

A CONTAINER STORAGE PROBLEM IN PORT OPERATIONS

BURCU ÇELİK

OCTOBER 2013

A CONTAINER STORAGE PROBLEM IN PORT OPERATIONS

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

BY

BURCU ÇELİK

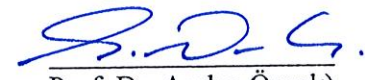
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE
OF
MASTER OF SCIENCE
IN
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

OCTOBER 2013

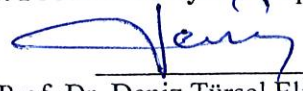
Approval of the Graduate School of Natural and Applied Sciences

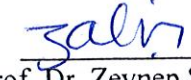

(Prof. Dr. Cüneyt Güzeliş)
Director

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


(Assoc. Prof. Dr. Arslan Örnek)
Head of Department

We have read the thesis entitled **A Container Storage Problem in Port Operations** prepared by **Burcu ÇELİK** under supervision of **Assoc. Prof. Dr. Deniz TÜRSEL ELİİYİ** and **Asst. Prof. Dr. Zeynep SARGUT**, and we hereby agree that it is fully adequate, in scope and in quality, as a thesis for the degree of **Master of Science in Intelligent Production Systems** option of **Intelligent Engineering Systems**.


(Assoc. Prof. Dr. Deniz Türsel Eliiyi)
Advisor


(Asst. Prof. Dr. Zeynep Sargut)
Co-Advisor

Examining Committee Members:
(Chairman, Supervisor and Members)

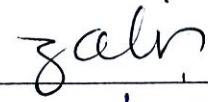
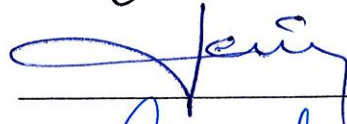
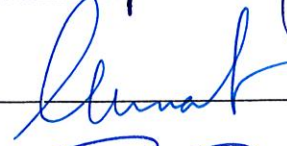
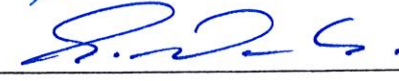

Asst. Prof. Dr. Zeynep SARGUT
Industrial Engineering Dept., IUE

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

Assoc. Prof. Dr. Murat FADİLOĞLU
Industrial Engineering Dept., Yaşar Uni.

Assoc. Prof. Dr. Arslan ÖRNEK
Industrial Engineering Dept., IUE

Asst. Prof. Dr. Erdiç ÖNER
Industrial Engineering Dept., IUE

ABSTRACT

A CONTAINER STORAGE PROBLEM IN PORT OPERATIONS

Çelik, Burcu

M.Sc. in Intelligent Engineering Systems
Graduate School of Natural and Applied Sciences

Advisor: Assoc. Prof. Dr. Deniz Türsel Eliiyi

Co-Advisor: Asst. Prof. Dr. Zeynep Sargut

October 2013, 105 pages

In this study, we consider the container storage problem of a transit container yard at a container terminal. There are various important decisions in container terminal management operations, many of which are interrelated. Container storage problem involves one of these decisions, which has implications on the total transportation cost of containers in the yard. We assume that the berth allocations for the incoming and outgoing vessels are fixed, and the arrival and departure times are predetermined. The objective of the problem is to minimize the total transportation cost of the transit containers from the vessels to storage locations, and from the storage locations to outgoing vessels. Two types of movements are of concern: vertical and horizontal. The vertical transportation cost involves the reshuffling of the containers by cranes; the associated cost is proportional to the number of containers to be removed to reach the target container in a container stack. On the other hand, the horizontal cost is related with ground transportation by trucks or trailers. We propose three mathematical models with differing sets of assumptions,

which reduce this three-dimensional storage problem into a two-dimensional problem. We provide computational results for exact solution of the problem by CPLEX and also consider a Lagrangean relaxation-based approach. Two heuristics are developed and presented to come up with quick handy solutions for especially large problem instances. The results are discussed and analyzed, together with future research directions.

Keywords: Container Storage Problem, Mathematical Modeling, Lagrangean Relaxation, Heuristics.

ÖZ

LİMANLARDA KONTEYNER DEPOLAMA PROBLEMİ

Çelik, Burcu

Akıllı Mühendislik Sistemleri Yüksek Lisans Programı

Fen Bilimleri Enstitüsü

Tez Danışmanı: Doç. Dr. Deniz Türsel Eliiyi

Ortak Tez Danışmanı: Yard. Doç. Dr. Zeynep Sargut

Ekim 2013, 105 sayfa

Bu çalışmada konteyner terminallerindeki transit konteyner depolama problemi ele alınmaktadır. Konteyner terminal yönetim sistemlerinde birbirleriyle ilişkili birçok problem bulunmaktadır. Konteyner depolama problemi liman depolama alanındaki toplam depolama maliyetini etkileyen önemli problemlerden birisidir. Çalışmamızda limana giriş çıkış yapan gemiler için rıhtım yerleştirmesinin bilindiği varsayılmaktadır. Aynı zamanda gemilerin limana varış ve ayrılma zamanları da önceden bilinmektedir. Tezde ele alınan konteyner depolama problemi, rıhtımda depolanan transit konteynerlerin gemiden depolama alanına ve depolama alanından gemiye ulaşımındaki toplam taşıma maliyetini enazlamayı amaçlar. Dikey taşıma maliyeti yeniden elleçleme maliyetini içermektedir. Yeniden elleçme bir istiftteki belirli bir konteynere ulaşmak için yapılan dikey hareketlerdir. Yatay maliyet ise tır veya römork vasıtasıyla yapılan terminal içi taşıma maliyetlerini kapsar. Çalışmada farklı varsayımlar ile bu üç boyutlu depolama problemini iki boyutlu probleme indirgeyecek üç farklı matematiksel model önerilmiştir. Problemlerin optimal çözüm

sonuçları için CPLEX kullanılmıştır. Bu modeller için aynı zamanda Lagrangean gevşetme yöntemine dayalı bir yaklaşım denenmiştir. Yeni geliştirilen sezgisel yaklaşımlar ile problemin çözümüne daha kısa sürede ulaşmak hedeflenmiştir. Sonuçlar ve gelecek çalışma önerileri analiz edilerek tartışılmıştır.

Anahtar Kelimeler: Konteyner Depolama Problemi, Matematiksel Modelleme, Lagrange Gevşetme, Sezgisel Yaklaşımlar.

ACKNOWLEDGMENTS

I would like to express my gratitude to my advisor Assoc. Prof. Dr. Deniz Türsel Eliyi and co-advisor Asst. Prof. Dr. Zeynep Sargut for their useful comments, remarks and engagement through the learning process and writing of this master thesis. Without their guidance and persistent help this master thesis would not have been possible.

During the period of three years, many friends are helpful to color my life. I have to acknowledge all my colleagues in research room A311 for their assistances in many aspects that I cannot list them all because of limited space. Res. Assistant Burak Gökgür, for teaching me everything I know about GAMS, not leaving me alone during my studies. As well as Oktay, my dear friend, supported me during this whole time. Serkan Osman Dibek (SOD) has been our happiness and true friend. My all colleagues have often had to bear the brunt of my frustration and rages against the world with equanimity and friendship. I will never forget their friendship.

I would like to thank my other colleagues, Res. Assistant Ceren ÖCAL, Res. Assistant Zeynep Nihan ODABAŞ and Res. Assistant Nur UYLAŞ for your encouragement and understanding throughout this work.

Last but not the least important, I owe more than thanks to my parents, Nurdan and İbrahim Çelik for their financial support and encouragement throughout my life. I love my sister Burçin Çelik and my brother Emrecaan Çelik. They are wonderful gifts from God. Without my family's support, it is impossible for me to finish my college and graduate education seamlessly. My friend Sevin İbrikçi who I see as a sister supported to me. I also thank everyone who helps me during this period.

TABLE OF CONTENTS

ABSTRACT	ii
ÖZ	v
ACKNOWLEDGMENTS	vii
TABLE OF CONTENTS	viii
LIST OF TABLES	x
LIST OF FIGURES	xi
CHAPTER	
1 INTRODUCTION	1
2 LITERATURE REVIEW	5
3 CONTAINER STORAGE PROBLEM	17
4 CONTAINER STORAGE PROBLEM WITH EXCESS CONTAINERS	31
4.1. Mathematical Model with No Restriction	32
4.2. Mathematical Model with First-Come First-Served Restriction	33
5 SOLUTION METHODS	36
5.1. Lagrangean Relaxation-based Approach	37
5.1.1. Lagrangean Relaxation for Two Constraints	37
5.1.2. Lagrangean Relaxation for Single Constraint	45
5.1.3. Lagrangean Relaxation with Single Lagrangean Multiplier	47

5.2. Heuristics.....	48
6 COMPUTATIONAL RESULTS.....	59
6.1. Experiment Design	59
6.2. Exact Solutions	61
6.3. Computational Results.....	67
6.3.1. Results on Lagrangean Relaxation.....	67
6.3.2. Results of Heuristics.....	87
7 CONCLUSION.....	96
REFERENCES.....	100

LIST OF TABLES

Table 1 Experimental Design.....	60
Table 2 CPLEX solution for CSPX-1	63
Table 3 CPLEX solution for CSPX-2	66
Table 4 Summary of LRTC for CSPX-1, number of iterations = 50.....	70
Table 5 Summary of LRTC for CSPX-1, number of iterations = 100.....	71
Table 6 Summary of LRSC for CSPX-1, number of iterations = 50.....	72
Table 7 Summary of LRSC for CSPX-1, Number of iterations = 100	74
Table 8 Summary of LRSLM for CSPX-1. Number of iterations = 50.....	75
Table 9 Summary of LRSLM for CSPX-1. Number of iterations = 100.....	76
Table 10 Summary of LRTC for CSPX-2. Number of iterations = 50.....	77
Table 11 Summary of LRTC for CSPX-2. Number of iterations =100.....	78
Table 12 Cost Value given by LRTC for CSPX-1. Number of iterations =50	79
Table 13 Cost Value given by LRTC for CSPX-1. Number of iterations =100.....	80
Table 14 Cost Value given by LRTC for CSPX-2. Number of iterations =50	81
Table 15 Cost Value given by LRTC for CSPX-2. Number of iterations = 100.....	82
Table 16 Comparison of LRTC with optimal solution for CSPX-1. Number of iterations =100.....	83
Table 17 Comparison of LRTC with optimal solution for CSPX-2. Number of iterations =100.....	84
Table 18 Cost Value given by H1 heuristics.....	88
Table 19 Cost Value given by H2 heuristics.....	89
Table 20 Comparison of H1 with optimal solution.....	90
Table 21 Comparison of H2 with optimal solution.....	92

LIST OF FIGURES

Figure 1 Related Network Flow Problems.....	11
Figure 2 Decision Problems in Container Terminals.....	18
Figure 3 Berth in line and in the indented type.....	19
Figure 4 Stack in storage location.....	19
Figure 5 Side cross-section of the container a column.....	20
Figure 6 Container Storage Problem.....	22
Figure 7 Time interval.....	23
Figure 8 Storage Area under the “covering assumption”.....	26
Figure 9 Relocate the container.....	50
Figure 10 Swap the containers.....	51
Figure 11 Time interval for sample case.....	51
Figure 12 Placement at each period for small case.....	52
Figure 13 Placement behavior of initial condition.....	56
Figure 14 Number of reshuffles based on the occupancy for CSPX-1.....	64
Figure 15 Number of Reshuffling based on Instances’ Parameters for CSPX-1.....	65
Figure 16 Number of reshuffles for each problem set.....	67
Figure 17 Number of Reshuffles Comparison for LRTC and LRSC.....	73
Figure 18 Running Time Comparison for LRTC and LRSC.....	73
Figure 19 Comparison of Utilization Values for CSPX-1 and CPSX-2.....	78
Figure 20 The behavior of Lower Bound on Each Iteration - LRTC.....	85
Figure 21 Number of reshuffles for LR, H1 and GAMS.....	91
Figure 22 Number of reshuffles for LR, H1, H2 and GAMS.....	93
Figure 23 Transport cost for LR, H1 and H2.....	93
Figure 24 The behavior of bounds of all iteration (LTRC).....	94

CHAPTER - 1

INTRODUCTION

An international container service between the US East Coast and several points in the Caribbean, Central and South America has begun around 1961 as a regular sea container service (Steenken et al., 2004). A container is a standard-sized metal box that can be easily transferred between different modes of transportation, such as ships, trains and trucks. Transportation via containers quickly received considerable investments in special ship designs, suitable equipment and purchase of containers. The economic efficiency and market share started to grow through a large number of container transshipments. Transfer or change from one conveyance to another with a temporarily limited storage on the container yard is described as transshipment in this context.

The twenty-foot equivalent unit (TEU) is a unit that defines the capacity of containers, container ships and container terminals. According to this measure, the volume of a standard twenty-foot-long (6.1 m) container is referred to as 1 TEU, and the volume of a forty-foot-long one is 2 TEU.

The transportation between economically strong and stable countries has been containerized up to 100%, and over 60% of the world's deep-sea general cargo is transferred via containers. According to an international containerization market analysis, in 1995 9.2 million TEU were in circulation. During these 10 years the

container fleet had almost doubled from a size of 4.9 million TEU (Steenken et al., 2004). In addition to, the number of containers transported by vessel is 77.8 million TEU in 2002 and 139 million TEU in 2010. The forecast for the number of containers in 2015 is equal to 177.6 million TEU. The number of handling at ports is much more than the number of containers transported by vessel. The worldwide container traffic is 560 million TEU in 2010 according to container market analyst Alphaliner. (Alphaliner, 2012)

A report by Global Industry Analysts Inc. in 2012 forecasted that world container throughput could attain more than 730 million TEUs by 2017. (Global Industry Analysts, 2012)

Maritime transport is the most preferred form of transport due to the ability to load large amounts at a time, reliability and its minimal cost (about 14 times cheaper than airline transportation, about 7 times cheaper than road, and 3.5 times cheaper than rail) (Eliiyi et al., 2008). According to the reports of World Trade Organization, in 2007, more than 90% in 2006, part of the import and export loads are transported by sea. Also the volume of international trade carried by sea is increasing. (WTO, 2007) For near future, the maritime container throughput posts positive growth patterns with respect to report by Global Industry Analysts in 2012. The growth on containerization backed largely by increasing use of containerization for shipping bulk cargo, use of cutting edge technology such as automated handling system and satellite tracking system for speed and efficiency in operation in the port. The growing awareness of energy efficient and environment friendly products among the shippers and consumers are set to generate more opportunities in the market. That's why incremental advancements in technologies governing alternative fuels, and pollution control systems immensely support the growth of maritime transport. (Global Industry Analysts, 2012)

Due to the high growth of the number of container shipments and maritime transportation, higher demands on the seaport container terminals, management concerns, and also technical equipment requirements have emerged. This development resulted in an increased competition between geographically close

container terminals. It is now crucial to operate container terminals in an efficient and effective manner. In Turkey's State Planning Organization's 9th Development plan (2007-2013) "developing our important ports as logistic centers" was stated as one of the priorities of Turkey, which is in parallel with global trends. Therefore, operating container terminals more efficiently is critical for the near future of our country. In this thesis, we consider the container storage allocation problem, which has applications not just in Turkey but at container terminals all over the world.

The scope of our study is limited to transit containers, which arrive to the container terminal via a vessel, and depart via another. The reason why transit (or transshipment) containers are considered is that, the arrival/departure times of export/import containers, which are transferred through railway or road are seldom known with any certainty. Due to traffic and other external factors, estimating the retrieval time of a container from the terminal by a vehicle (usually truck in our country) is very hard. On the other hand, as transit containers arrive and depart by ships, their times can be estimated more precisely, which simplifies storage decisions. Besides this technical point, according to the report of the Ministry of Transport in year 2009, transit containers comprise about 22% of the total containers handled in Turkey. For example, at Marport, the largest container port in Turkey with 1.6 million TEUs in a year of container capacity, transit containers comprise nearly half of the operational volume. This is mainly due to the fact that ports of Turkey are located on a strategic junction of the Eastern Mediterranean and Black Sea routes and international transport corridors in the East-West and North-South directions. Through this advantageous location, our container terminals can attract transit shipments. However, Turkish ports must reduce operating costs of the terminals to be able to compete in the highly competitive international markets (Gürgeç, 2010). This is another motivation for our study.

Decisions in the container storage problem are interrelated; each storage decision has implications on the total transportation cost of the containers in the yard. In this study, we assume that the arriving bookings are directly placed over the containers currently staying at the storage area. This assumption will be explained in detail in the following chapters. As a result, we reduce the three-dimensional storage

space into two-dimensions. We propose mathematical models for the problem. The first mixed integer programming (MIP) model assumes that the capacity at the yard at any time is sufficient for the incoming container, that is, no infeasibility due to capacity constraint occurs during the planning horizon. The second and third models relax this assumption by incorporating the usage of a temporary storage space for excess containers that violate the capacity constraint. The usage of this temporary space is penalized in the objective function. GAMS 23.9.4 with CPLEX solver is used for the exact solutions, and to evaluate the performance of the Lagrangean Relaxation approach designed for the three models.

The remainder of this thesis is organized as follows. Chapter 2 provides a review of the literature on related previous work. Chapter 3 is devoted to explain the important concepts of container terminals, our problem definition and our first mathematical model. Chapter 4 introduces two mathematical models with temporary storage space. Solution methods; namely Lagrangean Relaxation approaches with subgradient optimization and heuristics for the Container Storage problem are discussed in Chapter 5. Experimental design and computational results are presented in Chapter 6. Conclusions are discussed in Chapter 7 along with future research directions.

CHAPTER - 2

LITERATURE REVIEW

At a container terminal, there are operations pertaining to the storage yard for the containers. These include the assignment of the incoming containers to their storage locations, handling and placement, storage until departure time, and the removal of the containers from the storage locations at the time of shipment. The handling of a container during its placement or removal can significantly affect the efficiency and cost at a container terminal, as it involves the removal of several containers from the top of the stack to reach the target container below. In literature, these unproductive moves of the handling equipment are called as *reshuffling* or *rehandling*, and there are many papers on the minimization of the number of reshuffles (i.e., the number of containers that have to be moved to place/reach the target container) during handling operations. In this thesis, the problem involving all the above operations is called as the *container storage problem (CSP)*. We summarize the relevant literature in this chapter.

Container terminal management operations have various important decisions and many of them are interrelated. Vis and the Koster (2003) and Steenken et al. (2004) point out that, for efficient management of container terminals, the integrated problem of all operational activities should be considered, if possible. However, such an approach has never been pursued in practice, mainly because of the intractable size of the integrated problem. In literature, the operational activities at a container

port are mostly handled separately; though some rare examples combining two or more operational activities are available. CSP is one of such problems studied separately.

In previous studies, two types of operations that cause reshuffling are defined within the context of CSP. The first operation is named as *relocation*, which involves changes in the location of the existing containers at the storage yard in order to assign and place the incoming container to a convenient location. The second operation is the relocation of the containers from the top of stack for the *removal* of a target container from the storage yard. The studies in literature mainly aim to minimize reshuffling caused by these two operations. The models in this thesis try to minimize reshuffling caused by the removal operations. Relocation is not considered, as will be explained in the following chapter.

Many studies in literature consider CSP under simplifying assumptions. Kim (1997) is one of the first researches to study CSP. In their research, a mathematical model was proposed to estimate the expected number of rehandles to pick up an arbitrary container and the total number of rehandles to pick up all the containers in a bay for a given initial stacking configuration. They assumed that every rehandle container could be moved within the same storage bay. In other words, their analysis of rehandles was restricted to a single bay (row) of containers, which simplified the computations a great deal. This assumption is not used in our study for a more realistic representation of the problem.

Kim et al. (2000) proposed a dynamic programming model to determine the storage location of an arriving export container. In their research, the incoming containers were separated into three groups according to their weight as a simplifying rule. Zhang et al. (2003) studied to balance workloads among container blocks (i.e., the total number of containers in each block) for even distribution of containers at the yard to minimize total transportation cost. The research was based on the structure and the constraints of the container terminal in Hong Kong, which had limited storage space and a large amount of reshuffling. The container terminals in Turkey have similar structures, as well.

Kim and Hong (2006) also studied the problem of minimizing the number of reshuffles in block stacking systems. A branch-and-bound algorithm and a heuristic rule were suggested in their study. Nishimura et al. (2009) studied storage problems encountered in a container terminal that only provided service to mega-containerships. As the mega-containerships have priority at the terminal, the containers from these ships are stowed onto a smaller feeder ship without even entering the port. Hence, their storage problem had a different structure from other CSP studies. As the container terminals in our country cannot accommodate mega-containerships due to technical restrictions, we do not consider mega-containerships in our study.

A mathematical model was developed by Park and Seo (2009) to minimize the number of obstructive object moves. Their paper utilized a genetic algorithm to solve the corresponding NP-hard problem. In their study, it was assumed that the top containers removed from a stack to reach the containers below were placed back to their location again after removing the target containers. This assumption is in line with ours, as will be explained in the following chapter. Lee and Lee (2010) also studied the problem of retrieving containers from a yard. Unlike other studies, their optimization goal was to minimize the number of container movements as well as the crane's working time. They suggested a three-phase heuristic that aimed to solve these two mixed integer problems through the phases.

In this thesis, we model CSP using an interval scheduling-based approach. For this reason, we provide a brief literature review of the relevant studies, as well. *Interval Scheduling (IS)* is a scheduling problem that is commonly used in the service and manufacturing sectors. In this problem, jobs (tasks) having predetermined ready times and deadlines are to be processed by parallel machines (resources). The problem is typical for reservation systems and has many real-life applications such as resource allocation, classroom scheduling, transportation systems, etc. The problem can be analyzed in two categories as *Fixed Job Scheduling (FJS)* and *Variable Job Scheduling (VJS)*. In FJS, the processing time of each job is exactly equal to the difference between its ready time and deadline, which means each task should start processing just as it arrives, or else it leaves the system without being processing. In

VJS, which is also referred to as parallel machine scheduling with time windows, the processing time is smaller than the difference between predetermined ready time and a deadline. Therefore, the scheduler has some flexibility on determining the start time of an arriving job, while ensuring that each job should start processing before its latest start time. Each of these two IS problems has two variants based on their objective functions. In *tactical* IS, the minimization of the total cost of the resources to process all jobs is of concern. Alternatively, *operational* IS assumes a fixed number of resources and tries to select a subset of jobs for processing in order to maximize the total profit/number of the processed jobs. We provide some examples from the literature below to each of these variants on many different contexts.

Fischetti et al. (1987) studied on the *Tactical Fixed Job Scheduling (TFJS)* problem. Their problem was defined as a bus driver scheduling problem and they aimed to find a proper set of driver duties at minimum cost. TFJS was also studied by Eliiyi et. al. (2009a) for a real-life vehicle scheduling problem. The authors considered the minimum-cost scheduling of different vehicle types on a predetermined set of one-way trips. In another application, Kroon (1990) utilized the TFJS problem as the core model in capacity planning of aircraft maintenance personnel for an airline company.

A later study by Kroon et al. (1995) dealt with the operational variant of this problem with a given number of maintenance engineers and priorities defined for the maintenance activities. The *Operational Fixed Job Scheduling (OFJS)* was also studied by Wolfe and Sorensen (2000) to model the problem of scheduling earth-observing satellites. OFJS was applied to another real-life application in classroom scheduling by Kolen and Kroon (1991).

Eliiyi and Azizoglu (2006, 2010, and 2011) considered the OFJS problem on identical machines with operating time constraints for the machines, where the objective was to maximize the total weight of the processed job subset. They proved that the problems with different forms of time constraints were strongly NP-hard, and investigated several special polynomially solvable cases. They proposed branch and bound algorithms that returned optimal solutions for small and medium-sized

problem instances in reasonable solution times, and developed efficient heuristic approaches for considerably large problem instances. In a later study, machine eligibility was taken into consideration for the OFJS problem (Eliiyi and Azizoglu, 2008). In this study, the jobs had machine- dependent weights, where each job yielded a different profit depending on the processing machine, and a negative profit meant that the corresponding machine was not eligible to process the job. Bekki and Azizoglu (2008) addressed an OFJS problem on uniform parallel machines having different processing speeds, and proposed a branch and bound algorithm. Two survey papers have been published by Kovalyov et al. (2007) and Kolen et al. (2007) on FJS. Kovalyov et al. suggested a general formulation of the problem, reviewed known models and algorithms whereas the study by Kolen et al. (2007) presented the complexity of various FJS problems and suggested appropriate solution algorithms. The interested reader is referred to these nice survey papers for insight on the FJS problem.

Rojanasoonthon et al. (2003) studied *Operational Variable Job Scheduling (OVJS)* in the context of data relay satellite system. Two efficient algorithms were proposed in their study. Eliiyi et al. (2009b) studied another real-life application on optimal berth allocation at seaports, which involved the assignment of vessels arriving at the port to appropriate berths within their time windows, while maximizing the total profit from the served vessels. This OVJS problem was considered with machine-dependent weight definitions for handling eligibility, as the ships differed in size and draft. An integer programming model was developed for the NP-hard problem and a constraint-graph-based construction algorithm was developed for generating near optimal solutions. The authors also used genetic algorithm and other improvement algorithms to enhance the solution.

