the corresponding cell. For example, the number in the first row and third column of Table 5.1 states that WLT-I formulation yields the optimum value in shorter CPU times than the WLT-II formulation on 9 instances. The last column shows the total number of such instances in each row. The WLT-II formulation produce better results than the WLT-I and WLT-III formulations on 37 and 18 instances, respectively. On the other hand, for the WLT-III formulation, these numbers are 36 and 31 instances over the WLT-I and WLT-II formulations. To sum up, the suggested WLT-II and WLT-III formulations generally yields better outcomes than the WLT-I formulation. Besides, the addition of constraints (23)-(24) in the WLT-III formulation mostly enhances the performance of the WLT-II formulation. Hence, we can say that the WLT-III formulation outperforms others in the majority of the test instances.

Table 5.1: A comparison of the performance of the formulations for the CRP on standard test instances.

Instance Info.			BRP-II*a		BRP-IIb		WLT-Ic		WLT-II		WLT-III	
H′	C	No.	UB	CPU(s)	UB	CPU(s)	UB	CPU(s)	UB	CPU(s)	UB	CPU(s)
3	3	1	6	1.18	6	3.85	6	0.06	6	0.06	6	0.08
		2	5	1.39	5	3.75	5	0.05	5	0.05	5	0.03
		3	2	1	2	0.92	2	0.02	2	0.02	2	0.02
		4	4	1.09	4	1.83	4	0.06	4	0.05	4	0.05
		5	1	0.68	1	1.5	1	0.03	1	0.03	1	0.03
Average			**3.6**	**1.07**	**3.6**	**2.37**	**3.6**	**0.04**	**3.6**	**0.04**	**3.6**	**0.04**
3	4	1	5	4.76	5	5.61	5	0.13	5	0.13	5	0.11
		2	3	18.39	3	5.71	3	0.19	3	0.16	3	0.09
		3	7	11.71	7	39.05	7	0.19	7	0.2	7	0.17
		4	5	16.06	5	18.01	5	0.2	5	0.19	5	0.17
		5	6	18.04	6	29.4	6	0.19	6	0.17	6	0.16
Average			**5.2**	**13.79**	**5.2**	**19.56**	**5.2**	**0.18**	**5.2**	**0.17**	**5.2**	**0.14**
3	5	1	6	83.09	6	3323.41	6	0.59	6	0.33	6	0.34
		2	7	75.95	7	3115.28	7	0.48	7	0.39	7	0.27
		3	8	100.71	8	2447.87	8	0.44	8	0.47	8	0.49
		4	6	95.31	6	559.52	6	0.55	6	0.55	6	0.59
		5	9	65.32	9	1418.47	9	0.92	9	0.77	9	0.42
Average			**7.2**	**84.08**	**7.2**	**2172.91**	**7.2**	**0.6**	**7.2**	**0.5**	**7.2**	**0.42**
3	6	1	11	124.11	11	10858.51	11	100.73	11	61.58	11	8.06
		2	7	113.29	7	370.5	7	1.02	7	1.06	7	0.88
		3	11	89.06	15	*	11	15.45	11	5.74	11	13.05
		4	7	93.12	7	*	7	1.52	7	1.33	7	1.25
		5	4	96.5	4	73	4	0.48	4	0.41	4	0.38
Average			**8**	**103.22**	**N/A**	**N/A**	**8**	**23.84**	**8**	**14.02**	**8**	**4.72**
3	7	1	7	182.14	7	3396.4	7	1.2	7	1.09	7	1.05
		2	10	284.29	10	10536.5	10	2.39	10	2.53	10	2.5
		3	9	119.2	-	*	9	6.78	9	2.23	9	3.7
		4	8	472.32	8	2560.7	8	2.06	8	2.06	8	1.92
		5	12	277.97	12	15273	12	12.02	12	2.06	12	19.3
Average			**9.2**	**267.18**	**N/A**	**N/A**	**9.2**	**4.89**	**9.2**	**2**	**9.2**	**5.69**