We model CSP as a TFJS problem, which is especially suitable for transit containers, as the arrival and departure times of the incoming ships can be taken as predetermined fixed parameters. In this manner, the dwell time of a container at the terminal can be defined as a predetermined time interval. As to the best of our knowledge, ours is the first study to model CSP as a TFJS problem. While TFJS is polynomially solvable in its basic form, the model for CSP has additional decision

variables and constraints that complicate the problem. For this reason, besides attempting exact solutions, we have tried *Lagrangean Relaxation (LR)* approach and new heuristics as possible solution procedures, which will be explained in detail in Chapter 5.

The LR method is widely used in integer and mixed integer problems. Through LR approach, many hard problems can be modeled as relatively easy problems. Many hard problems are composed of “nice” and “complicating” constraints. The aim of the LR approach is to relax the complicating constraints of the original problem so that relaxed model is an easy-to-solve one. The method multiplies these constraints with corresponding penalty costs (Lagrangean multipliers) and carries them into the objective function. By dualizing those constraints, a hopefully easy problem is obtained, whose optimal value gives an upper bound (lower bound) on the optimal value of the original maximization (minimization) problem.

Held and Karp (1970) used LR for the traveling salesman problem, but Geoffrion (1974) firstly used the name *Lagrangean Relaxation* for this approach and used it for obtaining bounds on the optimal objective function value in integer programming. Fisher (1981) mentioned *subgradient optimization* method which is used to update lagrangean multipliers for solving the lagrangean dual problem. LR is widely used for solving assignment, scheduling and sequencing problems in literature.

As the structure of our models also resembles the transportation problem with side constraints, we also provide a brief review of the problem together with some previous studies.

Network flow programming has several special cases that are transportation, assignment and transshipment problems. The transportation problem is one of the early applications of linear programming. The famous type of transportation problem first described by Hitchcock (1941) obtains the minimum cost flow through a special type of network where a number of suppliers (sources) are to provide a number of demand points (sinks) with a commodity. Unit costs are incurred for supplying the

commodity from each source to each sink, and the objective is to determine a shipping plan that satisfies the supply and demand constraints at minimum cost. Koopman (1947) is one of the pioneer researchers to study transportation problems in economics.

The transshipment problem is the generalization of the transportation problem. In this problem, it is possible to distribute the commodity through intermediate sources and intermediate sinks, as well as from the sources to the sinks. The assignment problem is a special case of the transportation problem. We can regard that it is a problem with m sources and n sinks, all with unit supply and demand. The algorithms designed for special cases are more efficient than more general ones. Figure-1 shows the relationships between these network flow models; as we move to the right in the figure, the problems become more special. All the problems to the right of the generalized minimum cost flow problem can be solved with an algorithm designed for this generalized problem.

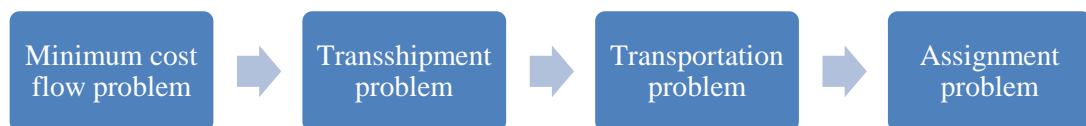


Figure 1 Related Network Flow Problems

Ahuja et al.(1993), Glover et al (1992), Kennington and Helgason (1980) and Murty (1992) developed efficient solution algorithms for the traditional transportation problem and network problems. The basic transportation problem can be solved efficiently. One special case of the transportation problem is named as the time-minimizing transportation problem, in which a time is associated with each shipping route. The objective of this problem is to minimize the maximum time to transport all supply to the destinations. Hammer (1969) firstly studied the problem and proposed an algorithm. The threshold algorithm for the problem is then developed by Garfinkel and Roa (1971). Merrill and Tobin (1969) consider the problem in which the total supply exceeds the total demand and propose an algorithm

for this problem. They generate a sequence of improving lower bounds on the maximum time. Sharma and Swarup (1978) are other authors to study the basic transportation problem. Laura (1964) studied on theoretical aspects of the problem, as Balas and Hammaer (1964) did.

Although the basic version is easy, many variants of the transportation problem are considerably harder. A more complex version of the transportation problem, namely, the generalized transportation problem, can be formulated as follows:

$$\text{Minimize } \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \quad (2.1)$$

Subject to

$$\sum_{j=1}^n x_{ij} \leq a_i \quad \forall i \quad (2.2)$$

$$\sum_{i=1}^m p_{ij} x_{ij} = b_j \quad \forall j \quad (2.3)$$

$$x_{ij} \geq 0 \quad \forall i, j \quad (2.4)$$

where

x_{ij} : the amount of items moved from source i to sink j

c_{ij} : the cost of moving one item from source i to sink j

p_{ij} : positive constraint rather than unity

a_i : the supply available at each source i

b_j : the demand at each sink j

m : total number of sources

n : total number of sinks.

Klingman and Russel (1975), Chen and Saigal (1977), Glover and Klingman (1985) discussed similar network problems with linear side constraints. Special techniques have been developed to solve these models in these studies. Many storage problems can be modeled as transportation problem with exclusionary side

constraints (TPESC) when the items from some pairs of sources cannot be stored simultaneously in the same warehouse. The TPESC was formulated by Sun (2002) as a binary mixed integer programming model. In their study, two branch and bound algorithms were developed and implemented. The TPESC can be formulated as follows.

$$\text{Minimize } \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \quad (2.5)$$

Subject to

$$\sum_{i=1}^m x_{ij} \leq C_j \quad \forall j \quad (2.6)$$

$$\sum_{j=1}^n x_{ij} = a_i \quad \forall i \quad (2.7)$$

$$y_{ij} + y_{kj} \leq 1 \quad \text{for } \{i,k\} \in B_j \text{ and } \forall j \quad (2.8)$$

$$x_{ij} - M_{ij} y_{ij} \leq 0 \quad \forall i, j \quad (2.9)$$

$$y_{ij} \in \{0,1\} \quad \forall i, j \quad (2.10)$$

$$x_{ij} \geq 0 \quad \forall i, j \quad (2.11)$$

where

c_{ij} : the cost of the moving one item from source (supplier) i to sink (warehouse) j

x_{ij} : the amount of items moved from source i to sink j

C_j : the demand (capacity) at sink (warehouse) j

a_i : the supply available at each source i

m : total number of sources

n : total number of sinks

B_j : $\{\{i,k\} | \text{items from sources } i \text{ and } k \text{ cannot be simultaneously shipped to destination } j\}$.

In our study, containers from different sources can be assigned at the same location, but the objective function can penalize some assignments of such, as they may create extra reshuffling.

Sun (1998) developed a nonlinear mixed binary integer programming formulation for the storage yard space assignment problem at a container terminal, which can mathematically be expressed as a transportation problem with exclusionary nonlinear side constraints. Cao (1992) also proposed a nonlinear programming formulation. Cao and Uebe (1995) studied on a transportation problem with nonlinear side constraints. The difficulty of the problem increased extremely in the presence of these constraints. They develop a Tabu Search approach to solve the problem. Their model, having the same notation as the above model, is expressed below:

$$\text{Minimize } \sum_{i=1}^m \sum_{i=1}^n c_{ij} x_{ij} \quad (2.12)$$

Subject to

$$\sum_{i=1}^m x_{ij} \leq C_j \quad \forall j \quad (2.13)$$

$$\sum_{j=1}^n x_{ij} = a_i \quad \forall i \quad (2.14)$$

$$x_{ij} x_{kj} = 0 \quad \text{for } \{i, k\} \in B_j \text{ and } \forall j \quad (2.15)$$

$$x_{ij} \geq 0 \quad \forall i, j \quad (2.16)$$

Cao (1992) developed a special-purpose branch and bound algorithm for this problem, and the commercial package CPLEX was used to solve some small test problems. The solutions of these problems have shown the complexity of the transportation problem with exclusionary side constraints.

Another type of transportation problem includes stochastic demand and can be named as stochastic transportation problem, which was discussed by Williams (1963)

with penalties for not satisfying the demand. The objective is to minimize total transportation costs plus the penalty cost. In this thesis, we incur a penalty cost when a temporary storage space is used for the incoming containers (over supply). We deal with this case in our models for the Container Storage Problem with Excess Containers in Chapter 4.

Kuno and Utsunomiya (2000) studied on a production-transportation problem with a nonlinear, concave and nondecreasing production cost. They formulated their problem as follows.

$$\text{Minimize } \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} + \sum_{i=1}^m f_i(y_i) \quad (2.17)$$

Subject to

$$\sum_{i=1}^m x_{ij} \leq y_i \quad 0 \leq y_i \leq u_i, \quad i \in M \quad (2.18)$$

$$\sum_{j=1}^n x_{ij} = b_j \quad j \in N \quad (2.19)$$

$$x_{ij} \geq 0 \quad (i, j) \in A \quad (2.20)$$

where

x_{ij} : the amount of items moved from source i to sink j

y_i : units of production

b_j : the demand at sink j

c_{ij} : the cost of shipping unit by route $(i, j) \in A$

$f_i(y_i)$: the cost of producing y_i

u_i : production capacity

M : total number of sources

N : total number of sinks

A : the set of transportation routes.

They proposed a branch and bound algorithm for solving the minimum cost production-transportation problem. The bounding operation is implemented through

two stages: the first stage based on a linear programming relaxation and the second stage based on a Lagrangean relaxation.

Romeijn and Sargut (2011) studied on stochastic transportation problems with single-sourcing constraints. They proposed a branch and price algorithm to solve this problem which account for both uncertainty in the demands and nonlinear cost structures. The branch-and-price algorithm is based on a set partitioning formulation of the problem. Besides, they study the corresponding pricing problem which turns out to be a knapsack problem with variable item sizes and concave costs.

Our models include a bi-partite summation as in the above model, where one part is nonlinear. The next chapter defines the problem, and presents the first mathematical model for the CSP. The resemblance to a nonlinear transportation problem with side constraints will be explained clearly, as well.

CHAPTER - 3

CONTAINER STORAGE PROBLEM

Maritime transport often does not carry the freight from the point of origin to the destination. Transportation movement involving more than one mode is called intermodal transportation.

In intermodal transportation systems, container terminals are facilities that allow the change on types of transport, maritime, road and railroad transportation or transfer from one ship to another. An important function of the container terminals is transshipment of the containers safely, rightly and on time. In container terminals provide temporary storage for the containers in order to transfer containers between a large-scale maritime transportation and medium-scale rail or small-scale road transportation. There are five main operations which affect the competitiveness of a container terminal (Yeo et al., 2008):

1. Ship entrance and berthing
2. Loading/unloading or loading/discharging
3. In -port container operations
4. Storage and stowage
5. Intermodal handling operation

The competitiveness of a container terminal is related with utilization of

operations in these systems. The following Figure-2 shows that these systems and decision problems in the operational level of these systems.

1. Ship entrance and berthing	2. Loading/ Unloading	3. In -port container operations	4. Storage and stowage	5. Intermodal handling operation
<ul style="list-style-type: none"> • Assignment of Berth • Acceptance of Ship Scheduling 	<ul style="list-style-type: none"> • Assignment of Crane • Loading/ Unloading Scheduling 	<ul style="list-style-type: none"> • Selection of Vehicle • Vehicle-Container Assignment • Vehicle Routing 	<ul style="list-style-type: none"> • Placement of Storage Location 	<ul style="list-style-type: none"> • Assignmnet of Transferring Vehicle • Scheduling of Transfers

Figure 2 Decision Problems in Container Terminals

Our problem belongs to storage and stowage category. In the rest of this section, the details of our problem and the terms used in the problem will be discussed.

Container terminals are transit areas and storage areas for containers during transport from one point to another. The different operational problems are occurred in the container storage area. It is difficult to handle the combinations of these problems. There are few papers which handle more than one category. We will also handle only storage operations in our study.

Berth is an area which ships are docked and containers are discharged from or loaded to the ship. In berth, there are quay crane for unloading and loading operations. There are two types of terminal layout. The containerships are serviced at berth in line type or in the indented type which are shown in Figure-3. At the linear type, loading and unloading operations can be made only one side, while these operations can be appropriate for two side of ship at intended type. (Imai et al., 2007).

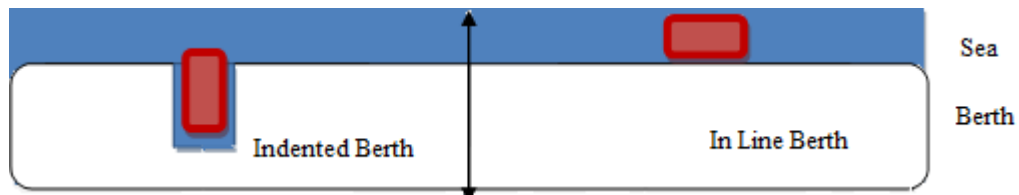


Figure 3 Berth in line and in the indented type

Containers are transported from these berths to temporary container storage area. Storage area is a place where containers wait their next transport vehicle. Containers are transported to the storage area after getting off the ship. Container storage problem decide where to place each container in storage area. Position of the storage area is 3-D structure. Row and column numbers determine horizontal position, but position of container cannot be defined only with two numbers. We also need to indicate the vertical distance in terms of number of containers. It is shown that there is three-dimensional stacking area in a storage location in Figure-4. Every container is placed to appropriate storage location with column, row and stack number. The maximum height of stack is equal to capacity of this stack.

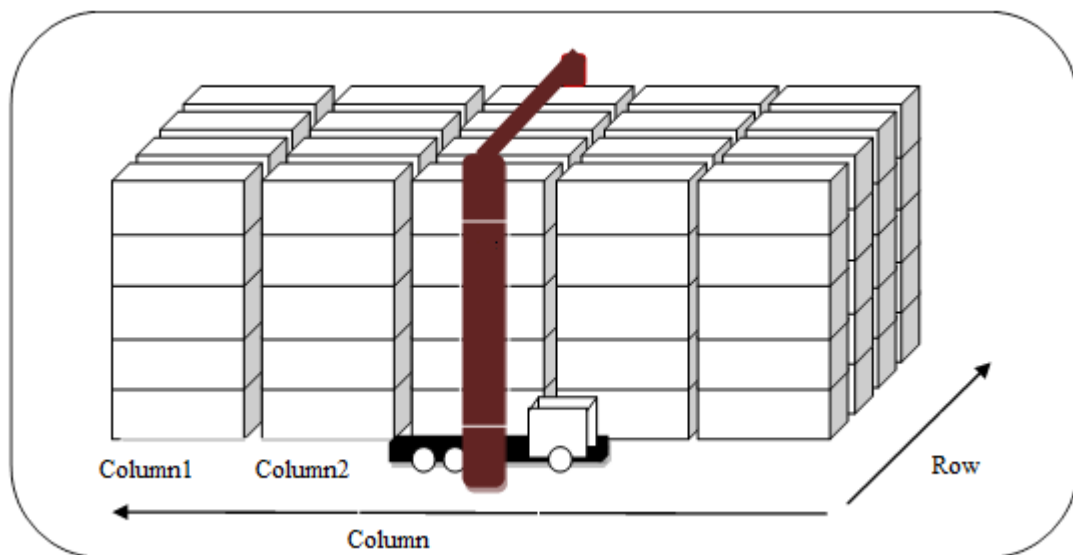


Figure 4 Stack in storage location

Figure-5 gives the side view of a column. After containers are carried to the column in horizontal plane, they are placed with the crane to its location. Transportation of containers from in their berth to their storage location or from

storage location to the berth and the placement of containers in the stack are known as container movements. These movements can be classified into two categories.

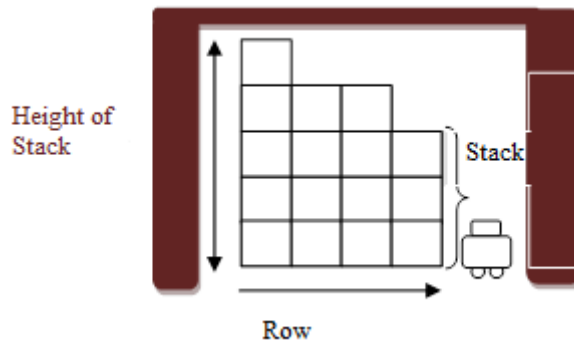


Figure 5 Side cross-section of the container a column

There are two movements of containers in the terminals.

1. Horizontal Movement

- a. After unloaded from the inbound vessel, incoming containers are carried to their allocated stack in the storage area by truck.
- b. Outgoing containers are carried by trucks from storage area to the outbound vessel.

These operations are known as ground transportation in the terminal containers.

2. Vertical Movement

- a. Incoming containers are placed with crane in a predetermined location.
- b. Outgoing containers are removed with crane from their position in the storage area, but the containers above of removal containers in the same stack should be removed and placed another or same location. This operation can be named as reshuffling or rehandling.

Ports of Turkey are located in a strategic location in the East-West and North-South direction is at the intersection of international transport corridors. To be able to compete in international markets, ports must reduce the costs of the port. Reshuffling is not profit, but also it is high cost operation for terminal containers. For this reason, it is important to minimize the number of reshuffling.

Four types of containers are stored in the port area. They are import, export, and transit and cabotage containers. Import containers come to the port from abroad by seaway and leave the port by road. Export containers come to the port by road and leave by seaway. Cabotage containers are defined as the containers that are transferred from one of the Turkey ports to the other one. Transit containers arrive with a vessel and depart with another vessel. In many container terminals, storage area is divided into three areas as import, export and transit areas. Another approach that is used in Hong Kong (one of the busiest Ports in the world) is dividing the storage area in stack level (not in block level). Therefore, there is only one type of container in each stack. After the stack is being empty, this stack may be dedicated to the different type of container (Zhang et al., 2003).

An important concept used in the project is booking. A booking, belongs to a single customer, represents a set of containers. This is a set of containers cannot be separated and must be carried on the same ship together, and a single bill of lading is held every booking. Containers of a booking can be stored at different points in the storage area.

The size of the problem depends on the number of docks that vessels can berth the number of stacks that are dedicated to the transfer bookings, the number of bookings that are taken into account along the planning period. We will encounter a large problem in busy ports because of the number of bookings.

In the container terminals, there are two factors which effects total cost. One of the factors is transportation cost, which includes the transport cost of transit containers unloaded from ship to the storage point and storage location to the loaded to their ship. This cost is directly related with the distances between the storage point and unloading/loading points. This is can be named as horizontal cost.

Other factor that affects total cost in storage area is related to the vertical position in the stack. In storage area of container terminals, containers are usually stored like four or five blocks in a stack. The position of containers in the stack

directly affects reshuffling cost, because the number of vertical movements (reshuffling number) of the crane is determined by the number of other containers on the container which we want to reach. Therefore, a second factor relates to the vertical movement and cost of container within the port area. The horizontal movement of the crane is required to reach a container in the block, but the cost of vertical movement is much higher.

The aim of the Container Storage Problem (CSP), inputs, and outputs of the constraints are given in Figure-6.

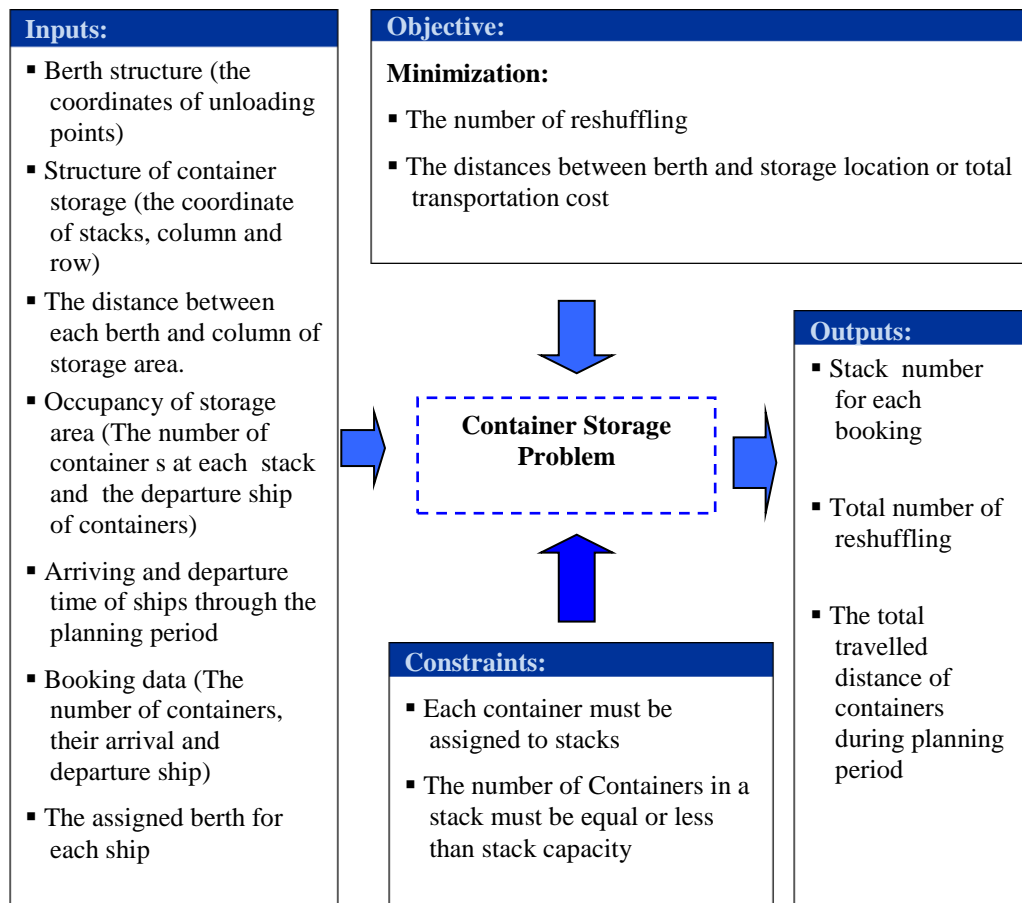


Figure 6 Container Storage Problem

Notations:

B : number of bookings

K : number of storage locations represented by a pair of row and bay numbers

b : index for bookings, $b = 1, \dots, B$

k : index for storage locations represented by a pair of row and bay numbers, $k = 1, \dots, K$

a_b : the arrival time of booking b to the container terminal (storage location)

d_b : the departure time of booking b from the container terminal (storage location)

n_b : number of container in booking b

t : index for time intervals, $t = 1, \dots, |T|-1$

t_p : length of the planning horizon

T : the set of time points defined by the sorted set of arrival and departure time of bookings, with duplicates removed. That is, T is the following set with elements sorted in non-decreasing order.

$$T = \bigcup_{b \in B} \{a_b, d_b\} \quad (3.1)$$

No booking arrives or departs during time interval (T_t, T_{t+1}) for all t . In this way, we create intervals of no event. Time intervals are shown in Figure-7.

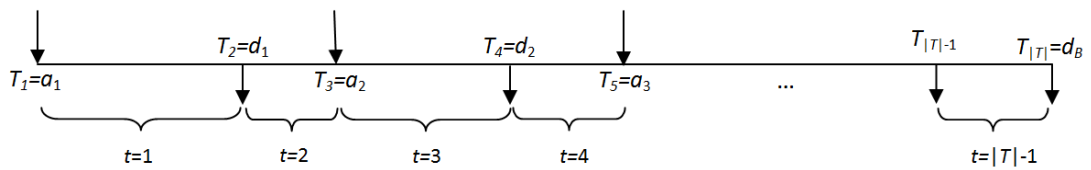


Figure 7 Time interval

$E(b)$: the set of bookings that arrive after booking b and that will leave after booking b . That is,

$$E(b) : \{j \mid j > b \text{ and } a_j < d_b < d_j\} \quad (3.2)$$

If two bookings b and $j \in E(b)$ arrive at the same time, if they are placed in the stacks in the order of their departure times, some unnecessary reshuffling moves can be avoided. That is, if $a_b = a_j$ and $d_b < d_j$, the containers of booking j should be placed below the ones of booking b , if they are placed in the same stack.

E_{bi} : the indicator of reshuffling

$$E_{bi} = \begin{cases} 1, & \text{if } i \in E(b) \\ 0, & \text{otherwise} \end{cases} \quad (3.3)$$

$E'(i)$: the set of bookings that b may cause reshuffling. That is,

$$E'(i) : \{b \mid i \in E(b)\} \quad (3.4)$$

$B(t)$: the set of bookings that will stay in the storage area during time interval t . The set is algebraically defined as follows:

$$B(t) : \{b \mid T_t \geq a_b \text{ and } T_{t+1} \leq d_b\} \quad (3.5)$$

I_{bt} : the indicator for booking and time pair.

$$I_{bt} = \begin{cases} 1, & \text{if } b \in B(t) \\ 0, & \text{otherwise} \end{cases} \quad (3.6)$$

C_k : maximum height of storage location k (maximum number of containers in stack k of the container storage area.

R : the unit cost function of reshuffling

c_{bk} : horizontal transportation cost of booking if it is stored at storage location k .

Horizontal transportation cost has four components: ground transportation from the inbound vessel to the storage area, crane movement in the stack area to the storage position, crane movement in the stack area from the storage position, and ground transportation from storage area to the outbound vessel.