3	8	1	8	84.66	8	11058.7	8	22.66	8	14.39	8	3.3
		2	10	14403.33	-	*	10	4.36	10	12.22	10	6.75
		3	9	8298.85	-	*	9	4.45	9	3.09	9	2.98
		4	10	250.33	-	*	10	21.05	10	7.42	10	3.09
		5	13	6384.65	-	*	13	8.72	13	5.08	13	12.5
Average			**10**	**5884.36**	**N/A**	**N/A**	**10**	**12.25**	**10**	**8.44**	**10**	**5.73**
4	4	1	10	71.96	-	*	10	0.63	10	0.49	10	0.58
		2	10	228.91	10	484	10	0.44	10	0.38	10	0.39
		3	10	71.99	10	1102	10	0.97	10	0.75	10	1.03
		4	7	65.25	7	90.77	7	0.42	7	0.36	7	0.34
		5	9	89.36	9	5544.12	9	0.55	9	0.5	9	0.49
Average			**9.2**	**105.49**	**N/A**	**N/A**	**9.2**	**0.6**	**9.2**	**0.49**	**9.2**	**0.57**
4	5	1	16	3544.86	-	*	16	90.47	16	81.23	16	199.77
		2	10	326.54	10	2256.4	10	1.73	10	1.56	10	2.7
		3	13	1023.98	-	*	13	10.78	13	15.38	13	8.61
		4	8	119.86	8	1493.2	8	0.83	8	0.8	8	0.63
		5	16	2656.67	17	1213.2	16	821.5	16	286.38	16	1680.09
Average			**12.6**	**1534.38**	**N/A**	**N/A**	**12.6**	**185.06**	**12.6**	**77.07**	**12.6**	**378.36**
4	6	1	17	4077.08	-	*	17	5814.58	17	1842.52	17	1603.97
		2	8	18985	8	177	8	2.84	8	2.03	8	1.84
		3	13	1706.75	-	*	13	35.08	13	24.27	13	9.08
		4	14	2376.86	-	*	14	16.86	14	8.06	14	145.63
		5	15	8564.2	-	*	15	1160.59	15	21.25	15	39.7
Average			**13.4**	**7141.98**	**N/A**	**N/A**	**13.4**	**1405.99**	**13.4**	**379.62**	**13.4**	**360.04**
4	7	1					17	134.55	17	133.84	17	3314.67
		2					18	3074.91	18	793.94	18	86.47
		3					13	103.94	13	11.05	13	7.52
		4					16	1641.25	16	1049.42	16	240.14
		5					16	1514.36	16	1517.84	16	30.72
Average			**N/A**	**N/A**	**N/A**	**N/A**	**16**	**1293.8**	**16**	**701.22**	**16**	**735.9**

a The results of the BRP-II* formulation are taken from Exposito-Izquierdo et al. (2015)
b The results of the BRP-II formulation are taken from Zehendner et al. (2015)
c The original CRP formulation proposed by Wan et al. (2009)

Table 5.2: Comparison of WLT formulations for the CRP.

	WLT-I	WLT-II	WLT-III	Total
WLT-I	-	9	13	22
WLT-II	37	-	18	55
WLT-III	36	31	-	67

In this following, we present our results obtained by using PI heuristic for the CRP. The CRP is tested on a standard test bed given by Caserta et al. (2011) that is employed as benchmark in most of the studies offering a CRP formulation. It consists of the first 5 instances randomly generated for 9 different yard-bay size. These test instances are generated so that initially the yard-bay consisting of C columns and H' tiers where the slots are fully loaded by $C \cdot H'$ containers. Then, the maximum height of the corresponding yard-bay is set as $P = H' + 2$ which means the yard-bay contains 2 empty tiers for each instance. Table 5.3 gives a comparison of the CRP heuristics RI and PI. The first and second columns stand for the number of fully loaded rows H' and the number of columns C. H' is chosen as 3 and 4 rows, thus, the maximum height P of the yard-bay takes values of 5 and 6 respectively. Total number of columns C of the yard-bay is selected from the set {3, 4, 5, 6, 7, 8}. The third column gives the instance number as shown in the work by Caserta et al. 2012). For each heuristic considered, the rows under UB indicate the total number of relocations, respectively. The columns 4 and 5 show the performance of PI and RI, respectively. Here, we reproduce their results and test on the standard CRP instances.

Table 5.3 A comparison of the CRP heuristics RI and PI

Instances			PI	RI
H'	C	No.	UB	UB
3	3	1	6	6
		2	5	6
		3	2	2
		4	4	4
		5	1	1
3	4	1	5	6
		2	3	3
		3	7	7
		4	5	5
		5	6	6
3	5	1	6	6
		2	7	7
		3	8	9
		4	6	6
		5	9	9
3	6	1	11	14
		2	7	7
		3	11	11
		4	7	7
		5	4	4
3	7	1	7	7
		2	10	10
		3	9	9
		4	8	8
		5	12	12
3	8	1	8	8
		2	10	11
		3	9	9
		4	10	10
		5	13	14
4	4	1	10	11
		2	10	11
		3	10	10
		4	7	7
		5	9	9
4	5	1	16	16
		2	10	11
		3	13	14
		4	8	9
		5	16	18

4	6	1	17	20
		2	8	8
		3	13	15
		4	15	15
		5	15	16

5.2 Dynamic Container Relocation Problem

For the DCRP, we follow a similar strategy with Akyüz and Lee (2014). The instances are randomly generated so that the yard crane movements do not get into a deadlock by trying to move a blocking container above the maximum height *P* of the yard-bay. That is, relocations only occur within the yard-bay. It is assumed that initially there are $S' = C\ (P - 1) = 2$ containers in the yard-bay. Here, the number of arrival containers are limited to a maximum of 30 containers considering the capabilities of the suggested DCRP formulation. In Table 5.4, the first three columns consecutively give the number of columns *C*, the number of rows *P* and the initial number of existing containers S' of the yard-bay. The fourth column states the instance number denoted with "No.". Note that 5 random instances are created for each instance combination. 5 different number of arrival containers are chosen from the set 10, 15, 20, 25, 30. In the columns from 5 to 13, for each number of arrival containers, we present the number of relocations made and the CPU time in seconds under the rows named "UB" and "CPU(s)", respectively. Hence, there are 125 randomly generated test instances in total. The average of 5 test instances is denoted with bold characters under them. Similar to the WLT formulations for the CRP, a CPU time limit of 7200 seconds is imposed for the DCRP formulation on the DCRP instances. The suggested DCRP formulation yields the optimal solution on 109 out of 125 test instances. The values indicated with "^{a}" are the best solutions reported on 7 test instances when computing time exceeds the CPU time limit. Lastly, our DCRP formulation can not produce an outcome on 9 out of 125 test instances within 7200 seconds. The DCRP formulation proposed by Akyüz and Lee (2014) can exactly solve up to 10 time steps where each time step includes either a container arrival or departure. However, the test instances solved with our DCRP formulation contains up to handling of 30 arrivals and 42 retrievals (departure of 12 existing containers and 30 arriving

containers) that it corresponds to a total of 72 time steps defined by Akyüz and Lee (2014). Therefore, our proposed DCRP formulation can solve larger instances then the former formulation suggested by Akyüz and Lee (2014).

Table 5.4: The performance of the DCRP formulation on randomly generated test instances.