Assumptions of the model

Many of these assumptions that are made in this section will be used throughout this paper.

- An inbound booking is instantaneously transported from the berth to the storage location. Since the arrival and departure times are usually defined in days in a container terminal, and the storage of the container is made within a day, this is not an unrealistic assumption.
- Similar to the first assumption, an outbound booking is instantaneously transported from the storage location to the outbound vessel.
- For two bookings b_1 and b_2 if $d_{b_1} = a_{b_2}$ then b_1 is removed first and then b_2 is placed.
- If there are multiple bookings arriving at the same time, the bookings are placed in the storage area in non-increasing order of their departure times. (e.g. if $a_{b_1} = a_{b_2}$ then $d_{b_1} > d_{b_2}$ then b_1 is stored before b_2)
- Reshuffling cost is proportional to the number of containers to be removed to reach the target container in a stack. This number is equal to the number of containers on top of the target container that are not due at the time of the target container's departure.
- Reshuffling cost does not include the removal of the containers that are due at the same time interval. In such a case, all containers will be removed from the stack, so no additional cost is assumed.
- Arriving bookings are placed over the containers currently staying at the storage area. We define this storage rule as blanket storage. That is, no shuffling is done for arriving containers.
- Moved containers are put back to their original slots, i.e. storage locations after the removal of the departing containers.
- We assume one size containers (1 TEU or 2 TEU).
- We consider only the transit containers.
- **Covering assumption:** The three-dimensional stacking area will be reduced to two-dimensional stacking area which facilitates solution of the problem. Every incoming container will be placed to the appropriate storage location (at the top

of the stack). The other containers which are located lower levels are not removed from their positions. In this case, all containers which should be substituted for booking are placed at the top storage place of the stacks; it is assumed that these current stacks have covered. When the departure time of one of the bottom container comes, crane unloads the containers which are at the top, and then the target container is removed and the top containers replaced to the stack. This assumption considerably simplifies modeling. With this assumption, three-dimensional space (row, column, and stack) of the problem decreases to two-dimensional space (row, column). This situation is illustrated in below Figure-8. Every row and column, a single stack number is specified. A stack height at any given instant is equal to the number of containers storing in the current stack.

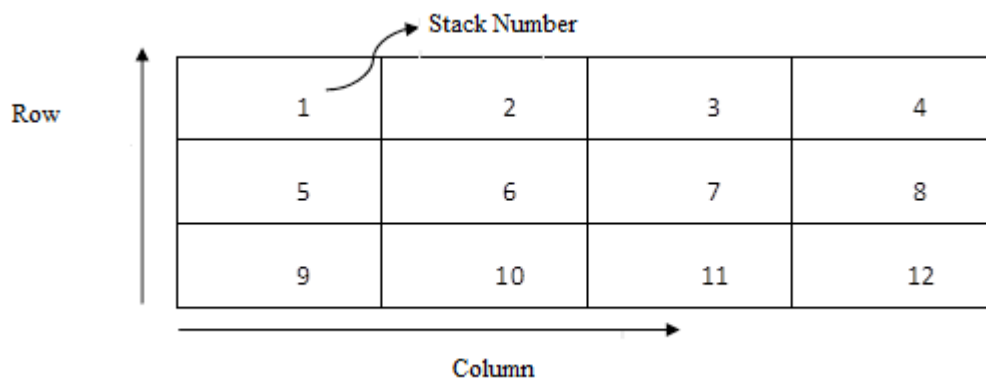


Figure 8 Storage Area under the “covering assumption”

The literature reveals that Container Storage Problem can be modeled as a Fixed Job Scheduling which is special case of IS. This model seems to be very convenient, especially for transit containers, since the port arrival and departure times of ships are already known in advance. (Road and rail connections, arrival and departure times will be unknown). Hence, the time that a container (or a booking) in storage location stays in port are also known. The model based on FJS is presented next. We have two sets of decision variables.

x_{bk} : the number of containers of booking b to be placed into storage location k .

$y_k(x_{1k}, x_{2k}, \dots, x_{bk})$: the number of reshuffles at location k , this a function of both $x_{1k}, x_{2k}, \dots, x_{Bk}$ and sets $E(b)$ for all b .

Our objective is to minimize the total cost of transportation during a planning horizon. The total cost is composed of ground transportation and crane movements in the storage area. We can now formulate our mixed integer programming model as below.

$$\text{Minimize } \sum_k R y_k(x_{1k}, x_{2k}, \dots, x_{bk}) + \sum_k \sum_b c_{bk} x_{bk} \quad (3.7)$$

$$\sum_b I_{bt} x_{bk} \leq C_k \quad \forall k, t \quad (3.8)$$

$$\sum_k x_{bk} = n_b \quad \forall b \quad (3.9)$$

$$x_{bk} \geq 0, \text{ integer} \quad \forall b, k \quad (3.10)$$

The first part of the objective function calculates the reshuffling (vertical crane movements) cost at all locations and the second part calculates the total horizontal transportation cost. Constraint set (3.8) assures that at any time interval t , at most C_k containers are assigned at each location k . Constraint set (3.9) assures that all containers in booking b are assigned to storage locations. Constraint sets (3.10) assure that if x_{ik} is nonnegative and integer.

This is a transportation problem with selective arcs and nonlinear cost function, which depends on all bookings assigned to a location. The (k, t) pairs are the suppliers and the bookings are the demand. Let us define x_{bkt} as quantity put from booking b on stack k at time period t . This amount is equal x_{bk} if booking b stays during time period t . In other words,

$$x_{bk} = \max_t x_{bkt} \quad \forall b, k, t \quad (3.11)$$

We also define T_b as the set of time periods that booking b covers.

$$\text{Minimize } \sum_k R(y_k(x)) + \sum_k \sum_b \sum_t \frac{C_{bk}}{|T_b|} x_{bkt} \quad (3.12)$$

$$\sum_b x_{bkt} \leq C_k \quad \forall k, t \quad (3.13)$$

$$\sum_{k,t} x_{bkt} \geq n_b |T_b| \quad \forall b \quad (3.14)$$

$$x_{bkt} = 0 \quad \forall b, k, t \notin T_b \quad (3.15)$$

$$x_{bkt_1} = x_{bkt_2} \quad \forall b, k, t_1 < t_2, t_1, t_2 \in T_b \quad (3.16)$$

This is a transportation problem with side constraints (3.16) and extra cost function that depends on the assigned bookings and their quantities for each location. The side constraints ensure that if some containers are assigned to a location, it will stay there during its visit at the port.

It is hard to represent the number of reshuffles y_k in terms of x_{bk} variables. y_k is equal to 0 for each $b = 1, \dots, B$. If x_{bk} is greater than 0, then add following term to y_k .

$$\sum_{i \in B} E_{bi} x_{ik} \quad (3.17)$$

This is a nonlinear function. Let us define binary variable w_{bk} , takes value 1 if some containers of booking b are assigned to location k , $x_{bk} > 0$. We can write the nonlinear objective function as below while adding the constraint set where $x_{bk} \leq n_b w_{bk}$.

$$y_k = \sum_{b \in B} \sum_{i \in B} E_{bi} w_{bk} x_{ik} \quad (3.18)$$

To linearize the objective function, we define two new decision variables to calculate the number of reshuffles.

y_{bik} : the number of reshuffles necessary for booking b at location k because of booking i .

z_{bik} : takes the value 1 if booking i creates reshuffling at location k for booking b , otherwise it takes 0.

The expression below computes the number of containers on the top of the containers from booking b at storage location (stack) k at the departure time of booking b .

$$\sum_{i \in E(b)} y_{bik} \quad (3.19)$$

This expression is equivalent to number of containers to be moved in order to reach to the containers of booking b at storage location k .

R is vertical transportation cost of each container or unit cost of reshuffling, the model can be stated as below.

Our objective is to minimize the total cost of transportation during a planning horizon. The total cost is composed of ground transportation and crane movements in the storage area. We can now formulate our mixed integer programming model as below.

$$\text{Minimize } \sum_k \sum_b \sum_{i \in E(b)} R y_{bik} + \sum_k \sum_b c_{bk} x_{bk} \quad (3.20)$$

Subject to

$$\sum_{b \in B(t)} x_{bk} \leq C_k \quad \forall k, t \quad (3.21)$$

$$\sum_k x_{bk} = n_b \quad \forall b \quad (3.22)$$

$$x_{ik} - y_{bik} \leq n_i (1 - z_{bik}) \quad \forall b, k, i \in E(b) \quad (3.23)$$

$$x_{bk} \leq n_b z_{bik} \quad \forall b, k, i \in E(b) \quad (3.24)$$

$$x_{bk} \geq 0, \text{ integer} \quad \forall b, k \quad (3.25)$$

$$y_{bik} \geq 0 \quad \forall b, k, i \in E(b) \quad (3.26)$$

$$z_{bik} \in \{0, 1\} \quad \forall b, k, i \in E(b) \quad (3.27)$$

The first part of the objective function counts the number of reshuffles (vertical crane movements) for each booking at each location and the second part calculates the total horizontal transportation cost. Constraint set (3.21) assures that at any time interval t , at most C_k containers are assigned to location k . Constraint set (3.22) assures that all containers in booking b are assigned to storage locations. Constraint sets (3.23) and (3.24) assure that if x_{ik} and x_{bk} are positive and $i \in E(b)$ then $y_{bik} = x_{ik}$, else $y_{bik} = 0$. In other words, if there are some containers from booking b at location k then each container put in location k from a booking i in set $E(b)$ is counted as reshuffling.

We will denote the total number of containers arrive on time period t as N , the total number of available location by C .

$$N = \sum_{b \in B(t)} n_b \quad (3.28)$$

$$C = \sum_{k \in K} C_k \quad (3.29)$$

According to equations (3.28) and (3.29), we can check the feasibility of problem as follow:

$$F = N - C \quad (3.30)$$

The problem is feasible if only if $F \geq 0$ and we assume this to be satisfied for CSP model.

CHAPTER - 4

CONTAINER STORAGE PROBLEM WITH EXCESS CONTAINERS

At any time period, total capacity of storage area can be less than sum of total number of incoming containers and total number of stored containers at before this time period. For our model, if there is not enough capacity at any time period for number of incoming containers, we get the infeasible solution. In real world problems, incoming containers can temporarily wait outside the storage area. The longer the containers wait on board, the longer the vessel has to stay at the port, this means higher cost for the vessel and the port has to wait a little while longer before serving another vessel.

For avoiding infeasible solution and adapting the real world problems, our model should contain dummy location in where incoming containers are placed until they leave container terminal.

4.1. Mathematical Model with No Restriction

New decision variable is added as a dummy variable in the model. New model, which is named as CSPX-1, can choose to place incoming containers into the dummy location. We do not need separate dummy storage location with columns and rows like storage location. Therefore dummy storage location is used to like as only one location.

s_b : the number of containers placed in dummy location of booking b .

Our additional objective is to minimize the number of containers placed in dummy location during a planning horizon.

If a container is placed in the dummy location, it should be charged with higher penalty cost. We use the symbol P_b to denote the penalty cost of assigning booking b to dummy location. This penalty cost depends on the arrival time of booking.

$$\max(P_b = t_p + 1 - a_b, 0) \quad (4.1)$$

If a_b is greater than planning horizon, the penalty cost is equal to 1. With this equation, we give the priority to bookings, which arrive in the storage location earlier.

Our mathematical model with no restriction is formulated as below.

$$\text{Minimize } \sum_k \sum_b \sum_{i \in E(b)} R y_{bik} + \sum_k \sum_b c_{bk} x_{bk} + \sum_b P_b s_b \quad (4.2)$$

Subject to

Constraint sets 3.21, 3.23, 3.24, 3.25, 3.26 and 3.27

$$\sum_k x_{bk} + s_b = n_b \quad \forall b \quad (4.3)$$

$$s_b \geq 0 \quad \forall b \quad (4.4)$$

The last part of the objective function calculates the penalty cost of assigning containers to the dummy location. Constraint set (4.3) also assures that all containers in booking k are assigned to storage locations or dummy storage location. Constraint set (4.4) assures that slack variable is equal or greater than 0. We can now formulate our mixed integer programming model as below.

4.2. Mathematical Model with First-Come First-Served Restriction

In the previous model (CSPX-1), incoming containers can be placed in the dummy location while there still are empty places in storage location. This is an expected result, as placing early containers to rather than later containers to the dummy location may decrease the total number of reshuffling. For this reason CSPX-1 only aims to minimize the reshuffling cost.

In real world problem however, incoming containers must be placed in the storage location if there is sufficient space. They do not wait outside because of the

containers arriving later on. Therefore, first-come first-served (FCFS) rule should apply for the incoming containers. If containers come into storage location from berth, they firstly are placed in the storage location irrespective of comparing the number of reshuffling of posterior containers. This new model (CSPX-2) makes reshuffling comparison only between the containers which arrive at the same period. We define an inventory type new decision variable.

v_t : the number of containers in storage area during period t .

The expression below computes the total number of containers times the number of periods in the storage location during the planning horizon.

$$\sum_{t \in T} v_t \quad (4.5)$$

This expression with a negative coefficient in the objective function helps to maximize the total number of containers in the storage location.

V : the unit benefit of placing a container in the storage location for a time period

$IB(t)$: the set of bookings that will come in the storage area at the beginning of time interval t . The set is algebraically defined as follows:

$$IB(t) : \{b \mid a_b = T_t\} \quad (4.6)$$

$OB(t)$: the set of bookings that will go from the storage area at the beginning of time interval t . The set is algebraically defined as follows:

$$OB(t) : \{b \mid d_b = T_t\} \quad (4.7)$$

According to all new set of definitions, the mathematical model, CSPX-2, is formulated as below.

$$\text{Minimize } \sum_k \sum_b \sum_{i \in E(b)} R y_{bik} + \sum_k \sum_b c_{bk} x_{bk} + \sum_b P_b s_b - \sum_t V v_t \quad (4.8)$$

Subject to

Constraint sets 3.21, 3.23, 3.24, 3.25, 3.26, 3.27 and 4.3

$$v_t = v_{t-1} + \sum_{b \in IB(t)} (n_b - s_b) - \sum_{b \in OB(t)} (n_b - s_b) \quad \forall t \quad (4.9)$$

$$v_t \geq 0 \quad \forall t \quad (4.10)$$

$$s_b \geq 0 \quad \forall b \quad (4.11)$$

This model is named as CSPX-2 can be modeled with new constraints(4.9). Constraint set (4.9) assures inventory conditions. It can be also named as inventory balance equation.

The constraint that says that the amount of containers in the next time period must equal the current container inventory plus what is arrived minus what is left the storage location. The language used is for the inventory control in the production scheduling problem, but this has become a standard system of equations that appears in many mathematical programs. Thus, the meaning of the variables can be very different. In this mathematical model, we use the constraint for avoiding placing the arrived containers to the dummy location.

In addition, negative coefficient of decision variable v_t in objective function force to increase the number of containers in storage area during period t .

CHAPTER - 5

SOLUTION METHODS

As we mentioned previous chapters, very huge numbers of containers are transported in ports. Moreover, it is known that the size of CSP depends on the input of problem as number of containers, the number of locations. When the number of containers is increased, the mathematical model is able to solve only small instances of the problem. We will encounter a large problem in busy ports. The problem with large instances can be solved in a long time or cannot be solved because of resource limits.

When a decision for determining the locations of containers must be made in real time, optimal solution procedure may not be appropriate in practice. This section proposes three heuristic rules that can be used in real time. These heuristics methods can be used to obtain approximate solutions for the CSP.

The solution methods for container storage location problem will be discussed in this chapter. We present each heuristic in detail.

5.1. Lagrangean Relaxation-based Approach

Lagrangean relaxation has been successfully employed by numerous researchers for integer mixed integer programming applications. In Lagrangean Relaxation method the complicating constraints are multiplied by the Lagrangean multipliers and added to objective function. We considered LR technique for obtaining lower bounds (minimization problem) on the optimal solution. Through LR approach, many hard problems can be modeled as relatively easy problems.

Two relaxations are derived for our problem. The first relaxation considers both complicating constraints (3.23) and (3.24) with two lagrangean multipliers. This approach is named as Lagrangean Relaxation for Two Constraints (LRTC). The second one considers only single constraint set (3.23). It is also named as Lagrangean Relaxation for a Single Constraint (LRSC).

5.1.1. Lagrangean Relaxation for Two Constraints

We now discuss the application of LR approach for problem in detail. Our solution method is based on the LR of the constraints (3.23) and (3.24). As we mentioned above, the first approach is named as LRTC.

We define the dual variable λ for the constraint set (3.23) and β for the constraint set (3.24). The formulation of relaxed problem is as follows:

$$L(\lambda, \beta) = \min \sum_k \sum_b \sum_{i \in E(b)} R y_{bik} + \sum_k \sum_b c_{bk} x_{bk}$$

$$+ \sum_k \sum_b \sum_{i \in E(b)} \lambda_{bik} (x_{ik} - y_{bik} - n_i + n_i z_{bik}) + \sum_k \sum_b \sum_{i \in E(b)} \beta_{bik} (x_{bk} - n_b z_{bik}) \quad (5.1)$$

Subject to

$$\sum_{b \in B(t)} x_{bk} \leq C_k \quad \forall k, t \quad (5.2)$$

$$\sum_k x_{bk} = n_b \quad \forall b \quad (5.3)$$

$$x_{bk} \geq 0, \text{ integer} \quad \forall b, k \quad (5.4)$$

$$0 \leq y_{bik} \leq x_{ik} \quad \forall b, k, i \in E(b) \quad (5.5)$$

$$z_{bik} \in \{0, 1\} \quad \forall b, k, i \in E(b) \quad (5.6)$$

When we reorganize the new objective function(5.1), we obtain the following mathematical model.

$$\begin{aligned} L(\lambda, \beta) = \min \sum_k \sum_b (c_{bk} + \sum_{j \in E(i)} \lambda_{jbk} + \sum_{i \in E(b)} \beta_{bik}) x_{bk} \\ + \sum_k \sum_b \sum_{i \in E(b)} (n_i \lambda_{bik} - n_b \beta_{bik}) z_{bik} + \sum_k \sum_b \sum_{i \in E(b)} (R - \lambda_{bik}) y_{bik} - \sum_k \sum_b \sum_{i \in E(b)} \lambda_{bik} n_i \end{aligned} \quad (5.7)$$

Subject to

$$\sum_{b \in B(t)} x_{bk} \leq C_k \quad \forall k, t \quad (5.8)$$

$$\sum_k x_{bk} = n_b \quad \forall b \quad (5.9)$$

$$x_{ik} - y_{bik} \leq n_i (1 - z_{bik}) \quad \forall b, k, i \in E(b) \quad (5.10)$$

$$x_{bk} \geq 0, \text{ integer} \quad \forall b, k \quad (5.11)$$

$$y_{bik} \geq 0 \quad \forall b, k, i \in E(b) \quad (5.12)$$

The Lagrangean problem is decomposed into three subproblems (for each decision variables). Equation (5.7) has four terms, the first term is a function x_{bk} 's and second term is a function of z_{bik} 's and third term is a function of y_{bik} 's. Last part includes only step size parameter term, can be considered with first term of this equation. This structure leads to the following three independent subproblems 1, 2 and 3.

Subproblem 1:

$$L_1(\lambda, \beta) = \min \sum_b \sum_k c'_{bk} x_{bk} \tag{5.13}$$

Subject to

$$\sum_{b \in B(t)} x_{bk} \leq C_k \quad \forall k, t \tag{5.14}$$

$$\sum_k x_{bk} = n_b \quad \forall b \tag{5.15}$$

$$x_{bk} \geq 0, \text{ integer} \quad \forall b, k \tag{5.16}$$

Where

$$c'_{bk} = c_{bk} + \sum_{j \in E(i)} \lambda_{j bk} + \sum_{i \in E(b)} \beta_{bik} \tag{5.17}$$

Subproblem 2:

Let us denote the optimal z_{bik} values of *Subproblem 2* with z_{bik}^*

$$L_2(\lambda, \beta) = \min \sum_k \sum_b \sum_{i \in E(b)} (n_i \lambda_{bik} - n_b \beta_{bik}) z_{bik} \quad (5.18)$$

Subject to

$$z_{bik} \in \{0, 1\} \quad \forall b, k, i \in E(b) \quad (5.19)$$

The solution of the *Subproblem 2* is trivial. If $n_i \lambda_{bik} - n_b \beta_{bik} < 0$, then $z_{bik}^* = 1$, otherwise $z_{bik}^* = 0$.

Subproblem 3:

We add a bound on y_{bik} to obtain a solution to the original problem. This bound is a valid inequality in the original problem. We also define the optimal y_{bik} value of *Subproblem 3* as y_{bik}^*

$$L_3(\lambda, \beta) = \min \sum_k \sum_b \sum_{i \in E(b)} (R - \lambda_{bik}) y_{bik} - \sum_k \sum_b \sum_{i \in E(b)} \lambda_{bik} n_i \quad (5.20)$$

Subject to

$$0 \leq y_{bik} \leq x_{ik}^* \quad \forall b, k, i \in E(b) \quad (5.21)$$

The solution of *Subproblem 3* is trivial. If $\lambda_{bik} \leq R$, $y_{bik}^* = 0$, else if $x_{ik}^* > 0$ and $x_{bk}^* > 0$ then $y_{bik}^* = x_{ik}^*$

We find a lower bound for the original problem $LB^t = L_1(\lambda, \beta) + L_2(\lambda, \beta) + L_3(\lambda, \beta)$, then best lower bound can be found as:

$$\max_{\lambda, \beta} \{L_1(\lambda, \beta) + L_2(\lambda, \beta) + L_3(\lambda, \beta)\} \quad (5.22)$$

$$\lambda, \beta \geq 0 \quad (5.23)$$

We use a subgradient optimization procedure to update the Lagrangean multipliers to obtain a good lower bound. It is widely known that subgradient optimization has satisfactory performance on many combinatorial optimization problems.

The subgradient method starts with initial values of λ^0 and β^0 . We generate a sequence of Lagrangean multipliers λ^t and β^t through the addition of direction vector. Hence, the updating of Lagrangean multipliers of the constraints (3.23) and (3.24) are done according to the equation.

$$\lambda_{\text{bik}}^{t+1} = \max \left\{ 0, \lambda_{\text{bik}}^t + s_{\lambda}^t (x_{\text{ik}}^t - y_{\text{bik}}^t - n_i + n_i z_{\text{bik}}^t) \right\} \quad (5.24)$$

$$\beta_{\text{bik}}^{t+1} = \max \left\{ 0, \beta_{\text{bik}}^t + s_{\beta}^t (x_{\text{bk}}^t - n_b z_{\text{bik}}^t) \right\} \quad (5.25)$$

This direction vectors are multiplied by step size s_{λ}^t and s_{β}^t . Step sizes are positive scalar. The step sizes are updated at each iteration t using the following equation:

$$s_{\lambda}^t = \frac{\alpha^t (bestUB - LB^t)}{\sum_{b,k,i \in E(b)} (x_{\text{ik}} - y_{\text{bik}} - n_i + n_i z_{\text{bik}})^2} \quad (5.26)$$

$$s_{\beta}^t = \frac{\alpha^t (bestUB - LB^t)}{\sum_{b,k,i \in E(b)} (x_{\text{bk}} - n_b z_{\text{bik}})^2} \quad (5.27)$$

Where the best upper bound (*bestUB*) is the objective value of the best-known feasible solution and LB^t is objective value of the LR problem with multipliers λ^t and

β which are greater than 0. The initial values for Lagrangean multipliers are set to 0.

The step sizes depend on the parameter α which is scalar between 0 and 2. In our subgradient method, initial value of step size parameter sets to 0.5.

There are two stopping criteria for our subgradient method. Stopping rules are discussed below:

1. When BestUB and LB^t is equal, the method returns optimal solution,
2. When iteration limit is reached,

The upper bound UB is obtained by applying a heuristic method.

Upper Bounding Scheme:

Let us denote the optimal y_{bik} values for upper bound with \check{y}_{bik} . If $x_{ik}^* > 0$, $x_{bk}^* > 0$, then we set $\check{y}_{bik} = x_{ik}^*$, else we set $\check{y}_{bik} = 0$.

Notations:

The notations which are used in subgradient optimization algorithm as like following:

<i>bestUB</i>	: Current best upper bound
<i>bestLB</i>	: Current best upper bound
<i>M</i>	: Large integer
<i>T</i>	: iteration index
<i>iteLimit</i>	: Maximum number of iterations

s_λ^t	: stepsize for λ at iteration t
s_β^t	: stepsize for β at iteration t
α^t	: stepsize parameter at step t
<i>optimal</i>	: indicator for optimal solution
<i>noImpCount</i>	: number of iteration for which <i>bestLB</i> does not improves
<i>noImpLimit</i>	: limit for number of iterations with no improvements in <i>bestLB</i>
UB^t	: upperbound at iteration t
LB^t	: lowerbound at iteration t
λ_{bik}^t	: lagrangean multiplier λ_{bik} at iteration t
β_{bik}^t	: lagrangean multiplier β_{bik} at iteration t
<i>multiplier</i>	: constant to update α^t

Lagrangean Algorithm

Step 1. Initialization

The initial values for Lagrangean multipliers (λ^0, β^0), step sizes (s_λ^t, s_β^t) are set to 0.

The step size parameter (α^t) is equal to 0.5.

bestUB = M and bestLB = 0

$t=1$, *iteLimit* = 50 and 100

$s_\lambda^t = 0$ and $s_\beta^t = 0$

$\alpha_t = 0.5$, *multiplier* = 0.5 and *optimal* = 0

noImpCount = 0 and *noImpLimit* = 10 or 5

Step 2. Begin to Subgradient Optimization

Subgradient optimization is used to update the Lagrangean multipliers to obtain a better lower bound until the stopping conditions are satisfied.

while ($t \leq \textit{iteLimit}$ and $\lceil \textit{bestLB} \rceil \neq \lfloor \textit{bestUB} \rfloor$)

{

Step 2.1. Compute Lower Bound and Upper Bound

Solve subproblem 1 using CPLEX, find $L_1(\lambda, \beta)$ and x_{bk}^* values

Solve subproblem 2 using CPLEX, find $L_2(\lambda, \beta)$ and z_{bik}^* values

Solve subproblem 3 using CPLEX, find $L_3(\lambda, \beta)$ and y_{bik}^* values

If ($x_{ik}^* > 0$ and $x_{bk}^* > 0$)

$$\tilde{y}_{bik} = x_{ik}^*$$

Compute Upper Bound

$$UB^t = \sum_k \sum_b c'_{bk} x_{bk}^* + \sum_b \sum_i \sum_k \tilde{y}_{bik}$$

If ($UB^t < bestUB$)

$$bestUB = UB^t$$

Compute Lower Bound

$$LB^t = L_1(\lambda, \beta) + L_2(\lambda, \beta) + L_3(\lambda, \beta)$$

If ($LB^t > bestLB$)

$$bestLB = LB^t$$

else

noImpLimit++

if ($\lfloor bestUB \rfloor = \lceil bestLB \rceil$)

{

 optimal = 1;

 break while;

}

Step 2.2. Continue to Subgradient Optimization

Calculate stepsize for λ and β

$$s_\lambda^t = \frac{\alpha^t (bestUB - LB^t)}{\sum_{b,k,i \in E(b)} (x_{ik} - y_{bik} - n_i + n_i z_{bik})^2}$$

$$s_\beta^t = \frac{\alpha^t (bestUB - LB^t)}{\sum_{b,k,i \in E(b)} (x_{bk} - n_b z_{bik})^2}$$

Update λ and β

$$\lambda_{bik}^{t+1} = \max \left\{ 0, \lambda_{bik}^t + s_{\lambda}^t (x_{ik}^t - y_{bik}^t - n_i + n_i z_{bik}^t) \right\}$$

$$\beta_{bik}^{t+1} = \max \left\{ 0, \beta_{bik}^t + s_{\beta}^t (x_{bk}^t - n_b z_{bik}^t) \right\}$$

If (*noImpCount* = *noImpLimit*)

{

$\alpha^t = \alpha^t * multiplier;$

noImpCont = 0;

}

}//end while

Step 3. Print Solution

If (optimal > 0)

Print optimal solution

Else

Print approximate solution

5.1.2. Lagrangean Relaxation for Single Constraint

The second relaxation method based on relaxed the only single constraint set (3.23). Now, we define the dual variable λ to multiply the constraint set (3.23).

$$L(\lambda) = \min \sum_k \sum_b \sum_{i \in E(b)} (R - \lambda_{bik}) y_{bik} + \left(\sum_k \sum_b c_{bk} + \sum_{i \in E(b)} \lambda_{bik} \right) x_{bk}$$

$$+ \sum_k \sum_b \sum_{i \in E(b)} \lambda_{bik} n_i z_{bik} - \sum_k \sum_b \sum_{i \in E(b)} \lambda_{bik} n_i \quad (5.28)$$

In this chapter, two independent subproblems are considered for our model. *Subproblem 1* includes the constraint set (3.24) which is not relaxed in this approach. The objective function of *Subproblem 1* represents as:

$$L_1(\lambda) = \min \sum_k \sum_b \left(c_{bk} + \sum_{i \in E(b)} \lambda_{bik} \right) x_{bk} + \sum_k \sum_b \sum_{i \in E(b)} n_i \lambda_{bik} z_{bik} \quad (5.29)$$

While *Subproblem 2* is same with LRTC approach' *Subproblem 2*, *Subproblem 1* has constraints set following:

$$x_{bk} \leq n_b z_{bik} \quad \forall b, k, i \in E(b) \quad (5.30)$$

$$z_{bik} \in \{0, 1\} \quad \forall b, k, i \in E(b) \quad (5.31)$$

The subgradient method for LRSC needs only a single lagrangean multiplier λ^0 which is used for constraint(3.23). Lagrangean multiplier is updated at each iteration t according to the following equation.

$$\lambda_{bik}^{t+1} = \max \left\{ 0, \lambda_{bik}^t + s_\lambda^t (x_{ik}^t - y_{bik}^t - n_i + n_i z_{bik}^t) \right\} \quad (5.32)$$

The step size, which is positive scalar, is updated at each iteration t using the following equation:

$$s_\lambda^t = \frac{\alpha^t (bestUB - LB^t)}{\sum_{b,k,i \in E(b)} (x_{ik}^t - y_{bik}^t - n_i + n_i z_{bik}^t)^2} \quad (5.33)$$

The step sizes depend on the parameter α^t which is scalar between 0 and 2. In our subgradient method, initial value of step size parameter sets to 0.5.

The upper bound UB is obtained by applying a heuristic method as LRTC.

5.1.3. Lagrangean Relaxation with Single Lagrangean Multiplier

For improving the lower bound value, we discuss a modification of our LR approach, i.e. *lagrangean relaxation with single lagrangean multiplier (LRSLM)*. Subproblem1 always gives a solution that satisfies the capacity restrictions of the locations. Also, the values of the decision variables z_{bik} are related with the values of y_{bik} in our model. If y_{bik} assumes a positive value, z_{bik} should be equal to 1. However, when we decompose the problem into sub-problems, z_{bik} behaves independent of y_{bik} . Based on this observation, we apply another relaxation to the model. Namely, we relax only constraint set(3.23), and ignore constraint set (3.24) defining a single lagrangean multiplier set λ_{bik} for constraint set(3.23).

Thus, the problem is decomposed into two subproblems. While new *Subproblem 1* is same with *Subproblem 1* of LRTC approach, the solution of *Subproblem 2* is different. In this subproblem, we add a bound on y_{bik} to obtain a feasible solution to the original problem. This bound is a valid inequality in the original problem. So *Subproblem 2* becomes:

$$L_2(\lambda) = \min \sum_k \sum_b \sum_{i \in E(b)} (R - \lambda_{bik}) y_{bik} - \sum_k \sum_b \sum_{i \in E(b)} \lambda_{bik} n_i \quad (5.34)$$

Subject to

$$0 \leq y_{bik} \leq x_{ik}^* \quad \forall b, k, i \in E(b) \quad (5.35)$$

The solution of *Subproblem 2* is trivial. When $\lambda_{bik} \leq R$, $y_{bik} = 0$, else if $x_{ik}^* > 0$ and $x_{bk}^* > 0$ then $y_{bik}^* = x_{ik}^*$. The values for z_{bik}^* are then determined depending on the optimal values for y_{bik} , i.e., y_{bik} . Namely, we set $z_{bik} = 1$ if $y_{bik}^* > 0$, else we let $z_{bik} = 0$.

5.2. Heuristics

The first heuristic approach (H1) consists of three phases; executed one after the other, namely the initial phase that generates a feasible placement for containers at the storage location with a simple mathematical model, the selection of the container that determines which containers should be relocated, the movement phase that determines new feasible location for selected containers. The heuristic terminates when the iteration reaches the iteration limit, determined by us.

The initial feasible solution of our first heuristic, H1 aims to minimize the horizontal cost on the port. In addition, this solution is found by avoiding the vertical cost. The following problem is solved to find the storage location of each container in the port. In other words, we can find the initial feasible x_{bk} values with this mathematical model as *subproblem1* for CSPX-1 of Lagrangean Relaxation method with small variants on the objective function.

$$\min \sum_b \sum_k c_{bk} x_{bk} + \sum_b P_b s_b \quad (5.36)$$

Subject to 5.18, 5.19 and 5.20

After finding the storage location of all containers, the reshuffling for each stored container can be found. If $x_{ik}^* > 0$ and $x_{bk}^* > 0$, we can find the reshuffling amount for the initial solution.

Containers are placed according to their horizontal cost and H1 is implemented to reduce total transportation cost. In other words, we attempt to find a better feasible solution for CSP with this improvement heuristic. In this section, the steps of H1 will be explained with details.

To minimize the number of reshuffles, we must swap containers into the storage area or relocate containers to empty locations, which are available location during the process time of these containers.

The implementation of proposed selection and movement rules of the container for H1 is represented in this section. We select the container caused the highest reshuffling in order to be swapped with other containers / relocate them different from their initial location. The rule for the selection of booking on storage area is determined according to the numbers of reshuffles. The initial solution for the heuristic provides a feasible placement. According to initial feasible placement, the numbers of reshuffles (y_{bim}) are calculated and sorted in descending order. This descending list provides which locations have maximum reshuffling values. After that, we can start by selecting the container from this list.

We can demonstrate this selection rule with a simple example: Let's say that the initial solution puts booking b and i at the same location m . If these bookings caused the number of reshuffles (y_{bim}) at location m and the amount of reshuffling is the highest one in descending list, we randomly select one of booking number. After that this selected container is used to relocated/removed from location m to different location. With this method, we aim to reduce the number of reshufflings at location m . In order words, this method is carried out to find out feasible solutions with minimum number of reshuffles

It is necessary here to clarify exactly what is meant by the symbol b^m . It denotes a container; belong to booking b , in location m .

Suppose that y_{bim} and y_{gjk} have same value in descending list, we randomly select the booking among b^m, i^m, j^k, g^k booking numbers. This detail will be discussed with an example later.

The other rule is for movement type of containers that cause higher number of reshuffling. There are two movement types for containers:

1. To remove the containers to a location, if this location is empty during the process time of removed container. This movement is illustrated in Figure-9.

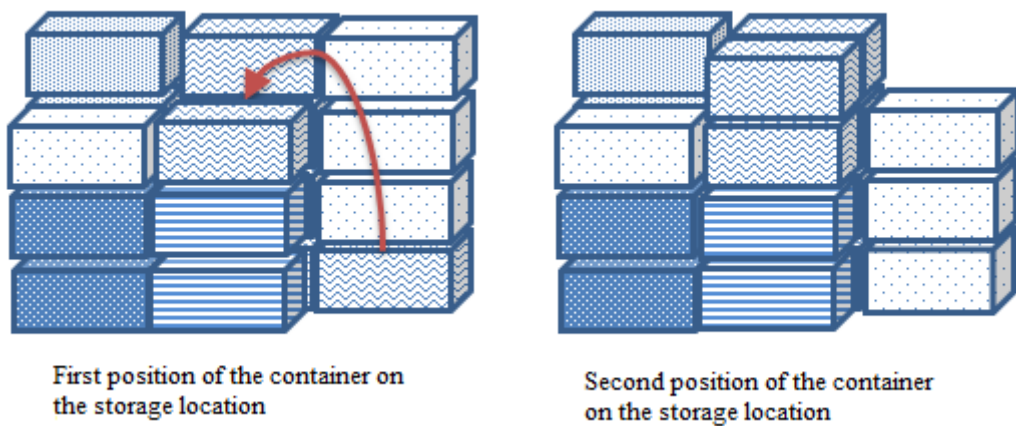


Figure 9 Relocate the container

2. To swap containers, if the process time of them is suitable to swap their location. Exchanged pair can be shown as (b^k, j^m) . It means that a container of booking b at k location and a container of booking j at m location will be swapped, if the arrival and departure times of swapped containers don't affect the other containers in these locations. It is not allowed swapping containers at different locations, if they belong to same booking number. This movement is illustrated in Figure-10.

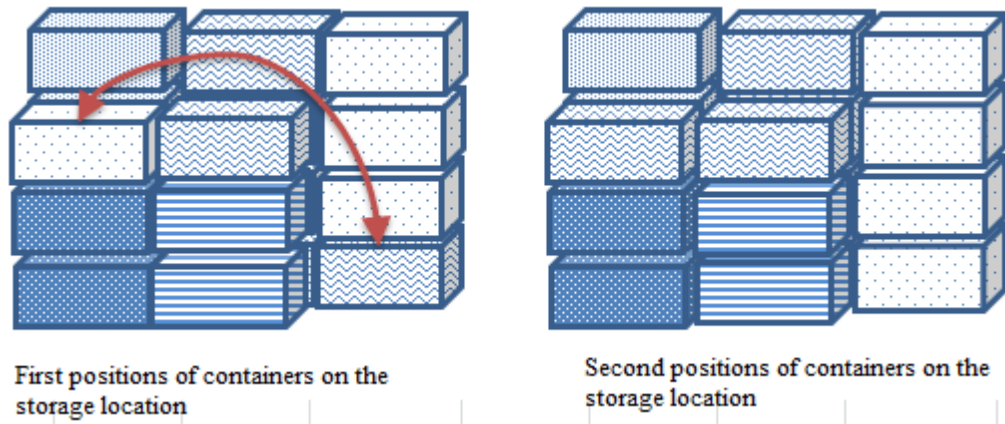


Figure 10 Swap the containers

We demonstrate the heuristic with simple example shown in Figure-11 and Figure-12. Suppose that there are four different bookings; b , i , j and g and two different locations; k and m in container storage area. According to bookings' arrival and departure times, time interval can be defined in the following Figure-11.

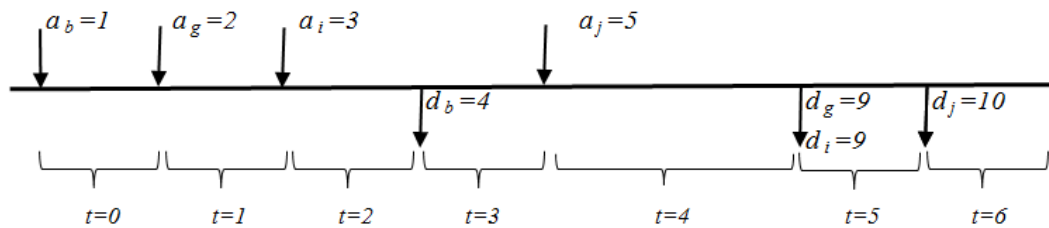


Figure 11 Time interval for sample case

In the following Figure-12 we suppose that incoming containers at each period are placed to the k or m location with respect to horizontal cost.

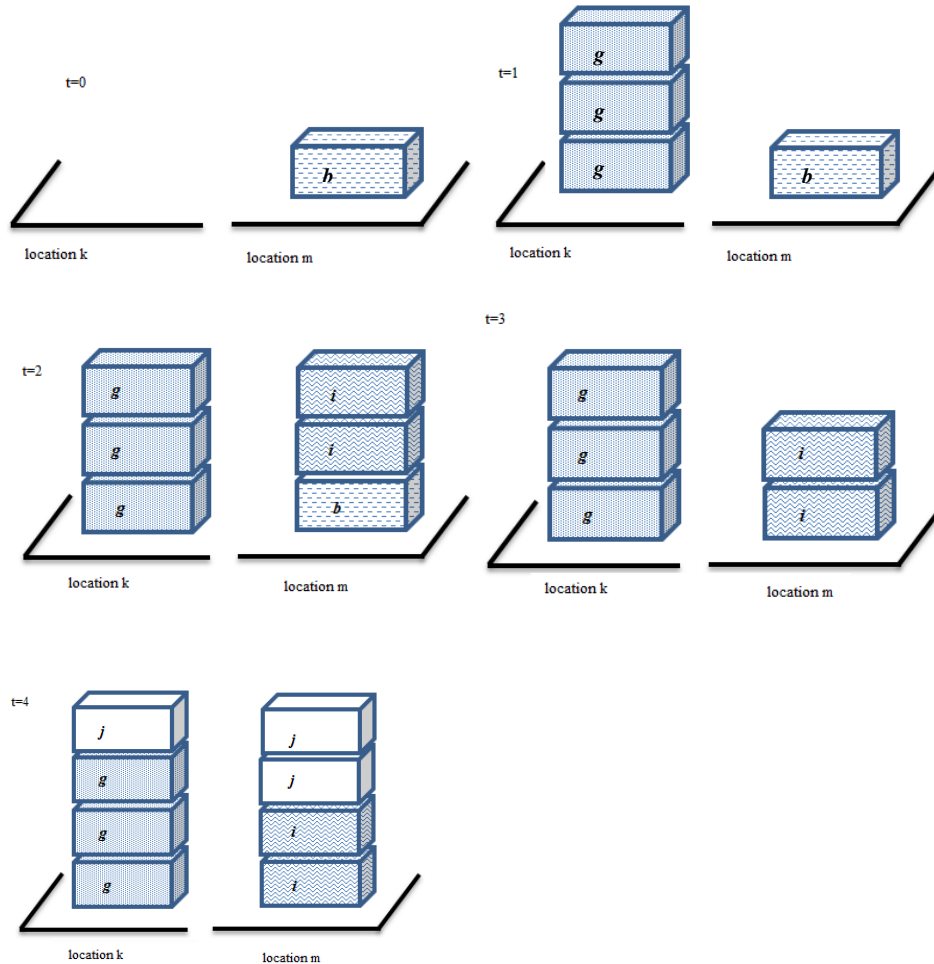


Figure 12 Placement at each period for small case

We further elaborate this process with a simple example. According to this placement, the total number of reshuffles for all locations is equal to 6. ($y_{bim}=2$, $y_{bjm}=0$, $y_{ijm}=2$ and $y_{gjk}= 2$). When the values of reshuffling are sorted in descending order, the list provides the maximum reshuffling values, which are y_{bim} , y_{ijm} and y_{gjk} . To reduce the amount of reshuffling, the booking can be randomly chosen between $b^m, j^m, j^k, i^m, g^m, g^k$. These booking numbers are associated with their location. If we start with booking j at location m , there are only three movements that can be considered; swapped with other container (j^m-j^k) and (j^m-g^k) or relocated to new location. Firstly, there is no empty location during the time between arrival time and departure time of booking j . For this reason, the container j cannot be relocated to

another location. Next, we forbid swapping the containers on locations m and k if they belong to the same booking. Besides, there are no possible swapping movements from m to k for booking j , because of capacity constraint for swapping pair j^m-g^k , so the number of containers of a location at each period must be at most 4.

If container of booking i at location m is randomly selected rather than booking j , there are again two pairs (i^m-g^k) and (i^m-j^k) , but no relocated movement. Suppose that this container is selected among containers which cause higher number of reshuffling. For swapping pairs (i^m-g^k) , there will be no improvement for the number of reshuffles or total cost, because the total amount of reshuffling is increased from 6 to 8 ($y_{bgm}=1, y_{bim}=1, y_{gjm}=2, y_{ijm}=2$ and $y_{gjk}=1, y_{ijk}=1$). When booking i at location m is selected by heuristics, there is another pair (i^m,j^k) . It can be also taken in consideration. When the booking i at location m and booking j at location k are swapped with each other, the total amount of reshuffling is decreased from 6 to 4 ($y_{bim}=1, y_{ijm}=3$). That is to say the initial feasible solution can be improved with reducing the cost of reshuffling.

These steps are explained to understand the selection and movements types of containers in our heuristic. If the movements is acceptable for feasible solution, new container will be selected from new placement of storage location. After the iteration, these feasible solutions are listed. As mentioned earlier, the heuristic terminates when the iteration reaches the iteration limit, determined by us.

Notations:

The notations which are used in H1 as like following:

- S_i : initial feasible solution
- S_t : solution found in iteration t
- S_{min} : minimum value of solution form $List_S$

- S_{H1_best} : best solution of Heuristic1.
 T : iteration index
 b_t^k : a container of booking number b in k location at t period
 $List_S$: feasible solution list
 $List_R$: number of reshuffling ordered in descending order. This list contains the reshuffles number and related booking number and location number.
 $iteLimit$: maximum number of iterations

All steps of heuristics method H1 are given below.

Step 1. Find the initial feasible solution S_i

Minimize horizontal cost at the storage location and obtain x_{bk}^* values

$$S_{H1_best} = S_i$$

no improvement = false

Step 2.

While($T < iteLimit$ and no improvement is true)

{

Find initial values and initial step for search algorithm

Step 2.1. For each $b \in B$, $i \in E(b)$ and $k \in K$

If ($x_{ik}^* > 0$ and $x_{bk}^* > 0$)

$$\tilde{y}_{bik} = x_{ik}^*$$

Step 2.2. Create $List_R$ and choose b_t^k booking number and its location which has highest value of $List_R$ at iteration t .

Step 3. Determine next feasible locations for selected container b_t^k

Step 3.1. Relocate movement

If (feasible location for b_t^k to relocate)

Relocate it,

Then **Step 3.4.**

Else

Go to **Step 3.2**.

Step 3.2. Swap movement

Randomly choose the other location and the containers is located in that (i^m)

If (Swapping pair (b^k, i^m) is feasible)

Swap them

Then **Step 3.4**.

Else

Go to **Step 3.3**.

Step 3.3.

If (no feasible movement for b^k)

Then it is placed to dummy location

Step 3.4. Try to find new location at the storage location for one container in the dummy location

Step 4. Calculate new feasible solutions S_t and add in the list of solutions $List_S$

}/end while

Step 5. Evaluate all solution in the $List_S$. Find minimum value S_{min}

$$S_{HI_best} = S_{min}$$

Step 6. Print best solutions S_{HI_best}

The initial feasible solution of H1 is based on minimizing horizontal cost. We develop the initial feasible solution with new procedure. This procedure can be used for adapting the real world problem. For this purpose, we place them into the storage location with respect to their arrival time, and attempt to hold containers of same customer or booking set together in same stacks. The horizontal transport cost is not very expensive, so this procedure can be used for a real world problem of a container

terminal. When there are empty places at the arrival time, the container is placed together with its container set without calculating the horizontal cost. It is an important point that the containers belonging to same booking stay together when they arrived to the storage location. Thus, the number of reshuffles may be decreased for some instances. The other important point is that new arrivals are attempted to be placed empty stacks. They aren't placed above the other bookings until all locations have at least one container.

Figure-13 illustrates the initial feasible solution of variant heuristic, named as H2.

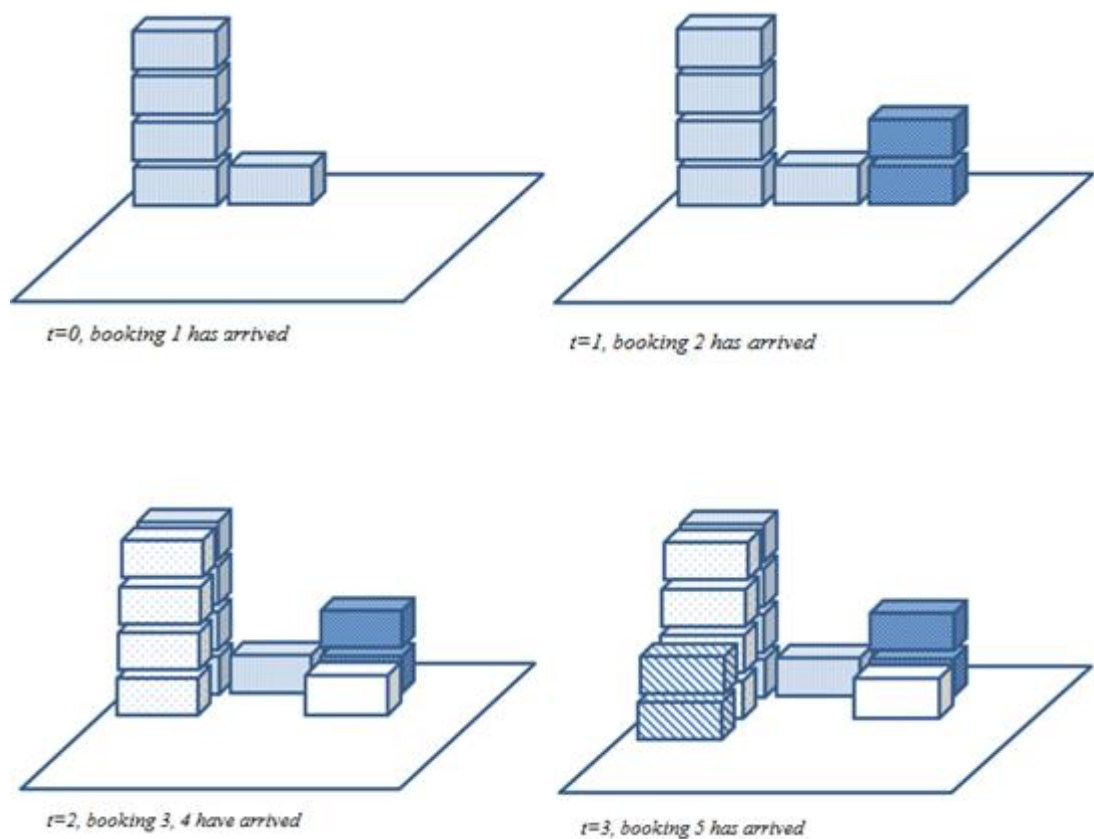
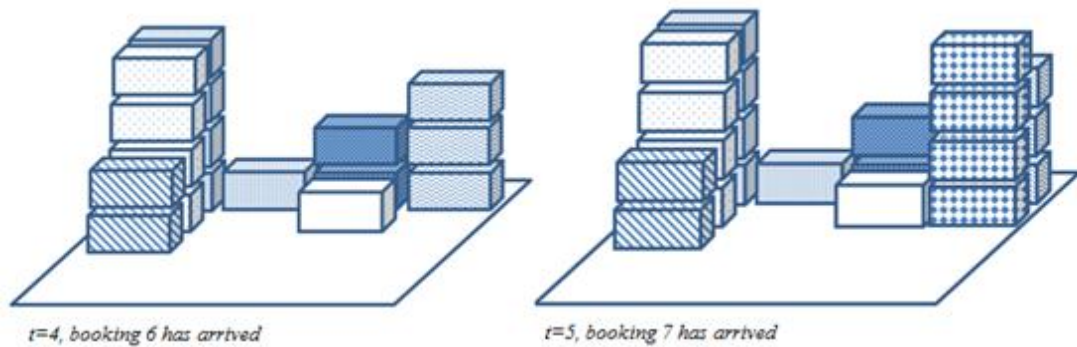


Figure 13 Placement behavior of initial condition



Cont. Figure 13 Placement behavior of initial condition

The second heuristic (H2) is a variation of H1. It is only different while finding the initial solution. An example of initial placement is presented in Figure-13. This example will be explained in next paragraph in detail.