Instance Info.				10		15		20		25		30	
C	P	S'	No.	UB	CPU	UB	CPU	UB	CPU	UB	CPU	UB	CPU
6	2	3	1	0	0.13	0	0.33	0	1.55	0	1.97	0	2.31
			2	0	0.08	0	0.22	0	1.25	0	8.2	0	10.59
			3	0	0.13	0	0.28	0	2.56	0	0.36	0	16.67
			4	0	0.13	0	0.16	0	1.16	0	3.06	0	11.74
			5	0	0.13	0	0.3	0	2.63	0	5.95	0	1.27
Average				0	0.12	0	0.26	0	1.83	0	3.91	0	8.52
6	3	6	1	0	0.38	0	1.19	0	1.78	0	21.69	1	321.8
			2	0	0.31	0	0.49	0	4.42	0	5.28	0	96.73
			3	0	0.33	0	0.99	1	488.7	0	9.47	2	6060.98
			4	0	0.27	0	1.27	0	3.61	1	397.95	1	165.73
			5	0	0.28	0	0.77	0	2.16	0	5.24	0	15.11
Average				0	0.31	0	0.94	0.2	100.13	0.2	87.93	0.8	1332.07
6	4	9	1	2	1.36	0	1.13	3	13.91	2	4928.67	1	144.02
			2	0	1.17	1	1.61	1	3.02	0	121.84	1	78.42
			3	1	1.03	0	2.27	1	5.33	1	13.05	8^{α}	7200.38
			4	1	0.66	1	3	1	18.25	1	380.02	1	3058.97
			5	2	1.94	2	6.67	1	2.25	3	35.84	24^{α}	7200.41
Average				1.2	1.23	0.8	2.94	1.4	8.55	1.4	1095.88	7	3536.44
6	5	12	1	1	2	3	7.03	1	13.39	3	58	1	2664.06
			2	3	1.78	2	6.3	3	3139.41	3	2747.52	2	19.61
			3	3	4.39	4	2108.25	2	348.5	2	321.39	2	836.52
			4	3	2.81	3	5.66	4	41.3	1	75.17	16^{α}	7200.47
			5	1	2.53	2	5.45	2	1407.13	3	2040.91	26^{α}	7200.48
Average				2.2	2.7	2.8	426.54	2.4	989.95	2.4	1048.6	9.4	3584.23
6	6	15	1	7	265.36	3	434.36	12^{α}	7200.35	-	7200.28	-	7200.38
			2	7	8.64	11^{α}	7200.28	21^{α}	7200.36	1	379.08	-	7200.31
			3	5	454.08	3	2275.53	5	2249.92	-	7200.28	-	7200.39
			4	4	17.03	4	1655.39	0	48.69	-	7200.24	-	7200.33
			5	7	5.92	-	7200.13	4	6659.98	2	14.28	-	7200.34
Average				6	150.21	N/A	3753.14	8.4	4671.86	N/A	4398.83	N/A	7200.35

We use the test bed given by Akyüz and Lee (2014). There are two groups of instances in the test bed: Group-I and Group-II instances. Each group consists of medium and high density of container traffic at the yard-bay with $C = 6$ columns. The range of height, H, is chosen from the set {2, 3, 4, 5, 6} and the number of containers, N, which departs from the yard-bay, is selected from the set {5, 50, 100, 200, 400, 800}. This makes a total of 60 different combinations for each group of instances. 20 test instances for each combination are randomly generated. Therefore, there are 1200 instances for each group. We refer to the work by Akyüz and Lee (2014) for more details on the test bed. In the following, we report our results obtained by the heuristic methods proposed for the DCRP. In Table 5.5, we summarize the performance of the TSA1 on Group-I and Group-II instances. The number of TS iterations is set to $K = 10000$. The first column indicates the group and the density of the test instances. The second column gives the size of the test instances so that (C, H) stands for the number of columns and rows (height) in the yard-bay, respectively. The tabu duration parameter b is set to be b = 3 after our preliminary experiments. The percentage to declare a column as tabu for an incoming container, denoted with parameter, is calibrated as $\alpha = 2$ and $\alpha = 3$ in the light of our initial experiments. The columns "UB" and "CPU" indicate the total number of relocations and the CPU times in seconds, respectively. Each cell gives the average of $20 \cdot 6 = 120$ test instances with different number of containers, N. Columns 3 to 6 include the results when $\alpha = 2$; $\beta = 3$ and $\alpha = 3$; $\beta = 3$. In column 7, we remove the limit on number of tabu iterations and impose a time limit of 4 seconds to run the TSA1 algorithm with the same parameters $\alpha = 2$, $\beta = 3$. The last two columns present the performance of the original RI heuristic whose results are taken from Akyüz and Lee (2014) for comparison. The best outcomes are shown with bold characters for each row. Clearly, the performance of TSA1 increases as the number of TS iterations (or CPU time limit) increases. Observe that the RI heuristic is more efficient than the TSA1. We observe that the suggested TSA1 performs better than the RI heuristic for $H = 2$ and $H = 3$ on all instances. Moreover, for Group-I and Group-II instances with medium density having a height of $H = 4$ the TSA1 yields better outcomes than the RI heuristic. The TSA2 works 22.2% faster than RI heuristic on the average. However, RI heuristic produces better upper bounds than the TSA 2. Therefore, the results of the TSA2 are not reported.