When time period t is equal to 0, booking set 1 is arrived to the storage location and five bookings are placed in two locations. Four of them are located same location and last one is located in the other location because of capacity of a stack. At time period 1, second booking set is arrived with two containers. These containers are placed at a different location than the location of booking 1, even though there is empty location on the container of booking 1. This procedure is applied until all location has a container at least. After that, arrived containers can be placed on the other containers belonging different container sets. If there is not enough capacity at any time period, the containers can be placed dummy location. Besides, this method carry out to applied first come first served rule.

With this initial procedure, Step 1 of H1 algorithm is changed. In addition, H2 is occurred with this small variation. As mentioned above, we designed a new heuristic H2 for adapting CSP to the real world problem. By ignoring the horizontal cost, the container is placed together with its container set. The horizontal cost is not

very expensive in real world, so the container sets are placed with avoiding this cost. To placed containers with their container sets may decrease the number of reshuffles.

CHAPTER - 6

COMPUTATIONAL RESULTS

In this section, we report the results of our computational experiments. We first explain our experimental design, and then provide computational results.

6.1. Experiment Design

We have tested our model on various problem parameter settings. We have determined levels for the following parameters.

- Number of bookings, B
- Number of storage locations, K
- The initial occupancy of the storage area, D : For adapting the real world problem, we consider the initial condition of the storage yard and the planning horizon. Moreover, the initial condition is related with the containers in the storage area at the beginning of planning horizon. Normally, the container storage area cannot be empty. Before starting the planning horizon, there should be the containers in the storage area. The

amount of initial containers is set to 25 or 50% of the capacity of storage area.

We generated 10 instances for problem parameters which are the number of bookings, the number of locations and the occupancy of storage locations at the beginning of time period.

The settings are given in Table-1. Numbers of bookings are decided regarding the volume of largest container port in Turkey.

Parameters	Values used	Total
(Number of bookings- Number of locations)	(10-5), (10-10), (20-10), (20-20), (30-15), (30-30), (40-20), (40-40), (50-25), (50-50), (100-50), (100-100)	12
Occupancy	0.25 , 0.50	2

Table 1 Experimental Design

A series of computational experiments was carried out on a PC with 1.70 GHz Intel ® Core™ i5-3317U processor and 4GB RAM. These problems are generated with same stream for using common random numbers. All parameters decided by uniform distribution use different seeds. Ten instances are randomly generated for each combination of B , K and D . For each problem combination which is represented in Table-1, we generate 10 instances.

The input data for CSP is generated in C programming language on the same computer configuration according to the following procedure:

- Length of the planning horizon is equal to 20.
- The unit reshuffling cost is an important parameter in the model. R is taken constant.

- The number of containers in a booking, arrival and departure time of booking sets are determined by using this input generator. The number of bookings is uniformly distributed between 1 and 10. This means there are at most 10 containers in a booking.
- The arrival time for initial containers, that provide us initial condition for the problem, should equal to 0. In addition, the arrival time can be assigned from 0 to 25 for other container sets. The arrival time of each booking is generated uniformly during the planning horizon.
- The departure time is calculated by adding a process time to the arrival time. The process time is generated uniformly between 5 and 15.
- We generate the arrival berth and departure berth and coordinates of each berth, to calculate the horizontal distances.

The performance of the procedure is evaluated using various problem sizes and problem parameters, which are the number of bookings, the number of locations, and occupancy of the storage area at the beginning of the horizon, which is given as the fraction of the locations occupied. Initial occupancy can take the values 0.25 and 0.50. While the arrival time of initial bookings is set to 0, the departure time and the amount of containers of these booking set are created as random.

The objective function coefficient of the decision variable y_{bik} , R is another significant parameter for our problem since it can change the importance of reshuffling moves as compared to the transportation cost.

6.2. Exact Solutions

The optimal solution to the mathematical models developed in Chapter 4 was solved using GAMS 23.9.4 CPLEX solver on a PC with 1.70 GHz Intel ® Core ™

i5-3317U processor and 4GB RAM. The optimal solution were possible for all problem instances with $B=10$. Besides, CPLEX is able to solve some of problem instances of size $B = 20, 30, 40, 50$ and $K= 25, 50$. For having higher number of bookings, it was not possible even to find a starting feasible solution within the given time limit. It was not able to solve all instances for $B=100, K=50, 100$ and $B= 200, K= 100, 200$ within 1-h CPU time limit. For instances with more number of bookings, it was not possible to obtain solutions within reasonable times. During our preliminary runs, it is also observed that the optimal solution times are insensitive to the number of bookings, although the arrival and departure time of bookings which causes the reshuffling are directly related with optimal solution time. Therefore, the optimal solutions for all instances were obtained with time limit.

In the following Table-2 for CSPX-1 and Table-3 for CSPX-2, we represent the solution for CPLEX, for all parameters calculation, average CPU (in seconds) under Running Time and number of reshuffles in the solution. The running time is reported in seconds and time limit is set to as ten minutes for all instances set.

The last column in these tables represent relative gap between the best integer solution and the best bound. For all instances, we set relative gap as 0 which is named as $OptCR$ in CPLEX. The $OptCR$ option asks to CPLEX to stop when

$$(BP - BF) / (1.0e - 10 + |BF|) < OptCR \quad (6.1)$$

Where BF is objective value of current best integer solution and BP is best possible solution integer solution.

The results for optimal solution of CSPX-1 are presented in Table-2. The average CPU time is reported as 364.25 minutes.

We first investigate that instances belonging to five problem sets cannot be solved within time limit, so the numbers of reshuffles of these instance sets are higher than the other instances sets. Table-2 reports that 20% of the parameter combination is run till time limit.

B	K	D	Running time Optimal solution (seconds)	Number of reshuffles Optimal solution	Best Feasible Solution	Relative Gap
10	5	25	2.00	1	3558770.80	0.00
10	5	50	2.00	2.5	5181542.20	0.00
10	10	25	300.00	0	1019060.80	0.55
10	10	50	128.00	1	846023.30	0.67
20	10	25	401.80	3.1	4889366.40	2.28
20	10	50	499.30	6.2	4390109.90	4.49
20	20	25	182.50	0	12840.50	0.01
20	20	50	121.80	0.2	13817.00	2.65
30	15	25	420.60	3.7	5246894.20	0.21
30	15	50	480.00	5.1	4700607.00	0.15
30	30	25	181.70	0	20002.70	0.01
30	30	50	61.80	0	19692.40	0.00
40	20	25	540.20	5	5879172.90	0.15
40	20	50	480.40	7.4	5374284.20	0.54
40	40	25	420.60	0	25751.20	0.04
40	40	50	241.20	0	25928.30	0.00
50	25	25	600.00	14	6476067.20	0.87
50	25	50	600.00	19	5567915.40	1.47
50	50	25	300.00	2	32289.40	0.01
50	50	50	438.00	7	31131.50	1.26
100	50	25	600.00	74.4	7957947.80	4.21
100	50	50	600.00	82.5	10672603.20	2.59
100	100	25	600.00	50	185443.40	42.91
100	100	50	540.00	59.5	135166.00	41.02
Average			364.25	14.32	3010934.49	4.42

Table 2 CPLEX solution for CSPX-1

We observe that the number of reshuffles is higher when the initial occupancy is 50 percent. (Figure-14) Here we list the problem settings just based on B and K .

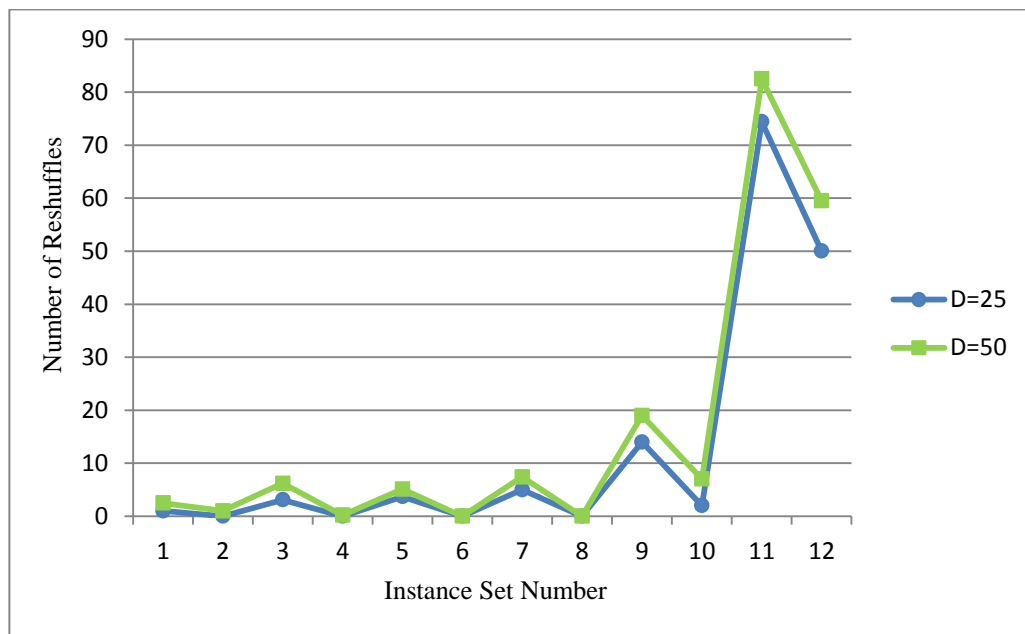


Figure 14 Number of reshuffles based on the occupancy for CSPX-1

In Figure-15, number of reshuffles is illustrated based on problem parameter. From this figure it can be seen that the number of reshuffles is changed regarding the number of location (K and $2K$) for same instances sets. When the number of location will be increased from K to $2K$, the amount of reshuffles will be dramatically decreased. For all instances sets, the amount of reshuffles for lower capacity of storage location is less than for higher capacity.

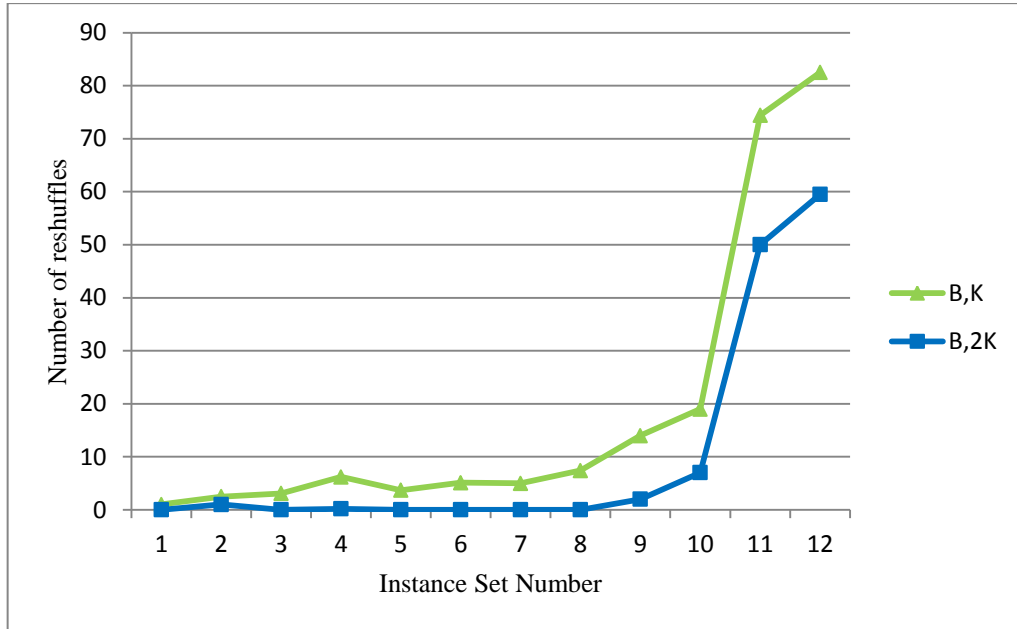


Figure 15 Number of Reshuffling based on Instances’ Parameters for CSPX-1

To have more flexible storage area and less reshuffles numbers at container terminal we must point out the number of location and occupancy of location. Thus initial occupancy of problem and location number is directly related with the number of reshuffles.

In Table-3, we summarize the average measures of the CPLEX solution for CSPX-2. The feasible solution is computed for CSPX-1 with average Relative Gap 4.42% at an average of 364.25 seconds of CPU time, while CSPX-2 is solved with Relative Gap 4.47% and 368.53 seconds. As can be seen form Table-3, the instances belonging five problem combinations cannot be solved within time limit.

B	K	D	Running time Optimal solution (seconds)	Number of reshuffles Optimal solution	Best Feasible Solution	Relative Gap
10	5	25	2.90	3.8	3048356.40	0.04
10	5	50	2.90	5.1	4880947.00	0.08
10	10	25	307.00	2.3	543842.40	0.61
10	10	50	128.90	2.2	242796.40	0.68
20	10	25	408.80	4.1	4050149.20	2.34
20	10	50	504.30	7.2	3451387.40	4.57
20	20	25	187.50	0	-1153347.20	0.09
20	20	50	128.80	1.2	-1550481.00	2.71
30	15	25	425.60	3.5	3739513.40	0.22
30	15	50	485.00	5.1	3012058.60	0.17
30	30	25	188.70	0	-2021375.40	0.03
30	30	50	66.80	0	-2267474.20	0.01
40	20	25	547.20	7	4447931.20	0.23
40	20	50	485.40	9.4	3464316.00	0.62
40	40	25	427.60	1	2662773.60	0.08
40	40	50	248.20	1	3103579.80	0.07
50	25	25	600.00	17	4817470.00	0.95
50	25	50	600.00	22	2353516.20	1.54
50	50	25	307.00	3	-3554338.00	0.05
50	50	50	445.00	10	-4115862.80	1.29
100	50	25	600.00	77.4	1549065.60	4.27
100	50	50	600.00	83.5	3798743.60	2.63
100	100	25	600.00	51	-7388342.20	42.94
100	100	50	547.00	60.5	-8763276.00	41.03
Average			368.53	15.72	764664.58	4.47

Table 3 CPLEX solution for CSPX-2

It is observed that the gaps of CSPX-1 and CSPX-2 are almost the same. The running times do not differ for both models. Also, to obtain the number of reshuffles of each instance Figure-16 is given in the following graph. It shows that the inventory balance equation on CSPX-2 does not add much complexity to the problem, whereas the number of reshuffles in the solution of CSPX-2 is higher than the number of reshuffles in the solution of CSPX-1.

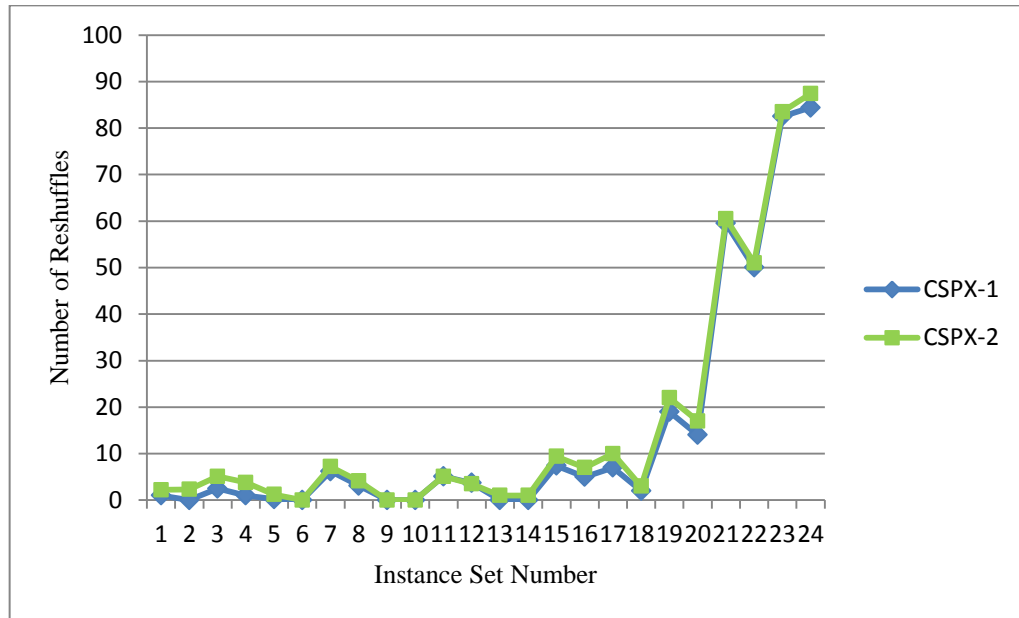


Figure 16 Number of reshuffles for each problem set

6.3. Computational Results

6.3.1. Results on Lagrangean Relaxation

In this section, we report the results of our computational experiments with Lagrangean Relaxation methods. The preliminary problems are solved with different performance parameters. The limit on the number of iterations for updating value of the step size parameter $noImpLimit$, and the limit on the total number of iterations $iteLimit$ are significant performance parameters to evaluate the Subgradient Optimization. The objective function coefficient of the decision variable y_{bik} , R is another significant parameter for our problem since it changes the importance of reshuffling moves as compared to the transportation cost. The different values of these parameters are used in our experiments. We evaluate the performances of lagrangean methods with using these parameters.

To test the effect of the *noImpLimit* on the best lower bound value, the value of *noImpLimit* is decreased from 10 to 5 and new runs are made. Decreasing the *noImpLimit* from 10 to 5 does not help improving the lower bound value after the first iteration.

On the other hand, the upper bound values increase with the smaller *noImpLimit*. As a result, this causes an increase in gap values. This parameter cannot positively affect the value of lower and upper bound for all instances. We observe that the initial lower bound values obtained in the first iteration stay the same throughout the remaining iterations. The behavior of the lower bound value would be investigated in the further below.

As the number of iterations increase, the *bestUB* values for Lagrangean methods become better for some instances. Regarding to our experiments, we decide to run the LR methods for 50 and 100 iterations.

This section describes our computational experiment designed to evaluate the performance of our heuristic algorithms. All instances are solved by using CPLEX 12.2 via Concert Technology on the same computer configuration. The tables given in this chapter represent the summary results of LRTC, LRSC and LRSLM with the Subgradient Optimization procedure for all instances. In particular, we report the percentage of lower bound and upper bound improvement that represent the improvement on first lower/upper bound (LB^1 , UB^1) and best lower/upper bound (*bestLB*, *bestUB*). Moreover, Gap 1 value that represents the percentage gaps between the best lower bound and the best upper bound. They are computed as below.

$$\text{Gap 1} = \frac{\text{bestUB} - \text{bestLB}}{\text{bestUB}} \times 100 \quad (6.2)$$

The utilization represents the usage of storage area rather than dummy location. In each period, the utilization is computed the ratio of total capacity in percentage. We also report the running times in seconds, but we give the time limit as ten minutes. The number of reshuffles in the upper bounding scheme is obtained in the table. We report the average results for each number of bookings (B), number of locations (K), and occupancy of the storage area (D), for each lagrangean relaxation method solution (LRTC, LRSC, LRSLM) and for given number of iterations.

Table-4 depicts the computational results the summary of LRTC for CSPX-1 model for 50 iterations, respectively. We observe that the initial lower bound values obtained in the first iteration stay the same throughout the remaining iterations. This result is investigated further below. While the lower bound remains the same for all instances, the upper bound improves through the iterations. LRTC approach cannot generate a sufficiently powerful lower bound.

The usage of area directly affects the number of reshuffles. In other words, the number of reshuffles increases, when utilization for lower capacity of storage location increases.

Upper bound for this lagrangean approaches is based on vertical cost. If the solutions of subproblem causes lower reshuffling number, better improvement on the upper bound is expected. If the Table-4 examined in detail, it appears that upper bound improvement is directly related with number of reshuffling for all categories.

B	K	D	% LB improvement	% UB improvement	Gap1	% Utilization	Running time (seconds)	Number of reshuffles
10	5	25	0.00	14.62	5.09	69.42	1.00	4.66
10	5	50	0.00	13.74	6.68	69.97	1.00	5.70
10	10	25	0.00	24.08	6.65	53.25	1.00	3.10
10	10	50	0.00	24.47	12.51	63.14	1.00	3.30
20	10	25	0.00	10.38	16.48	72.55	1.80	7.10
20	10	50	0.00	4.40	10.03	77.92	1.95	13.20
20	20	25	0.00	53.22	74.25	48.42	2.90	4.90
20	20	50	0.00	47.57	74.74	48.25	3.35	6.70
30	15	25	0.00	1.12	2.56	76.31	3.50	16.10
30	15	50	0.00	1.26	2.97	83.05	3.80	24.40
30	30	25	0.00	46.50	76.87	49.39	6.40	11.30
30	30	50	0.00	45.52	75.69	54.20	8.50	11.70
40	20	25	0.00	5.59	10.02	78.13	7.10	30.40
40	20	50	0.00	3.86	16.81	83.67	7.70	38.80
40	40	25	0.00	31.71	63.90	48.01	15.90	18.60
40	40	50	0.00	38.38	66.19	54.85	18.30	21.80
50	25	25	0.00	2.70	4.67	81.63	12.70	43.40
50	25	50	0.00	7.27	14.73	88.21	14.00	54.40
50	50	25	0.00	36.50	62.13	49.24	27.30	21.70
50	50	50	0.00	28.84	72.76	57.08	34.50	40.30
100	50	25	0.00	2.27	7.03	84.21	113.30	132.40
100	50	50	0.00	2.27	7.03	84.21	129.20	157.30
100	100	25	0.00	27.77	72.74	50.07	229.00	70.20
100	100	50	0.00	27.55	77.37	59.09	643.40	94.80
Average			0.00	20.90	35.00	66.01	53.69	34.84

Table 4 Summary of LRTC for CSPX-1, number of iterations = 50

The results for the same setting for 100 iterations are given in the Table-5. Lower bound values do not improve with the increasing number of iterations. At the moment, it can be observed that increasing the number of iterations does not provide better UB value, as can be seen from the Table-5.

B	K	D	% LB improvement	% UB improvement	Gap1	% Utilization	Running time (seconds)	Number of reshuffles
10	5	25	0.00	14.62	5.09	69.42	1.56	4.66
10	5	50	0.00	13.74	6.68	69.97	1.70	5.70
10	10	25	0.00	24.08	6.65	53.25	2.10	3.10
10	10	50	0.00	24.47	12.51	63.14	2.20	3.30
20	10	25	0.00	11.38	17.48	72.55	3.60	7.10
20	10	50	0.00	4.42	10.03	77.92	3.90	13.20
20	20	25	0.00	54.22	75.25	48.42	5.80	4.90
20	20	50	0.00	48.57	75.74	48.25	6.70	6.70
30	15	25	0.00	1.38	2.56	76.23	6.90	11.40
30	15	50	0.00	1.33	2.95	83.05	7.40	23.40
30	30	25	0.00	48.94	76.87	49.39	13.40	10.30
30	30	50	0.00	46.50	75.69	54.20	17.00	11.30
40	20	25	0.00	5.70	9.91	78.13	14.00	29.50
40	20	50	0.00	3.88	16.80	83.67	15.50	38.50
40	40	25	0.00	31.71	63.90	48.01	29.40	18.60
40	40	50	0.00	39.00	65.95	54.85	38.50	21.40
50	25	25	0.00	2.70	4.67	81.63	25.40	43.40
50	25	50	0.00	7.27	14.73	88.21	28.00	54.40
50	50	25	0.00	36.50	62.13	49.24	54.60	21.70
50	50	50	0.00	28.84	72.76	57.08	73.00	40.30
100	50	25	0.00	2.27	7.03	84.21	225.30	132.40
100	50	50	0.00	1.87	5.66	90.48	273.80	157.30
100	100	25	0.00	27.77	72.74	50.07	480.30	70.20
100	100	50	0.00	27.55	77.37	59.09	113.40	94.80
Average			0.00	21.20	35.05	66.27	60.14	34.48

Table 5 Summary of LRTC for CSPX-1, number of iterations = 100

Table-6 and Table-7 represent that the computational results the summary of LRSC for CSPX-1 model for 50 and 100 iterations. The average improvement on first upper bound value is seen as 20.90% for LRTC in Table-4, while the improvement on upper bounding procedure through iteration is at average 18.18% in Table-6. While the upper bounding procedure for LRSC yields average deviations of 37.30 % from the lower bound, the average deviation from the lower bound for LRTC approach is 35.00%. When we observe UB improvement of upper bound and Gap1, there is no significant difference between LRSC and LRTC approaches. Our heuristics seem not to generate quite powerful upper and lower bounds.

The total CPU time used by LRSC to solve the 10 problems in one category is 4 times of that used by LRTC for some instances.

While the number of reshuffles of LRSC from Table-6 is at an average 32.96, the number of reshuffles of LRTC is seen as 34.84 in Table-4. It can be seen that LRTC has an advantage over LRSC regarding the number of reshuffles.

B	K	D	% LB improvement	% UB improvement	Gap1	% utilization	Running time (seconds)	Number of reshuffles
10	5	25	0.00	4.56	4.90	69.79	1.90	4.30
10	5	50	0.00	4.78	16.03	69.97	2.50	6.20
10	10	25	0.00	21.98	20.65	53.34	2.50	3.10
10	10	50	0.00	12.43	32.33	63.14	3.10	3.80
20	10	25	0.00	8.87	14.26	72.24	11.00	7.80
20	10	50	0.00	4.57	8.78	77.92	6.10	13.10
20	20	25	0.00	40.16	60.15	48.42	15.90	6.90
20	20	50	0.00	36.99	65.42	60.31	17.60	8.90
30	15	25	0.00	1.24	2.64	76.31	12.20	1.24
30	15	50	0.00	1.47	3.10	83.07	15.00	1.47
30	30	25	0.00	42.97	78.08	49.39	25.60	11.50
30	30	50	0.00	40.26	78.68	54.20	28.90	11.90
40	20	25	0.00	6.18	12.76	78.14	33.20	29.50
40	20	50	0.00	3.81	9.63	83.68	35.10	39.50
40	40	25	0.00	42.88	78.43	48.01	63.50	14.00
40	40	50	0.00	33.21	81.67	54.85	58.80	22.60
50	25	25	0.00	2.96	4.56	81.63	46.70	41.60
50	25	50	0.00	8.57	14.38	88.21	142.30	55.70
50	50	25	0.00	35.88	64.34	49.24	103.30	20.80
50	50	50	0.00	25.51	76.37	57.08	138.40	35.30
100	50	25	0.00	2.40	7.08	168.42	311.30	130.60
100	50	50	0.00	1.78	5.96	180.86	323.70	160.20
100	100	25	0.00	27.68	73.92	200.29	660.11	68.00
100	100	50	0.00	25.22	81.12	236.35	908.80	93.00
Average			0.00	18.18	37.30	87.70	123.65	32.96

Table 6 Summary of LRSC for CSPX-1, number of iterations = 50

To get a visual impression of how each Lagrangean method performs over the number of reshuffles, Figure-17 is depicted. It is seen that there is no significant difference between the amounts of reshuffling for all instances.

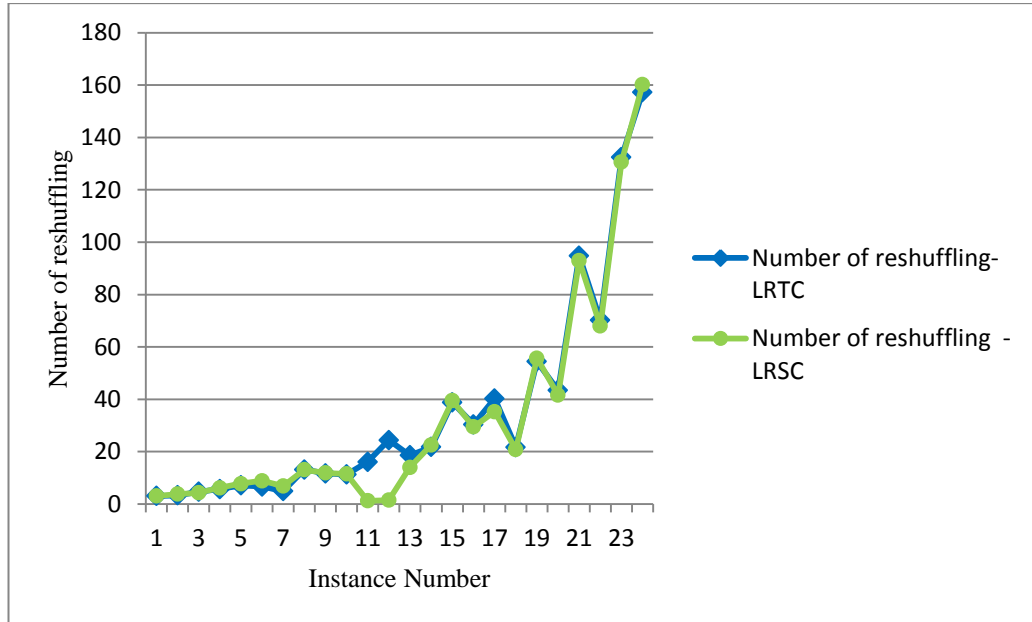


Figure 17 Number of Reshuffles Comparison for LRTC and LRSC

LRSC cannot provide better solution in order to reduce number of reshuffles. The average running time for each instances are denoted for Lagrangean approaches LRTC and LRSC in Figure-18. It is clearly seen that LRSC takes more time to compute solution rather than LRTC approach.

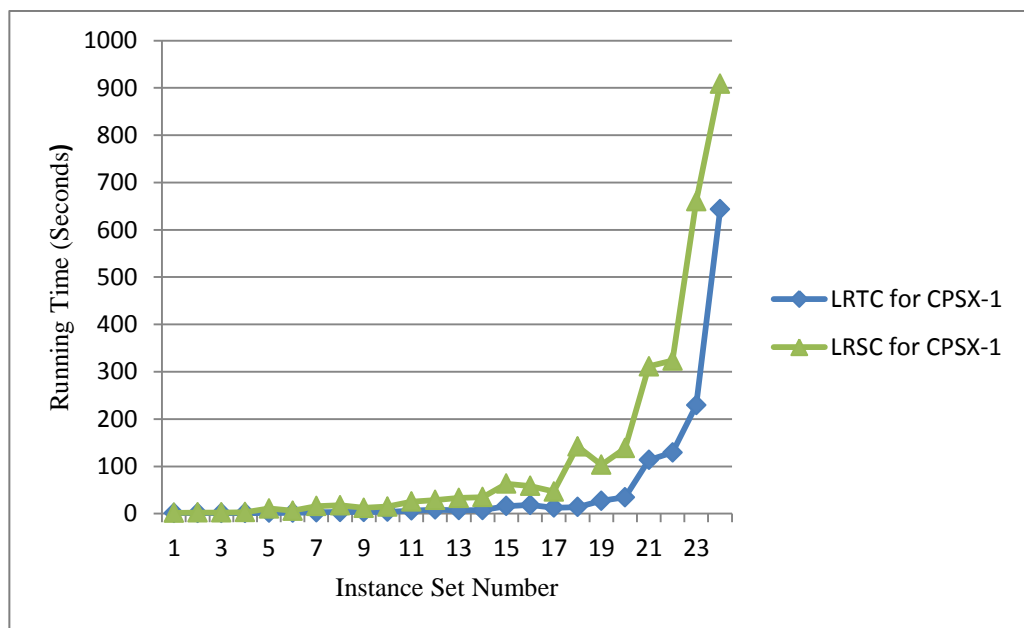


Figure 18 Running Time Comparison for LRTC and LRSC

As mentioned before, for the same setting of Table-6 is computed for 100 iterations and the results are given in the Table-7. There is no improvement when number of iteration is increased.

B	K	D	% LB improvement	% UB improvement	Gap1	% utilization	Running time (seconds)	Number of reshuffles
10	5	25	0.00	4.56	7.67	69.79	3.80	4.30
10	5	50	0.00	4.78	17.64	69.97	6.00	6.20
10	10	25	0.00	21.98	21.98	53.34	6.00	3.10
10	10	50	0.00	12.43	12.43	63.14	6.20	3.80
20	10	25	0.00	8.87	14.26	72.24	20.00	7.80
20	10	50	0.00	4.57	8.78	77.92	11.20	13.10
20	20	25	0.00	40.16	60.15	48.42	30.40	6.90
20	20	50	0.00	36.99	65.42	60.31	32.00	8.90
30	15	25	0.00	1.24	2.64	76.31	22.50	1.24
30	15	50	0.00	1.47	3.10	83.07	24.00	1.47
30	30	25	0.00	42.97	78.08	49.39	49.00	11.50
30	30	50	0.00	40.26	78.68	54.20	54.50	11.90
40	20	25	0.00	6.18	12.76	78.14	65.40	29.50
40	20	50	0.00	3.81	9.63	83.68	70.00	39.50
40	40	25	0.00	42.88	78.43	48.01	117.00	14.00
40	40	50	0.00	33.21	81.67	54.85	114.30	22.60
50	25	25	0.00	2.96	4.56	81.63	93.40	41.60
50	25	50	0.00	8.57	14.38	88.21	280.60	55.70
50	50	25	0.00	35.88	64.34	49.24	200.60	20.80
50	50	50	0.00	25.51	76.37	57.08	270.80	35.30
100	50	25	0.00	2.40	7.08	168.42	620.60	130.60
100	50	50	0.00	1.78	5.96	180.86	640.40	160.20
100	100	25	0.00	27.68	73.92	200.29	980.23	68.00
100	100	50	0.00	25.22	81.12	236.35	1298.60	93.00
Average			0.00	18.18	36.71	87.70	209.06	32.96

Table 7 Summary of LRSC for CSPX-1, Number of iterations = 100

The computational results the summary of LRSLM for CSPX-1 model for 50 and 100 iterations are depicted in Table-8 and Table-9. There is a significant difference between LRSC and LRSLM approaches for percentage of UB improvement, if the improvements on upper bound for these approaches are compared. The average improvement on first upper bounding procedure is seen as 20.90% for LRSC in Table-4, while it is at average 9.47% in Table-8. It can be seen

that the percentage gap between best lower and upper bound is 37.15% in Table-8, while the percentage gap is reported as 35% for LRTC approach in Table-4.

The number of reshuffles of LRTC is seen as 34.84 in Table-4, while the number of reshuffles of LRSC from Table-8 is at an average 41.13. Therefore we can say that LRTC also has an advantage over LRSC because of the number of reshuffles. Moreover, it can be easily noticed that CPU time used by LRTC is better than the time used by LRSLM for all instances. Hence, we can say that LRTC outperforms to LRSLM.

B	K	D	% LB improvement	% UB improvement	Gap1	% utilization	Running time (seconds)	Number of reshuffles
10	5	25	0.00	0.01	7.54	69.79	1.00	7.80
10	5	50	0.00	0.00	17.24	69.97	1.00	9.30
10	10	25	0.00	3.22	28.74	53.25	1.50	8.50
10	10	50	0.00	0.21	33.38	63.14	1.50	7.80
20	10	25	0.00	2.42	16.81	72.55	1.65	15.30
20	10	50	0.00	0.09	9.94	77.92	1.75	21.30
20	20	25	0.00	15.72	69.01	48.42	2.45	12.70
20	20	50	0.00	18.71	70.35	60.31	2.95	13.80
30	15	25	0.00	0.40	2.17	76.28	3.00	22.60
30	15	50	0.00	0.49	2.48	83.03	3.50	34.20
30	30	25	0.00	24.88	67.17	49.39	5.60	18.70
30	30	50	0.00	12.34	72.30	54.20	7.00	22.70
40	20	25	0.00	2.94	10.86	78.12	6.00	33.10
40	20	50	0.00	3.83	7.23	83.67	6.00	39.20
40	40	25	0.00	21.32	70.17	48.01	12.00	25.40
40	40	50	0.00	18.71	74.58	54.85	15.35	32.10
50	25	25	0.00	1.90	5.43	81.67	11.20	54.10
50	25	50	0.00	6.06	19.38	88.78	12.13	59.88
50	50	25	0.00	26.10	64.96	49.24	21.55	25.00
50	50	50	0.00	22.76	74.52	57.08	28.35	38.20
100	50	25	0.00	1.86	7.42	84.21	75.00	136.10
100	50	50	0.00	1.73	5.80	90.42	90.00	165.80
100	100	25	0.00	22.12	74.62	50.07	145.00	77.30
100	100	50	0.00	19.38	79.42	59.09	190.00	106.20
Average			0,00	9,47	37,15	66,81	59,36	41,13

Table 8 Summary of LRSLM for CSPX-1. Number of iterations = 50

For the same setting of Table-8 is computed for 100 iterations and the results are given in the Table-9.

B	K	D	% LB improvement	% UB improvement	Gap1	% utilization	Running time (seconds)	Number of reshuffles
10	5	25	0.00	0.01	7.54	69.79	1.60	7.80
10	5	50	0.00	0.00	17.24	69.97	1.70	9.30
10	10	25	0.00	3.22	28.74	53.25	2.20	8.50
10	10	50	0.00	0.21	33.38	63.14	2.40	7.80
20	10	25	0.00	2.42	16.81	72.55	3.30	15.30
20	10	50	0.00	0.09	9.94	77.92	3.50	21.30
20	20	25	0.00	15.72	69.01	48.42	4.90	12.70
20	20	50	0.00	18.71	70.35	60.31	5.90	13.80
30	15	25	0.00	0.40	2.17	76.28	6.10	22.60
30	15	50	0.00	0.49	2.48	83.03	7.20	34.20
30	30	25	0.00	24.88	67.17	49.39	11.20	18.70
30	30	50	0.00	12.34	72.30	54.20	14.70	22.70
40	20	25	0.00	2.94	10.86	78.12	13.20	33.10
40	20	50	0.00	3.83	7.23	83.67	12.90	39.20
40	40	25	0.00	21.32	70.17	48.01	24.60	25.40
40	40	50	0.00	18.71	74.58	54.85	30.70	32.10
50	25	25	0.00	1.90	5.43	81.67	22.40	54.10
50	25	50	0.00	6.06	19.38	88.78	24.25	59.88
50	50	25	0.00	26.10	64.96	49.24	43.10	25.00
50	50	50	0.00	22.76	74.52	57.08	56.70	38.20
100	50	25	0.00	1.86	7.42	84.21	171.70	136.10
100	50	50	0.00	1.73	5.80	90.42	197.00	165.80
100	100	25	0.00	22.12	74.62	50.07	331.80	77.30
100	100	50	0.00	19.38	79.42	59.09	431.70	106.20
Average			0.00	9.47	37.15	66.81	26.89	41.13

Table 9 Summary of LRSLM for CSPX-1. Number of iterations = 100

Regarding previous experiments for CSPX-1, we decide to observe the computational results of LRTC for CSPX-2. The initial lower bound values cannot change throughout all iterations. While the lower bound remains the same for all instances, the upper bound improves through the iterations. Improvement on upper bound is affected regarding to the number of reshuffles.

B	K	D	% LB improvement	% UB improvement	Gap1	% Utilization	Running time (seconds)	Number of reshuffles
10	5	25	0.00	1.00	1.42	69.79	2.00	4.90
10	5	50	0.00	1.60	0.71	70.18	1.00	5.90
10	10	25	0.00	5.97	3.60	53.36	2.10	4.10
10	10	50	0.00	45.74	4.73	63.67	2.40	3.60
20	10	25	0.00	2.66	1.56	72.68	2.55	7.30
20	10	50	0.00	2.43	1.67	78.12	2.65	12.60
20	20	25	0.00	2.69	0.99	48.42	4.15	4.80
20	20	50	0.00	2.52	1.04	60.31	5.45	6.70
30	15	25	0.00	5.32	5.75	76.60	5.15	16.00
30	15	50	0.00	4.50	3.87	83.15	5.45	24.10
30	30	25	0.00	2.61	1.18	50.18	8.30	9.20
30	30	50	0.00	1.84	1.32	54.20	10.65	12.10
40	20	25	0.00	3.49	5.64	78.46	9.45	28.50
40	20	50	0.00	3.23	8.44	84.33	10.55	41.40
40	40	25	0.00	1.60	1.66	48.01	21.05	17.60
40	40	50	0.00	1.74	1.63	54.85	23.15	20.60
50	25	25	0.00	23.25	11.47	82.16	16.00	44.80
50	25	50	0.00	2.66	5.19	88.62	20.20	59.30
50	50	25	0.00	1.50	1.61	49.24	31.55	22.40
50	50	50	0.00	1.27	2.13	57.08	41.90	34.30
100	50	25	0.00	18.84	51.72	84.44	120.00	136.70
100	50	50	0.00	10.44	19.35	90.86	134.35	159.30
100	100	25	0.00	1.43	2.35	50.07	240.65	68.50
100	100	50	0.00	1.37	2.63	59.09	357.85	91.70
Average			0.00	6.24	5.90	66.99	44.94	34.85

Table 10 Summary of LRTC for CSPX-2. Number of iterations = 50

During planning horizon, the usage of storage area for CSPX-2 is higher than utilization ratio for CSPX-1. Because that we force arrival bookings to place in the storage are regarding to structure of CSPX-2. The results of LRTC for CSPX-2 are shown in Table-10 and Table-11. It can be seen that the percentage of utilization is higher than utilization for CPSX-1 for some instances sets. The utilization values for all instances set are shown in Figure-19. There is a significant difference between the utilization of CSPX-1 and CSPX-2 only for one instances set. We can say that there is no significant difference between utilization of these two models for other instances sets, but it can be seen that CSPX-2 always guarantee more usage of storage area rather than dummy location. For big problem instances, it will be more suitable to provide better usage of storage area.

B	K	D	% LB improvement	% UB improvement	Gap1	% Utilization	Running time (seconds)	Number of reshuffles
10	5	25	0.00	1.00	1.42	69.79	3.00	4.90
10	5	50	0.00	1.60	0.71	70.18	2.90	5.90
10	10	25	0.00	5.97	3.60	53.36	3.20	4.10
10	10	50	0.00	45.74	4.73	63.67	3.10	3.60
20	10	25	0.00	2.66	1.56	72.68	5.10	7.30
20	10	50	0.00	2.43	1.67	78.12	5.30	12.60
20	20	25	0.00	2.69	0.99	48.42	8.30	4.80
20	20	50	0.00	2.52	1.04	60.31	10.90	6.70
30	15	25	0.00	5.32	5.75	76.60	10.30	16.00
30	15	50	0.00	4.50	3.87	83.15	10.90	24.10
30	30	25	0.00	2.61	1.18	50.18	16.60	9.20
30	30	50	0.00	1.84	1.32	54.20	21.30	12.10
40	20	25	0.00	3.49	5.64	78.46	18.90	28.50
40	20	50	0.00	3.23	8.44	84.33	21.10	41.40
40	40	25	0.00	1.60	1.66	48.01	42.10	17.60
40	40	50	0.00	1.74	1.63	54.85	46.30	20.60
50	25	25	0.00	23.25	11.47	82.16	32.00	44.80
50	25	50	0.00	2.66	5.19	88.62	40.40	59.30
50	50	25	0.00	1.50	1.61	49.24	63.10	22.40
50	50	50	0.00	1.27	2.13	57.08	83.80	34.30
100	50	25	0.00	18.84	51.72	84.44	255.70	136.70
100	50	50	0.00	10.44	19.35	90.86	284.70	159.30
100	100	25	0.00	1.43	2.35	50.07	519.30	68.50
100	100	50	0.00	1.37	2.63	59.09	715.70	91.70
Average			0.00	6.24	5.90	66.99	92.67	34.85

Table 11 Summary of LRTC for CSPX-2. Number of iterations =100

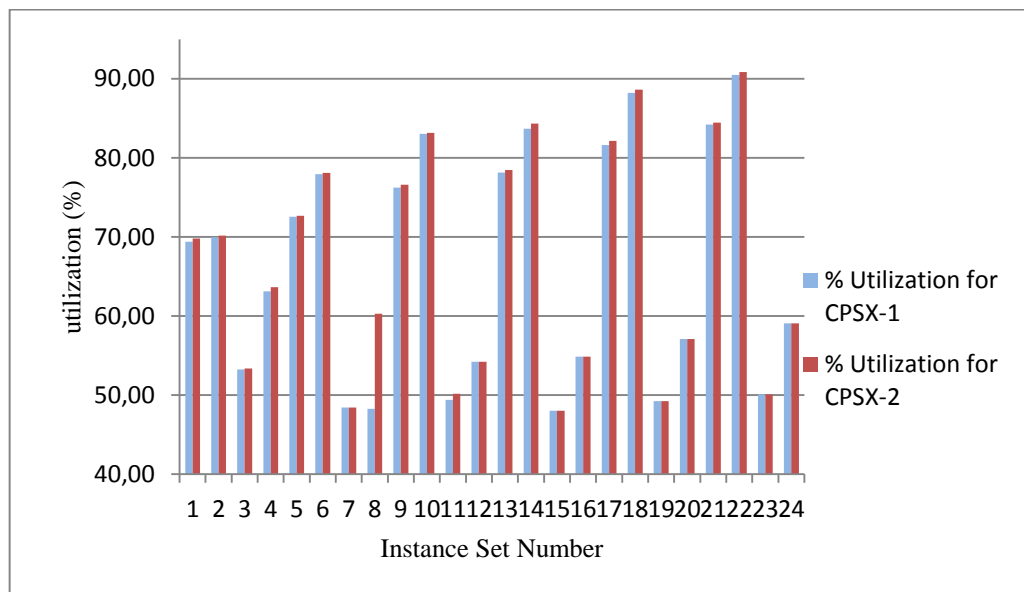


Figure 19 Comparison of Utilization Values for CSPX-1 and CPSX-2

For CSPX-1, we show the cost value of upper bound heuristic by given LR in Table-12 for 50 iterations and in Table-13 for 100 iterations. The upper bound vertical cost denotes the average reshuffling cost of best feasible solutions for each instances set.

In most case, there is no difference for running 50 iterations or 100 iterations. We found the same results for cost value of the best upper bound. Thus, we observe that upper bound value cannot improve through the number of iteration.

B	K	D	UB number of reshuffles	Utilization	UB cost of reshuffles	UB vertical cost	Vertical and Horizontal Cost
10	5	25	4.66	69.42	11650.00	4566.44	16216.44
10	5	50	5.70	69.97	14250.00	4047.60	18297.60
10	10	25	3.10	53.25	7750.00	6500.80	14250.80
10	10	50	3.30	63.14	8250.00	7027.00	15277.00
20	10	25	7.10	72.55	17750.00	9146.40	26896.40
20	10	50	13.20	77.92	33000.00	9551.40	42551.40
20	20	25	4.90	48.42	12250.00	12852.80	25102.80
20	20	50	6.70	48.25	16750.00	13322.80	30072.80
30	15	25	16.10	76.31	40250.00	15133.60	55383.60
30	15	50	24.40	83.05	61000.00	15284.60	76284.60
30	30	25	11.30	49.39	28250.00	27860.60	56110.60
30	30	50	11.70	54.20	29250.00	19696.40	48946.40
40	20	25	30.40	78.13	76000.00	21545.60	97545.60
40	20	50	38.80	83.67	97000.00	20871.80	117871.80
40	40	25	18.60	48.01	46500.00	25133.60	71633.60
40	40	50	21.80	54.85	54500.00	25921.20	80421.20
50	25	25	43.40	81.63	108500.00	28199.00	136699.00
50	25	50	54.40	88.21	136000.00	27827.60	163827.60
50	50	25	21.70	49.24	54250.00	31764.00	86014.00
50	50	50	40.30	57.08	100750.00	31137.80	131887.80
100	50	25	132.40	84.21	331000.00	57117.80	388117.80
100	50	50	157.30	84.21	393250.00	56780.80	450030.80
100	100	25	70.20	50.07	175500.00	64698.80	240198.80
100	100	50	94.80	59.09	237000.00	56780.80	293780.80
Average			34.84	66.01	87110.42	24698.72	111809.14

Table 12 Cost Value given by LRTC for CSPX-1. Number of iterations =50

Table-13 denotes that the number of reshuffles, the utilization, vertical and horizontal cost given by LRTC for 100 iterations.