Table 5.5: Summary of the performance of the TSA1 on Group-I and Group-II instances

Instance Group	Size	α=2	β=3	α=3	β=3	α=2 β=3	RI	
	(C,H)	UB	CPU	UB	CPU	UB (4 s.)	UB	CPU
Group-I medium	(6,2)	**0.04**	1.19	**0.04**	0.92	**0.04**	0.11	0.08
	(6,3)	2.24	1.41	2.3	1.16	**2.2**	3.37	0.08
	(6,4)	24.68	1.72	24.9	1.66	**24.37**	25.41	0.08
	(6,5)	63.03	2.22	64.03	2.34	62.33	**57.6**	0.08
	(6,6)	113.86	2.71	115.16	2.94	113.55	**98.84**	0.09
Group-I high	(6,2)	25.41	1.2	25.96	0.93	**25.05**	30.57	0.08
	(6,3)	83.42	1.72	83	1.42	**82.16**	86.3	0.09
	(6,4)	144.05	2.31	144.15	2.2	143.19	**138.65**	0.09
	(6,5)	221.84	3.13	222.48	3.09	221.38	**211.95**	0.1
	(6,6)	293.23	3.68	293.64	3.98	292.51	**276.32**	0.11
Group-II medium	(6,2)	4.91	0.87	4.92	1.02	**4.85**	5.24	0.07
	(6,3)	44.67	1.29	44.83	1.36	**44.45**	45.88	0.08
	(6,4)	92.41	2.12	92.35	1.76	**91.62**	91.63	0.08
	(6,5)	121.82	2.57	122.24	2.16	122.08	**117.45**	0.09
	(6,6)	171.92	3.28	171.8	2.7	171.22	**158.92**	0.1
Group-II high	(6,2)	71.24	1.06	71.68	1.27	**70.81**	76.48	0.08
	(6,3)	139.43	1.65	139.6	1.8	**139.23**	139.89	0.09
	(6,4)	200.71	2.5	201.55	2.17	200.16	**188.72**	0.1
	(6,5)	278.15	3.46	278.81	2.85	278.13	**247.38**	0.11
	(6,6)	368.43	4.14	369.25	3.67	368.43	**315.57**	0.13

The results of the MMD heuristic is obtained on the so called "Group-I" test instances proposed by Akyüz and Lee (2014). In Table 5.6, the performance of the MMD heuristic is summarized on the test instances. The first column indicates the density of the test instances: medium and high. The second and third column gives the size of the test instances so that *C* and *H* are the number of columns and rows (height) in the yard-bay. The results of the heuristic procedures indicate the number of relocations. Column 4 shows the best Index Based (IB) heuristic result reported by Akyüz and Lee (2014). As a remark, IB heuristics use some rule of thumb to give weights to columns of the yard-bay in order to decide the location of a container in the yard-bay. Columns 5 and 6 contain the results of the MMD and MMD-JV heuristics. Last column states the best outcome of the BS heuristic reported by Akyüz and Lee (2014). Each cell from columns 4 to 7 of

Table 5.6 gives the average of 120 test instances. Further, outcomes of the best performing heuristic method are shown with bold characters. The running times of MMD and MMD-JV heuristics are negligible, and thus, CPU times are not reported here. Nevertheless, the MMD and MMD-JV heuristics are both more efficient than IB and BS heuristics. The MMD heuristic performs better than the IB heuristics on high density instances for all height values H of the yard-bay and on medium density instances for $H \geq 4$. Broadly speaking, the MMD heuristic outperforms the MMD-JV heuristic in yard-bays having smaller height value H. The converse holds in favor of MMD-JV heuristics when H gets larger. In particular, the MMD-JV heuristic is superior than the MMD heuristic on medium and high density instances having $H \geq 4$. A similar result can be drawn between the MMD-JV and BS heuristics. The MMD-JV heuristic gives poor results than the BS heuristics on medium and high density instances with $H \leq 3$ and $H \leq 4$, respectively. On the remaining instances the MMD-JV heuristic performs better than the BS heuristic. Notice that, the CPU time required for the BS heuristic can be prohibitive. Therefore, for yard-bays having a height of $H \geq 5$, the MMD-JV heuristic is a better alternative. The MMD-JV heuristic is also an efficient choice for yard-bays having a height of $H \leq 4$.

Table 5.6: The performance of the heuristic procedures for the DCRP on standard test instances

Instance Info.			IB	MMD	MMD-JV	BS
Density	C	H	Heuristic	Heuristic	Heuristic	Heuristic
Medium	6	2	0.11	0.22	1.24	0.03
		3	3.08	3.13	3.01	1.77
		4	23.82	16.18	15.86	17.14
		5	53.29	35.88	35.51	44.99
		6	95.45	63.11	61.01	82.17
High	6	2	30.16	27.42	47.88	15.59
		3	78.16	70.39	88.73	54.39
		4	133.25	117.17	120.26	103.32
		5	210.64	185.79	176.06	177.7
		6	271.03	239.43	227.03	244.97

6. CONCLUSION

In this work, we address the CRP and its dynamic extension DCRP. The CRP tries to discharge existing containers from a single yard-bay while minimizing total number of container relocations. Unlike the CRP, the DCRP permits new containers to join the yard-bay as well. We propose mathematical programming formulations for the CRP and DCRP. Efficient heuristics are also suggested for the CRP and DCRP. An extensive set of computational experiments is performed on both standard and randomly generated test instances.

WLT-I formulation originally developed by Wan et al. (2009) is modified and two new formulation is proposed for the CRP. WLT-II and WLT-III compared with existing formulations in literature. Our results show that WLT-II and WLT-III formulations yields better outcomes than existing formulations in literature. Next, we propose a new DCRP formulation wich performs better than the former formulation suggested by Akyüz and Lee (2014). In particular, the new DCRP formulation can solve instances having a planning period of up to seven times longer than we can obtain with the existing formulation.

Two TS based heuristic algorithms are proposed for the DCRP. The first algorithm, TSA1, uses a random selection strategy for tabu declarations while the second algorithm TSA2 employs a fixed number of steps for that purpose. According to our computational experiment we observe that the proposed TSA-I is efficient and yields promising outcomes. Next, we develop the PI heuristic which performs better than the RI heuristic for the CRP.

Two efficient heuristic procedures are devised for the DCRP. Basically, they are enhancements of the MM algorithm. The MMD and MMD-JV heuristics are tested on

standard test instances. Our computational experiments state that both MMD and MMD-JV heuristics are very efficient and yield promising outcomes compared to other heuristic procedures from the literature for the DCRP.

Introducing valid inequalities for existing CRP and DCRP formulations and the design of exact solution procedures can be a worthwhile further research. Moreover, unrestricted CRP and DCRP, which relax assumption A3 by allowing container pre-marshalling operations, may be a fruitful research area. In particular, enhanced formulations can be designed for the unrestricted problems as a future work.

REFERENCES

Akyüz M H, Lee C (2014). A mathematical formulation and efficient heuristics for the dynamic container relocation problem., *Naval Research Logistics* 61:101–118

Borjian, S., Manshadi, V. H., Barnhart, C., & Jaillet, P. (2015). Managing Relocation and Delay in Container Terminals with Flexible Service Policies. *arXiv preprint arXiv: 1503.01535* Accessed 06.04.2016.

Caserta, M., Voss, S., & Sniedovich, M. (2011). Applying the corridor method to a blocks relocation problem., *OR Spectrum, 33*, 915-929.

Caserta M, Schwarze S, Voβ S (2012). A mathematical formulation and complexity considerations for the blocks relocation problem., *European Journal of Operational Research* 219:96–104

Casey, B., & Kozan, E. (2012). Optimising container storage processes at multimodal terminals., *Journal of the Operational Research Society, 63*, 1126-1142.

Expósito-Izquierdo, C., Meli_an-Batista, B., & Moreno-Vega, J. M. (2015). An exact approcah for the blocks relocation problem., *Expert Systems with Applications, 42*, 6408-6422.