B	K	D	UB number of reshuffles	Utilization	UB cost of reshuffles	UB vertical cost	Vertical and Horizontal Cost
10	5	25	4.66	69.42	11650.00	4566.44	16216.44
10	5	50	5.70	69.97	14250.00	4047.60	18297.60
10	10	25	3.10	53.25	7750.00	6500.80	14250.80
10	10	50	3.30	63.14	8250.00	7027.00	15277.00
20	10	25	7.10	72.55	17750.00	9146.40	26896.40
20	10	50	13.20	77.92	33000.00	9551.40	42551.40
20	20	25	4.90	48.42	12250.00	12852.80	25102.80
20	20	50	6.70	48.25	16750.00	13322.80	30072.80
30	15	25	11.40	76.23	28500.00	15136.00	43636.00
30	15	50	23.40	83.05	58500.00	15284.60	73784.60
30	30	25	10.30	49.39	25750.00	27862.00	53612.00
30	30	50	11.30	54.20	28250.00	19689.80	47939.80
40	20	25	29.50	78.13	73750.00	21545.80	95295.80
40	20	50	38.50	83.67	96250.00	20867.80	117117.80
40	40	25	18.60	48.01	46500.00	25133.60	71633.60
40	40	50	21.40	54.85	53500.00	25921.20	79421.20
50	25	25	43.40	81.63	108500.00	28199.00	136699.00
50	25	50	54.40	88.21	136000.00	27827.60	163827.60
50	50	25	21.70	49.24	54250.00	31764.00	86014.00
50	50	50	40.30	57.08	100750.00	31137.80	131887.80
100	50	25	132.40	84.21	331000.00	57117.80	388117.80
100	50	50	157.30	90.48	393250.00	56780.80	450030.80
100	100	25	70.20	50.07	175500.00	64698.80	240198.80
100	100	50	94.80	59.09	237000.00	56780.80	293780.80
Average			34.48	66.27	86204.17	24698.44	110902.61

Table 13 Cost Value given by LRTC for CSPX-1. Number of iterations =100

In Tables-14 and Table-15, we observe the same conclusion like above table for CSPX-1. CSPX-2 is solved by LRTC and the following tables represent the value of all cost given by upper bound approach of LR for 50 iterations and for 100 iterations. Generally, there is no improvement when we apply 100 iterations.

B	K	D	UB number of reshuffles	Utilization	UB cost of reshuffles	UB vertical cost	Vertical and Horizontal Cost
10	5	25	4.90	69.79	12250.00	4463.40	16713.40
10	5	50	5.90	70.18	14750.00	4055.80	18805.80
10	10	25	4.10	53.36	10250.00	6497.40	16747.40
10	10	50	3.60	63.67	9000.00	7090.80	16090.80
20	10	25	7.30	72.68	18250.00	9199.20	27449.20
20	10	50	12.60	78.12	31500.00	9587.40	41087.40
20	20	25	4.80	48.42	12000.00	12852.80	24852.80
20	20	50	6.70	60.31	16750.00	26019.00	42769.00
30	15	25	16.00	76.60	40000.00	15256.00	55256.00
30	15	50	24.10	83.15	60250.00	15374.80	75624.80
30	30	25	9.20	50.18	23000.00	20024.60	43024.60
30	30	50	12.10	54.20	30250.00	19694.60	49944.60
40	20	25	28.50	78.46	71250.00	19694.60	90944.60
40	20	50	41.40	84.33	103500.00	21740.40	125240.40
40	40	25	17.60	48.01	44000.00	21238.20	65238.20
40	40	50	20.60	54.85	51500.00	25136.40	76636.40
50	25	25	44.80	82.16	112000.00	28547.60	140547.60
50	25	50	59.30	88.62	148250.00	28116.20	176366.20
50	50	25	22.40	49.24	56000.00	31762.00	87762.00
50	50	50	34.30	57.08	85750.00	31137.20	116887.20
100	50	25	136.70	84.44	341750.00	57765.60	399515.60
100	50	50	159.30	90.86	398250.00	57293.60	455543.60
100	100	25	68.50	50.07	171250.00	64707.80	235957.80
100	100	50	91.70	59.09	229250.00	66874.00	296124.00
Average			34.85	66.99	87125.00	25172.06	112297.06

Table 14 Cost Value given by LRTC for CSPX-2. Number of iterations =50

B	K	D	UB number of reshuffles	Utilization	UB cost of reshuffles	UB vertical cost	Vertical and Horizontal Cost
10	5	25	4.90	69.79	12250.00	4463.40	16713.40
10	5	50	5.90	70.18	14750.00	4055.80	18805.80
10	10	25	4.10	53.36	10250.00	6497.40	16747.40
10	10	50	3.60	63.67	9000.00	7090.80	16090.80
20	10	25	7.30	72.68	18250.00	9199.20	27449.20
20	10	50	12.60	78.12	31500.00	9587.40	41087.40
20	20	25	4.80	48.42	12000.00	12852.80	24852.80
20	20	50	6.70	60.31	16750.00	26019.00	42769.00
30	15	25	16.00	76.60	40000.00	15256.00	55256.00
30	15	50	24.10	83.15	60250.00	15374.80	75624.80
30	30	25	9.20	50.18	23000.00	20024.60	43024.60
30	30	50	12.10	54.20	30250.00	19694.60	49944.60
40	20	25	28.50	78.46	71250.00	19694.60	90944.60
40	20	50	41.40	84.33	103500.00	21740.40	125240.40
40	40	25	17.60	48.01	44000.00	21238.20	65238.20
40	40	50	20.60	54.85	51500.00	25136.40	76636.40
50	25	25	44.80	82.16	112000.00	28547.60	140547.60
50	25	50	59.30	88.62	148250.00	28116.20	176366.20
50	50	25	22.40	49.24	56000.00	31762.00	87762.00
50	50	50	34.30	57.08	85750.00	31137.20	116887.20
100	50	25	136.70	84.44	341750.00	57117.80	398867.80
100	50	50	159.30	90.86	398250.00	56780.80	455030.80
100	100	25	68.50	50.07	171250.00	64698.80	235948.80
100	100	50	91.70	59.09	229250.00	66872.80	296122.80
Average			34,85	66,99	87125,00	25123,28	112248,28

Table 15 Cost Value given by LRTC for CSPX-2. Number of iterations = 100

Gap 2 value represents the percentage gaps between the best upper bound and the feasible solution from CPLEX solution. BP_{cplex} .

$$\text{Gap 2} = \frac{\text{bestUB} - BP_{cplex}}{BP_{cplex}} \times 100 \tag{6.3}$$

The running time, number of reshuffling, total transportation cost from feasible solution by given CPLEX and LRTC are represented in Table-16 and Table-17. Thus we can compare the solution of CPLEX and LRTC approach for CSPX- 1 and CSPX-2. For all instances, number of reshuffling from LRTC is higher than the

number from feasible CPLEX. Total cost values of Lagrangean solution, illustrated in Table-16 and Table-17, are high value because of higher number of reshuffles. On the other hand, running time of LRTC is better than running time of CPLEX solution.

B	K	D	Gap2	Running Time (LR)	Running Time (GAMS)	Number of reshuffles (LR)	Number of reshuffles (GAMS)	Total Cost (LR)	Total Cost (GAMS)
10	5	25	7.28	1.56	2.00	4.66	1.00	3560528.00	3558770.80
10	5	50	4.77	1.70	2.00	5.70	2.50	5183297.60	5181542.20
10	10	25	10.30	2.10	300.00	3.10	0.00	1019250.80	1019060.80
10	10	50	27.90	2.20	128.00	3.30	1.00	850277.00	846023.30
20	10	25	23.54	3.60	401.80	7.10	3.10	4899396.40	4889366.40
20	10	50	15.70	3.90	499.30	13.20	6.20	4407551.40	4390109.90
20	20	25	90.72	5.80	182.50	4.90	0.00	25102.80	12840.50
20	20	50	115.50	6.70	121.80	6.70	0.20	30072.80	13817.00
30	15	25	1.00	6.90	420.60	11.40	3.70	5274136.00	5246894.20
30	15	50	1.45	7.40	480.00	23.40	5.10	4746284.60	4700607.00
30	30	25	129.92	13.40	181.70	10.30	0.00	45863.20	20002.70
30	30	50	144.44	17.00	61.80	11.30	0.00	47939.80	19692.40
40	20	25	30.83	14.00	540.20	29.50	5.00	5942795.80	5879172.90
40	20	50	10.72	15.50	480.40	38.50	7.40	5449617.80	5374284.20
40	40	25	181.33	29.40	420.60	18.60	0.00	71633.60	25751.20
40	40	50	207.78	38.50	241.20	21.40	0.00	79421.20	25928.30
50	25	25	4.16	25.40	600.00	43.40	14.00	6564199.00	6476067.20
50	25	50	26.77	28.00	600.00	54.40	19.00	5668827.60	5567915.40
50	50	25	170.41	54.60	300.00	21.70	2.00	86014.00	32289.40
50	50	50	278.89	73.00	438.00	40.30	7.00	117387.80	31131.50
100	50	25	3.23	225.30	600.00	132.40	84.40	8078117.80	7957947.80
100	50	50	2.96	273.80	600.00	157.30	82.50	10857530.80	10672603.20
100	100	25	120.90	480.30	600.00	70.20	50.00	240198.80	185443.40
100	100	50	160.08	113.40	540.00	94.80	59.50	303872.80	135166.00
Average			73.77	60.14	364.25	34.48	14.73	3064554.89	3010934.49

Table 16 Comparison of LRTC with optimal solution for CSPX-1. Number of iterations =100

B	K	D	Gap2	Running Time (LR)	Running Time (GAMS)	Number of reshuffles (LR)	Number of reshuffles (GAMS)	Total Cost (LR)	Total Cost (GAMS)
10	5	25	8.28	2.00	2.90	4.90	3.80	3051113.40	3048356.40
10	5	50	5.77	1.00	2.90	5.90	5.10	4883205.80	4880947.00
10	10	25	13.30	2.10	307.00	4.10	2.30	548347.40	543842.40
10	10	50	30.90	2.40	128.90	3.60	2.20	246290.80	242796.40
20	10	25	26.54	2.55	408.80	7.30	4.10	4058149.20	4050149.20
20	10	50	16.70	2.65	504.30	12.60	7.20	3464887.40	3451387.40
20	20	25	91.72	4.15	187.50	4.80	0.00	-1141347.20	-1153347.20
20	20	50	118.50	5.45	128.80	6.70	1.20	-1536731.00	-1550481.00
30	15	25	2.00	5.15	425.60	16.00	3.50	3769956.00	3739513.40
30	15	50	2.45	5.45	485.00	24.10	5.10	3059524.80	3012058.60
30	30	25	132.92	8.30	188.70	9.20	0.00	-1998375.40	-2021375.40
30	30	50	145.44	10.65	66.80	12.10	0.00	-2237255.40	-2267474.20
40	20	25	33.83	9.45	547.20	28.50	7.00	4484629.60	4447931.20
40	20	50	11.72	10.55	485.40	41.40	9.40	3522556.20	3464316.00
40	40	25	184.33	21.05	427.60	17.60	1.00	2704263.60	2662773.60
40	40	50	210.78	23.15	248.20	20.60	1.00	3152579.80	3103579.80
50	25	25	5.16	16.00	600.00	44.80	17.00	4887470.00	4817470.00
50	25	50	27.77	20.20	600.00	59.30	22.00	2446766.20	2353516.20
50	50	25	173.41	31.55	307.00	22.40	3.00	-3505838.00	-3554338.00
50	50	50	281.89	41.90	445.00	34.30	10.00	-4055112.80	-4115862.80
100	50	25	4.23	120.00	600.00	136.70	87.40	1672315.60	1549065.60
100	50	50	3.96	134.35	600.00	159.30	83.50	3988243.60	3798743.60
100	100	25	123.90	240.65	600.00	68.50	51.00	-7344592.20	-7388342.20
100	100	50	163.08	357.85	547.00	91.70	60.50	-8685276.00	-8763276.00
Average			75.77	44.94	368.53	34.85	16.14	809823.81	764664.58

Table 17 Comparison of LRTC with optimal solution for CSPX-2. Number of iterations =100

The *bestLB* bound is always identical with the lower bound found in the first iteration. None of the approaches could improve this lower bound value. Next, we concentrate on this behavior of the lower bound.

Firstly, LRTC approach is analyzed in detail with different value of *noImpLimit*. In the following, Figure-20 indicates the variation on the lower bound value of an instance. This fifth instance in our input list includes 100 bookings and 50 locations.

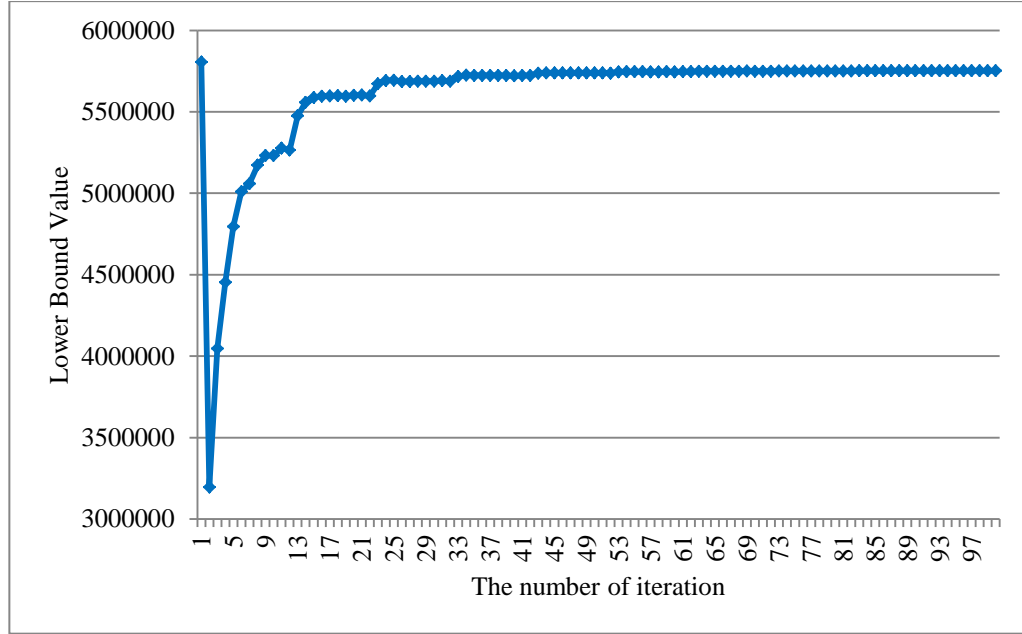


Figure 20 The behavior of Lower Bound on Each Iteration - LRTC

After the first iteration, the lower bound value dramatically decreases in the second iteration for all instances. For all instances solved by LRTC and LRSC show a gradual increase in lower bound values between second and third iteration. After third iteration, the lower bound converges to the value of first iteration. We can consider why the values of $bestLB$ for each instance are not affected by increasing the number of iterations or applying new approaches or using new parameter like $noImpLimit$. When we analyze the lower bounds from output file, $BestLB$ is always equal to lower bound found by first iteration. At the first iteration, λ_{bik}^0 and β_{bik}^0 values are equal to 0. When and they are equal to 0. The objective values of Subproblem2 and Subproblem3 for LRTC are represented as:

$$L_2(\lambda^0, \beta^0) = \min \sum_k \sum_b \sum_{i \in E(b)} z_{bik} \quad (6.4)$$

subject to

$$z_{bik} \in \{0,1\} \quad \forall b, k, i \in E(b) \quad (6.5)$$

The solution of Subproblem2 is trivial. To minimize the objective value, z_{bik}^* is always equal to 0.

$$L_3(\lambda^0, \beta^0) = \min \sum_k \sum_b \sum_{i \in E(b)} c_v * y_{bik} \quad (6.6)$$

subject to

$$0 \leq y_{bik} \leq x_{ik}^* \quad \forall b, k, i \in E(b) \quad (6.7)$$

The Subproblem3 at the first iteration is solved like Subproblem2. We find the lower bound at the first iteration for the original problem as the summation of $L_1(\lambda) + 0 + 0$ and then best lower bound is equal to the objective value of Subproblem1.

Hence a better solution cannot be found at further iterations. When lambda and beta parameters receive nonzero values, the value of objective function will decrease and it cannot be better than the lower bound at previous iteration. So, the value of the objective function of Subproblem2 and Subproblem3 should give negative values. The solution methods for these subproblems are therefore not effective.

For these reasons, we applied another method LRSLM for our CSP, lower bound value which is not changing with iterations. After first iteration, LB value is not dramatically decreased. When we compare these graphics with previous ones, overall, these graphs show that LRSLM approach prevent the lower bound value from sudden decline at the second iteration. On the other hand, there no noticeable variation on lower bound value.

The behavior of the upper bound and lower bound for all instances are examined. As we said before, the value of the lower bound at the first iteration is *bestLB*. There is no compliance between the upper and lower bound. The variation on the lower bound value does not affect the upper bound value.

6.3.2. Results of Heuristics

Three different approaches for solving these models were developed. We applied Subgradient Optimization Method in order to solve Lagrangean dual. Our algorithm was solved with subgradient heuristic for both small and large instances, but it does not provide a good quality solution. Furthermore, there was no improvement on the lower bound when we compute Lagrangean Relaxation. Subproblem1 can be solved in very small computation times. After first solution, this solution can be obtained as the lower bound, and upper bound heuristic can be improved with this solution. It may be a considerable advantage when quick solutions are preferred to a guarantee of optimality.

In order to verify the validity of the heuristics methods, we carried out several experiments. The summary results of new heuristics H1 and H2 are presented in this section. These approaches are applied for all mentioned instances set.

Table-18 represents the computational results the summary of H1 with mathematical model of initial solution based on CSPX-1 model. In Table-18, we present the vertical and horizontal cost value and running time (seconds) of H1.

B	K	D	Running time (H1)	Number of reshuffling (H1)	Cost of reshuffling (H1)	Vertical cost (H1)	Vertical and Horizontal Cost (H1)	Total Cost (H1)
10	5	25	1.20	4.56	11388.89	4251.56	15640.44	3559952.00
10	5	50	1.20	3.78	9444.44	5900.67	15345.11	5181545.11
10	10	25	1.68	2.83	7083.33	6579.67	13663.00	1019663.00
10	10	50	3.80	3.86	9642.86	9198.57	18841.43	853841.43
20	10	25	3.18	6.70	16750.00	10579.80	27329.80	4899829.80
20	10	50	3.09	13.38	33437.50	12938.75	46376.25	4411376.25
20	20	25	4.68	1.96	4900.00	11852.80	16752.80	16752.80
20	20	50	5.65	1.68	4200.00	13222.80	17422.80	17422.80
30	15	25	14.31	4.56	11400.00	14136.00	25536.00	5256036.00
30	15	50	15.34	9.36	23400.00	15384.60	38784.60	4711284.60
30	30	25	11.72	4.12	10300.00	17862.00	28162.00	33162.00
30	30	50	12.72	4.52	11300.00	20689.80	31989.80	31989.80
40	20	25	21.16	11.80	29500.00	21205.80	50705.80	5898205.80
40	20	50	22.48	15.40	38500.00	20867.80	59367.80	5383767.80
40	40	25	45.34	7.44	18600.00	23133.60	41733.60	41733.60
40	40	50	52.94	6.56	16400.00	24086.20	40486.20	40486.20
50	25	25	38.58	17.36	43400.00	2765.00	46165.00	6473665.00
50	25	50	42.32	21.76	54400.00	27607.60	82007.60	5587007.60
50	50	25	88.62	8.68	21700.00	30764.00	52464.00	52464.00
50	50	50	152.42	16.12	40300.00	31122.80	71422.80	96422.80
100	50	25	320.05	72.96	182400.00	56027.80	238427.80	7928427.80
100	50	50	384.67	82.92	207300.00	55689.80	262989.80	10670489.80
100	100	25	576.43	68.08	170200.00	64698.80	234898.80	234898.80
100	100	50	137.88	64.92	162300.00	56210.80	218510.80	228602.80
Average			81.73	18.97	47426.96	23199.04	70626.00	3027042.82

Table 18 Cost Value given by H1 heuristics

The horizontal and vertical cost, running time and number of reshuffles for H2 are denoted in Table-19. The computational results are presented as the summary of H2. When we compare the running times, it can be seen H1 is solved faster than H2. While average running time from Table-18 is 81.73 seconds, average running time of H2 is seen as 88.70 seconds in Table-19. Therefore, movement types of container in our heuristics algorithm are more suitable for H1.

After choosing the container from storage location, we have no flexibility over choice of storage location to relocate or swap them. Therefore containers are placed at a dummy location because of initial solution of H2. Regarding the initial solution, it can be seen that our goal of collecting together containers belonging to the same bookings has no positive effect on the number of reshuffles in limited time and iteration.

B	K	D	Running time (H2)	Number of reshuffling (H2)	Cost of reshuffling (H2)	Vertical cost (H2)	Vertical and Horizontal Cost (H2)	Total Cost (H2)
10	5	25	1.33	5.47	13666.67	4251.56	17918.22	3567229.78
10	5	50	1.40	4.53	11333.33	5900.67	17234.00	5182234.00
10	10	25	1.78	3.40	8500.00	6579.67	15079.67	1100079.67
10	10	50	4.80	4.63	11571.43	9198.57	20770.00	847770.00
20	10	25	4.45	8.04	20100.00	10579.80	30679.80	4890679.80
20	10	50	6.09	16.05	40125.00	12938.75	53063.75	4403063.75
20	20	25	8.43	2.35	5880.00	11852.80	17732.80	17732.80
20	20	50	9.48	2.02	5040.00	13222.80	18262.80	18262.80
30	15	25	16.08	5.47	13680.00	14136.00	27816.00	5258316.00
30	15	50	28.64	11.23	28080.00	15384.60	43464.60	4715964.60
30	30	25	12.98	4.94	12360.00	17862.00	30222.00	42722.00
30	30	50	14.27	5.42	13560.00	20689.80	34249.80	34249.80
40	20	25	22.45	14.16	35400.00	21205.80	56605.80	5904105.80
40	20	50	23.16	18.48	46200.00	20867.80	67067.80	5391467.80
40	40	25	48.36	8.93	22320.00	23133.60	45453.60	45453.60
40	40	50	55.43	7.87	19680.00	24086.20	43766.20	43766.20
50	25	25	59.76	20.83	52080.00	2765.00	54845.00	6482345.00
50	25	50	65.32	26.11	65280.00	27607.60	92887.60	5597887.60
50	50	25	91.62	10.42	26040.00	30764.00	56804.00	56804.00
50	50	50	156.42	19.34	48360.00	31122.80	79482.80	104482.80
100	50	25	344.55	87.55	218880.00	56027.80	274907.80	7964907.80
100	50	50	401.72	99.50	248760.00	55689.80	304449.80	10711949.80
100	100	25	598.38	81.70	204240.00	64698.80	268938.80	268938.80
100	100	50	151.98	77.90	194760.00	56210.80	250970.80	261062.80
Average			88.70	22.76	56912.35	23199.04	80111.39	3037978.21

Table 19 Cost Value given by H2 heuristics

Gap 3 value represents the percentage gaps between the best value from H1 and the feasible solution from CPLEX solution. BP_{cplex}

$$\text{Gap 3} = \frac{S_{H1_best} - BP_{cplex}}{S_{H1_best}} \times 100 \tag{6.8}$$

Table-20 denotes the running time, number of reshuffles, total transportation cost from feasible solution by given CPLEX and H1. Thus we can compare the solution of CPLEX and H1 heuristics.

B	K	D	Gap3	Running Time (H1)	Running Time (GAMS)	Number of reshuffling (H1)	Number of reshuffling (GAMS)	Total Cost (H1)	Total Cost (GAMS)
10	5	25	2.71	1.20	2.00	4.56	1.00	3559952.00	3558770.80
10	5	50	2.3	1.20	2.00	3.78	2.50	5181545.11	5181542.20
10	10	25	1.32	1.68	300.00	2.83	0.00	1019663.00	1019060.80
10	10	50	1.2	3.80	128.00	3.86	1.00	853841.43	846023.30
20	10	25	3.66	3.18	401.80	6.70	3.10	4899829.80	4889366.40
20	10	50	4.15	3.09	499.30	13.38	6.20	4411376.25	4390109.90
20	20	25	25.98	4.68	182.50	1.96	0.00	16752.80	12840.50
20	20	50	24.7	5.65	121.80	1.68	0.20	17422.80	13817.00
30	15	25	1.28	14.31	420.60	4.56	3.70	5256036.00	5246894.20
30	15	50	1.02	15.34	480.00	9.36	5.10	4711284.60	4700607.00
30	30	25	42.85	11.72	181.70	4.12	0.00	33162.00	20002.70
30	30	50	39.37	12.72	61.80	4.52	0.00	31989.80	19692.40
40	20	25	10.48	21.16	540.20	11.80	5.00	5898205.80	5879172.90
40	20	50	4.45	22.48	480.40	15.40	7.40	5383767.80	5374284.20
40	40	25	41.2	45.34	420.60	7.44	0.00	41733.60	25751.20
40	40	50	46.82	52.94	241.20	6.56	0.00	40486.20	25928.30
50	25	25	1.27	38.58	600.00	17.36	14.00	6477665.00	6476067.20
50	25	50	5.16	42.32	600.00	21.76	19.00	5587007.60	5567915.40
50	50	25	46.44	88.62	300.00	8.68	2.00	52464.00	32289.40
50	50	50	71.97	152.42	438.00	16.12	7.00	96422.80	31131.50
100	50	25	2.6	320.05	600.00	72.96	84.40	7969427.80	7957947.80
100	50	50	1.8	384.67	600.00	82.92	82.50	10680489.80	10672603.20
100	100	25	25.29	576.43	600.00	68.08	50.00	234898.80	185443.40
100	100	50	44.11	137.88	540.00	64.92	59.50	228602.80	135166.00
Average			18.84	81.73	364.25	18.97	14.73	3029334.48	3010934.49

Table 20 Comparison of H1 with optimal solution

For all instances, number of reshuffles from H1 is higher than the number from feasible CPLEX. The average number of reshuffles is 18.97 for H1 while the average value for GAMS is 14.73. The running time of H1 is also better than running time of CPLEX solution, we get better solution form CPLEX solution. In addition, the average gap is 32.26.