Forster, F., & Bortfeldt, A. (2012). A tree search procedure for the container relocation problem., *Computers & Operations Research, 39*, 299-309.

Glover F, Laguna M (1997) Tabu search. Kluwer academic publishers, Boston

Jin B, Zhu W, Lim A (2015). Solving the container relocation problem by an improved greedy look-ahead heuristic., *European Journal of Operational Research* 240: 837–847

Jovanovic R, Voss S (2014). A chain heuristic for the blocks relocation problem., *Computers & Industrial Engineering* 75:79–86

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

Kim K H, Hong G P (2006). A heuristic rule for relocating blocks., *Computers & Operations Research* 33:940– 954

Konig, F. G., Lübbecke, M., Möhring, R., Schafer, G., & Spenke, I. (2007). Solutions to real-world instances of PSPACE-complete stacking. In L. Arge, M. Hoffmann, & E. Welzi (Eds.), Algorithms - *ESA 15th Annual European Symposium*, Lecture Notes in Computer Science, , Berlin, Springer. vol. 4698 (pp. 729:749)

Ku, D., & Arthanari, T. S. (2016). On the abstraction method for the container relocation problem., *Computers & Operations Research: 68,* 110-122.

Lee, Y., & Lee, Y.-J. (2010). A heuristic for retrieving containers from a yard., *Computers & Operations Research, 37,* 1139-1147.

Lehnfeld J, Knust S (2014). Loading, unloading and premarshalling of stacks in storage areas: survey and classification., *European Journal of Operational Research:* 239:297–312.

Murty K, Liu J, Wan Y et al (2005) A decision support system for operations in a container terminal., *Decision Support Systems* 39:309-332.

Petering M E H, Hussein M I (2013) A new mixed integer program and extended look-ahead heuristic algorithm for the block relocation problem., *European Journal of Operational Research* 231:120–130.

Rei, R., & Pedroso, J. P. (2013). Tree search for the stacking problem., *Annals of Operations Research 203*: 371-388.

Stahlbock R, Voβ S (2008) Operations research at container terminals: a literature update., *OR Spectrum* 30:1–52.

Steenken, D., Voβ, S., & Stahlbock, R. (2004). Container terminal operation and operations research - a classification and literature review., *OR Spectrum, 26*:3-49.

Tang, L., Zhao, R., & Liu, J. Y. (2012). Models and algorithms for shuffling problems in steel plants., *Naval Research Logistics, 59*: 502-524.

UNCTAD (2014) Review of Maritime Transportation 2014. Paper presented at the United Nations Conference on Trade and Development, New York and Geneva. http://unctad.org/en/PublicationsLibrary/rmt2014 en.pdf. Cited 28 June 2015

Ünlüyurt, T., & Aydın, C. (2012). Improved rehandling strategies for the container retrieval process., *Journal of Advance Transportation, 46*: 378-393.

Wan Y, Liu J, Tsai P (2009) The assignment of storage locations to containers for a container stack., *Naval Research Logistics* 56:699–713.

Zehendner, E., Caserta, M., Feillet, D., Schwarze, S., & Voβ, S. (2015). An improved mathematical formulation for the blocks relocation problem., *European Journal of Operational Research, 245*, 415{422.

Zhang, R., Liu, S., & Kopfer, H. (2016). Tree search procedures for the blocks relocation problem with batch moves. *Flexible Services and Manufacturing Journal*, article in press, 1-28.

BIOGRAPHICAL SKETCH

Osman Karpuzoğlu born on July 28, 1991 in Bursa, Turkey. He studied Tan Fen Private High School where he was graduated in 2009. He started his undergraduate studies at the Industrial Engineerin Department of Galatasaray University in 2009. In 2014, he obtained the B.S degree in Industrial Engineering. Since April 2014, he has been working as a Tubitak project asistant in Galatasaray University. Currently, he is working towards master's degree in Lojistics and Financial Management under supervision of Yrd. Doç. Dr. M. Hakan AKYÜZ at Institute of Science and Engineering, Galatasaray University.