The number of reshuffles for three different solution methods, LR, H1 and GAMS, are presented in Figure-21. It is seen that number of reshuffles is fewer for H1 when compared with LR. This also positively affects the total cost value, because the cost of reshuffling is very high and has an important role in the structure of our problem.

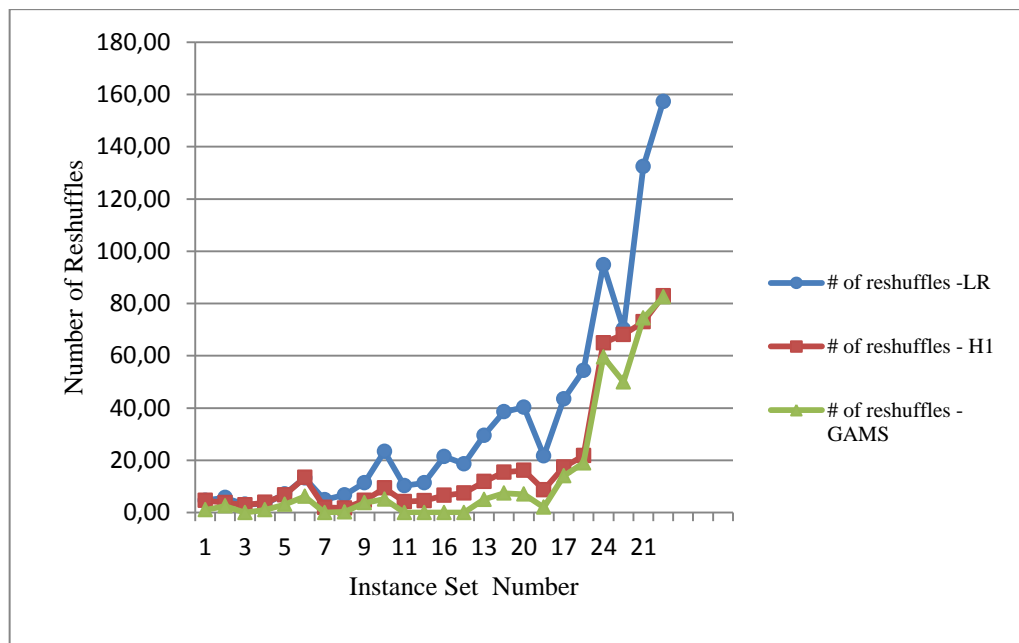


Figure 21 Number of reshuffles for LR, H1 and GAMS

The comparison of running time, number of reshuffles and total cost of H2 and CPLEX solutions is depicted in Table-21. The average number of reshuffles for H2 is greater than average value for CPLEX. The lower value for the average number of

reshuffles can be obtained from the above tables. This difference is directly related to the total cost of H2, which is higher than CPLEX solutions.

Gap 4 is denoted in following table as the percentage gaps between the best value from H1 and the feasible solution from CPLEX solution. BP_{cplex} .

$$\text{Gap 4} = \frac{S_{H2_best} - BP_{cplex}}{S_{H2_best}} \times 100 \tag{6.9}$$

B	K	D	Gap4	Running Time (H2)	Running Time (GAMS)	Number of reshuffling (H2)	Number of reshuffling (GAMS)	Total Cost (H2)	Total Cost (GAMS)
10	5	25	1.78	1.33	2.00	5.47	1.00	3567229.78	3558770.80
10	5	50	2.56	1.40	2.00	4.53	2.50	5182234.00	5181542.20
10	10	25	1.20	1.78	300.00	3.40	0.00	1100079.67	1019060.80
10	10	50	1.77	4.80	128.00	4.63	1.00	847770.00	846023.30
20	10	25	4.33	4.45	401.80	8.04	3.10	4890679.80	4889366.40
20	10	50	5.98	6.09	499.30	16.05	6.20	4403063.75	4390109.90
20	20	25	27.29	8.43	182.50	2.35	0.00	17732.80	12840.50
20	20	50	26.41	9.48	121.80	2.02	0.20	18262.80	13817.00
30	15	25	1.30	16.08	420.60	5.47	3.70	5258316.00	5246894.20
30	15	50	1.61	28.64	480.00	11.23	5.10	4715964.60	4700607.00
30	30	25	54.52	12.98	181.70	4.94	0.00	42722.00	20002.70
30	30	50	43.49	14.27	61.80	5.42	0.00	34249.80	19692.40
40	20	25	11.57	22.45	540.20	14.16	5.00	5904105.80	5879172.90
40	20	50	5.18	23.16	480.40	18.48	7.40	5391467.80	5374284.20
40	40	25	45.64	48.36	420.60	8.93	0.00	45453.60	25751.20
40	40	50	48.51	55.43	241.20	7.87	0.00	43766.20	25928.30
50	25	25	3.18	59.76	600.00	20.83	14.00	6482345.00	6476067.20
50	25	50	7.50	65.32	600.00	26.11	19.00	5597887.60	5567915.40
50	50	25	47.58	91.62	300.00	10.42	2.00	56804.00	32289.40
50	50	50	78.55	156.42	438.00	19.34	7.00	104482.80	31131.50
100	50	25	3.14	344.55	600.00	87.55	84.40	7964907.80	7957947.80
100	50	50	3.41	401.72	600.00	99.50	82.50	10711949.80	10672603.20
100	100	25	58.86	598.38	600.00	81.70	50.00	268938.80	185443.40
100	100	50	53.72	151.98	540.00	77.90	59.50	261062.80	135166.00
Average			22.46	88.70	364.25	22.76	14.73	3037978.21	3010934.49

Table 21 Comparison of H2 with optimal solution

The number of reshuffles given by LR, H1, H2 and GAMS are shown in Figure-22. The number of reshuffles is lower than that of LR.

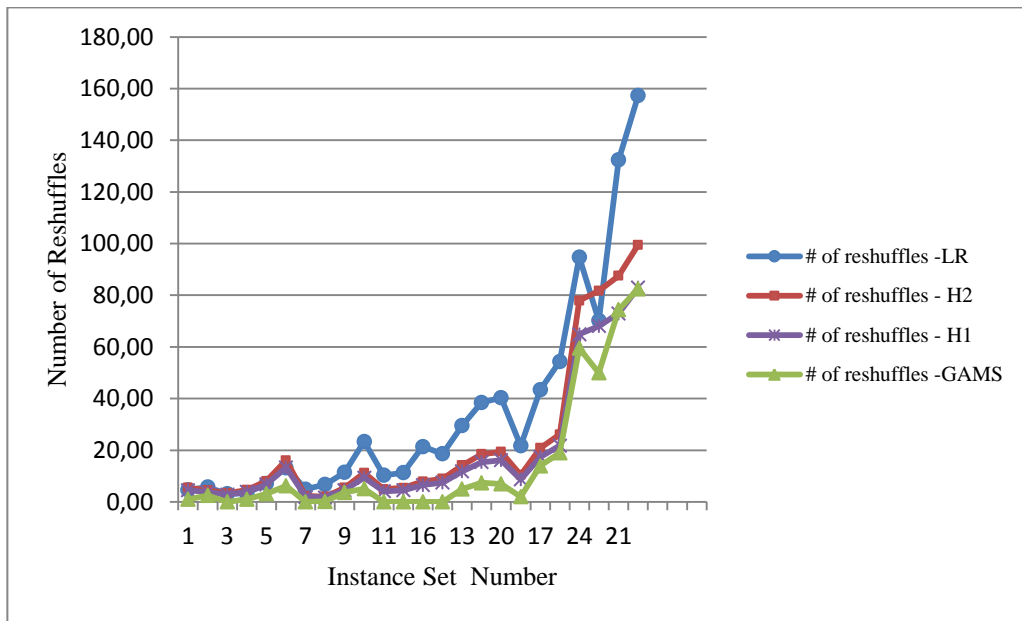


Figure 22 Number of reshuffles for LR, H1, H2 and GAMS

Figure-23 demonstrates total transport cost for all heuristics methods.

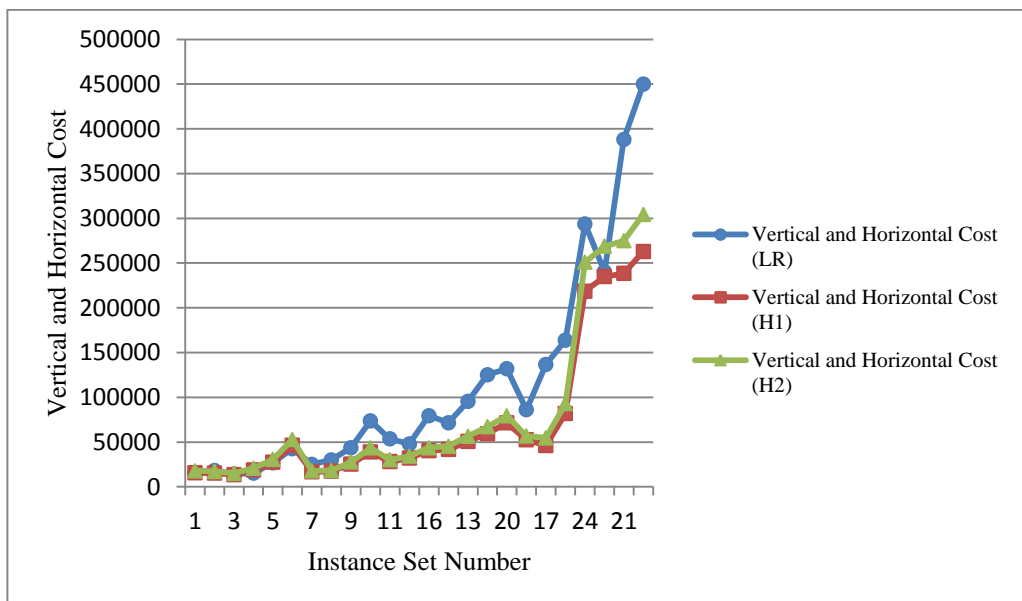


Figure 23 Transport cost for LR, H1 and H2

Another point worth noting is that number of reshuffles can be reduced using our proposed heuristic methods H1 and H2. These two heuristics clearly outperform LR. This is important because reducing the number of reshuffles affects transportation costs at storage location.

As can be seen from Figure-23, the transportation cost given by LR is not acceptable for some instance sets. In other words, we can say that LR approach cannot generate a sufficiently powerful upper bound. When we observe how the upper bound value of LR varies with respect to lower bound, it can be seen that there is no compliance between the upper and lower bound. The variation in the lower bound value does not affect the upper bound value. Therefore, we can say that LR is not a suitable method for our model. Figure-24 also denotes the behavior of lower, upper bounds and optimal solution for one solution, revealing that Subgradient Optimization Method is not effective on this problem.

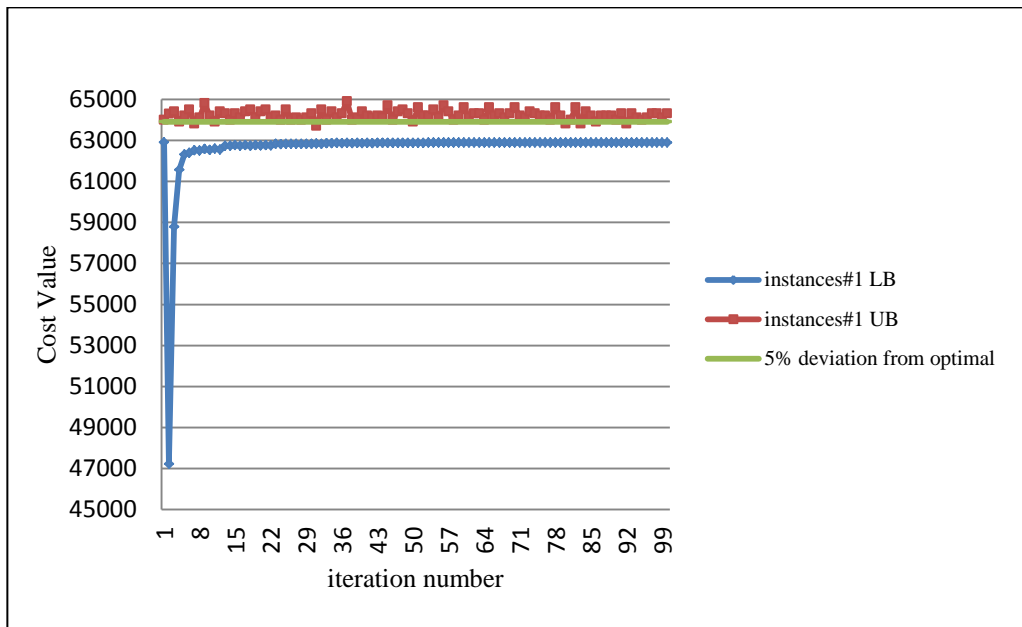


Figure 24 The behavior of bounds of all iteration (LTRC)

Figure-24 gives an example of upper, lower bound and optimal solution of an instance. After the first iteration, we find the solution of lower bound and upper

bound of Lagrangean Relaxation in an attempt to improve this solution with the newly proposed heuristics. These heuristics, H1 and H2, seem to generate quite powerful upper bounds, and no significant performance difference is observed between the two; however, it can be stated that H1 has a (small) advantage over H2.

CHAPTER - 7

CONCLUSION

In this study, we considered the mixed integer programming model for container storage problem (CSP). The objective of the problem is to minimize the total transportation cost at the container terminal. Two types of movements are of concern: vertical and horizontal.

We propose three mathematical models with differing sets of assumptions, which reduce this three-dimensional storage problem to two-dimensions. To avoid infeasibility, we improved first mathematical model CSP, and formulated CSPX-1 and CSPX-2.

We also developed Lagrangean Relaxation-based heuristic algorithm. Our mathematical models are designed to provide an optimal solution using CPLEX. Some instances could not be solved for large inputs within the 1 hour CPU time limit. We used CPLEX solution, which was employed as a benchmark in order to measure the performance of Lagrangean Relaxation Algorithm. Three different approaches to solving these models were developed. In this process, we applied

subgradient optimization method in order to solve lagrangean dual. Our algorithm was solved with subgradient heuristic for both small and large instances.

We observe that Lagrangean relaxation for the last constraints does not provide good quality solutions, i.e. there was no improvement in the lower bound during the iterations. The developed LRTC procedure generates the solution for first iteration in very small computation times. After first iteration, we were able to achieve the solution as lower bound, and this solution enabled upper bound heuristic to be improved.

To generate a more effective solution in a shorter time, we proposed a new upper bound heuristics approach. The most important aspect is that heuristic approach, H1 uses our LR-based bounding procedures only for iteration. In other words, the initial phase of H1 generates a feasible placement for containers at the storage location with *Subproblem 1* only for iteration. The initial feasible solution of our first heuristic, H1 aims to minimize the horizontal cost at the port. We designed a new technique for initial feasible solution. This heuristic, named as H2, has a new initial solution, which has small variants in initial feasible solution. The container is placed together with its container set without considering horizontal cost. This new heuristics was found to be appropriate for adapting CSP to the real world problem.

The proposed approach, LR was tested on different instances and proved to be successful in regard to timing. Unfortunately, however, it can be seen that there is no compliance between the upper and lower bounds. Another issue is that the variation on the lower bound value does not affect the upper bound value. None of the results of LR obtained indicate that this approach can generate powerful upper bound. In contrast, our proposed heuristics, H1 and H2 seem to generate quite powerful upper bounds. It has been demonstrated that no significant performance difference is observed between the two heuristics. Although it can be seen that H1 has a (slight) advantage over H2 regarding the performance of the upper bound. Another point

worth noting is that number of reshuffles can be reduced using our proposed heuristic methods H1 and H2. Reducing the number of reshuffles positively affects transportation cost on storage location. This study has clearly shown these heuristics methods are significantly superior to LR in terms of performance.

As the CPU times are very small for all heuristics (LR, H1 and H2), they are suitable method for solving extremely large instances for large-capacities storage locations. We evaluated the performance of our heuristics using GAP values for all instances, and found that for the ten instance set, the GAP was not acceptable. For all instances set, H1 was computed with GAP 18.84% at an average of 81.73 seconds of CPU time, while H2 is computed with GAP 22.46% and 88.70 seconds. The gap value of H1 is lower than that of H2 for the five instance set. It can therefore be concluded that H1 outperforms H2 regarding GAP values. Thus, for any problem instance, it is advisable to use all heuristics and select the best solution, as the computation times are very small and no heuristic seems to consistently outperform the others.

From the study that has been conducted, it is possible to conclude that H1 generally outperforms H2 and LR approaches.

Based on the results, it can be concluded that this research into minimizing the number of reshuffles has been successful. The proposed method H1 can be readily used in practice. However further study of the issue is still required. In our future research we intend to concentrate on different container types and sizes to place to the storage location. For example, import and export containers have differences. Future research should consider special container types, including open top, and refrigerated containers. Further study of the issue in terms of the usage of dummy location would also be of interest. In the real world, arriving containers can be placed to a temporary location. When the storage location becomes available, they can be moved from the temporary area. However, they cannot be stored in the temporary

storage location until they leave the berth with their ship. During this time period, the model can allow for the containers to move from the dummy location to the storage location.

On the basis of the positive findings presented in this study, work on the related issues is continuing and will be presented in future papers.

REFERENCES

- [1] Ahuja, R.K., Magnanti, T.L., Orlin, J.B., 1993. Network Flows, Prentice-Hall, Upper Saddle River. NJ.
- [2] Alphaliner, 2012. Weekly Alphaliner Report. <http://www.alphaliner.com/>, 2012.
- [3] Balas, E., Hammaer, P.L., 1964. On the generalized transportation problem. Management Sci., 11. 188-202.
- [4] Bekki, O.B., Azizoglu, M., 2008. Operational fixed interval scheduling problem on uniform parallel machines, International Journal of Production Economics 112, 756-768.
- [5] Cao, B., 1992. Transportation problems with nonlinear side constraints: A branch-and-bound approach, Zeitschrift für Operations Research 36, 185–197.
- [6] Cao, B., Uebe, G., 1995. Solving transportation problems with nonlinear side constraints with tabu search. Computers and Operations Research 22 (6), 593–603. Heuristics 3 (4), 305–326.
- [7] Chen, H., Saigal, R., 1977. Aprimal algorithm for solving a capacitated network flow problem with additional linear constraints. Networks 7 (1), 59–79.
- [8] Eliiyi, D.T., 2004. “Operational Fixed Job Scheduling Problem”. Doktora Tezi. Endüstri Mühendisliği Bölümü, Orta Doğu Teknik Üniversitesi, Türkiye.
- [9] Eliiyi, D.T., Azizoğlu, M., 2005. Approximation Algorithms for Operational Fixed Job Scheduling Problems, Proceedings of 35th International Conference on Computers & Industrial Engineering.1. 567-572, İstanbul.

- [10] Eliiyi, D.T., Azizoğlu, M., 2006. Spread time constraints in operational fixed job scheduling, *International Journal of Production Research*, 44. 4343-4365.
- [11] Eliiyi, D.T., Azizoğlu, M., 2009c. A Fixed Job Scheduling Problem with Machine-Dependent Job Weights, *International Journal of Production Research*, 47. 2231 - 2256.
- [12] Eliiyi, D.T., Kandiller, L., 2008. A Decision Support System for the Cell Formation Problem, *International Journal of Industrial and Systems Engineering*, 3. 348-367.
- [13] Eliiyi, D.T., Korkmaz, A.G., Çiçek, A.E., 2009b. Operational Variable Job Scheduling with Eligibility Constraints: A Randomized Constraint-Graph-Based Approach, *Technological and Economic Development of Economy*, 15. 245 - 266.
- [14] Eliiyi, D.T., Örnek, A., Karakütük, S.S., 2009a. A Vehicle Scheduling Problem with Fixed Trips and Time Limitations, *International Journal of Production Economics*, 117. 150-161.
- [15] Eliiyi, D.T., Özlen, M., 2008. A Lagrangean relaxation approach for the mixed-model flow line sequencing problem, *Computers & Operations Research* 35 (2008) 933 – 943.
- [16] Fischetti, M., Martello, S., Toth, P., 1987. The Fixed Job Schedule Problem with Spread-Time Constraints, *Operations Research*, 35. 849-858.
- [17] Fisher, M.L., 1981. The Lagrangian relaxation method for solving integer programming problems, *Management Science* 27, 1–18.
- [18] Geoffrion, A. M., 1974. Lagrangian relaxation and its uses in integer programming, *Math. Programming Stud.*, 2 82-114.
- [19] Glover, F., Klingman. D., 1985. Basis exchange characterizations for the simplex SON algorithm for LP/Embedded networks, *Mathematical Programming Study* 24, 141–157.

- [20] Glover, F., Klingman, D., Phillips N., 1992. *Network Models in Optimization and Their Applications in Practice*, John Wiley & Sons, New York.
- [21] Global Industry Analysts Inc., 2012. *Maritime Containerization: A Global Strategic Business Report.*, 09.04.2012
- [22] Granfinkel, R.S., Rao. M.R., 1971. The bottleneck transportation problem, *Naval Logistics Q.*, 18.465-72.
- [23] Gürgenç, K., 2010. “Limanlar ve destekleme paketleri” UTIKAD, <http://www.utikad.org.tr/haberler/?id=4741>, 19 Ocak 2010.
- [24] Hammer, P.L., 1969. Time minimizing transportation problems, *Naval Res. Logistics Q.*, 16.345-57.
- [25] Held, M., Karp, R. M., 1970. The traveling salesman problem and minimum spanning trees, *Oper. Res.*, 18 1138-1162.
- [26] Imai, A., Nishimura, E., 2007. Hattori M and Papadimitriou S. Berth allocation at indented berths for mega-containerships, *European Journal of Operational Research*, 179. 579-593.
- [27] Hitchcock, F. L., 1941. The distribution of a product from several sources to numerous localities, *J.Math. Phys.*, 20. 224-30.
- [28] Kennington, J.L., Helgason, R.V., 1980. *Algorithms for Network Programming*. Wiley, New York. NY.
- [29] Kim, K.H., 1997. Evaluation of the number of rehandles in container yards. *Computers and Industrial Engineering*, 32(4). 701–11.
- [30] Kim, K.H., Park, Y.M., Ryu, K.R., 2000. Deriving decision rules to locate export containers in container yards, *European Journal of Operational Research*, 124. 89–101.

-
- [31] Kim, K.H., Hong, G.P., 2006. A heuristic rule for relocating blocks, *Computers and Operations Research*, 33. 940–954.
- [32] Klingman, D., Russel, R., 1975. Solving constrained transportation problems, *Ops. Res*, 23. 91-106.
- [33] Kolen, A.J.W., Lenstra, J.K., Papadimitriou, C.H., Spiessma, F.C.R., 2007. Interval scheduling: A survey. *Naval Research Logistics*, 54. 530 - 543.
- [34] Koopman, T.C., 1947. Optimum utilization of transportation system, *Proc.Intern.Statist Conf. Washington. D.C.*
- [35] Kovalyov, M.Y., Ng, C.T., Cheng, T.C.E., 2007. Fixed interval scheduling: Models, applications, computational complexity and algorithms, *European Journal of Operational Research*, 178. 331-342.
- [36] Kroon, L.G., 1990. Job Scheduling and Capacity Planning in Aircraft Maintenance. PHD Thesis. Rotterdam School of Management, Erasmus University, Holland.
- [37] Kroon. L.G., Salomon, M., Van Wassenhove, L.N., 1995. Exact and Approximation Algorithms for the Operational Fixed Interval Scheduling Problem. *European Journal of Operational Research*, 82. 190-205.
- [38] Kolen, A.J.W., Kroon, L.G., 1991. On the Computational Complexity of (Maximum) Class Scheduling. *European Journal of Operational Research*, 54. 23-38.
- [39] Kuno, T., Utsunomiya, T., 2002. A Lagrangian Based Branch-and-Bound Algorithm for Production-transportation Problems, *Journal of Global Optimization*, 18: 59–73. 2000.
- [40] Laura, J. R., 1964. Topology and computation of the generalized transportation problem, *Management Sci.*, 177-87.

-
- [41] Lee, Y., Lee, Y.J., 2010. A heuristic for retrieving containers from a yard, *Computers and Operations Research*, 37. 1139–1147.
- [42] Murty, K.G., 1992. *Network Programming*, Prentice-Hall. Englewood Cliffs. NJ.
- [43] Merrill, O.H., Tobin, R.L., 1969. An algorithm for solving a transportation problem with inequalities and some negative cost, Tech. Report 69-80, Dept, O Industrial Engineering, Univ of Michigan.
- [44] Narciso, M.G., Lorena, L.A.N., 1999. Lagrangian/surrogate relaxation for generalized assignment problems, *European Journal of Operational Research* 114 (1), 165–177.
- [45] Nishimura, E., Imai, A., Janssens, G.K., Papadimitriou, S., 2009. Container storage and transshipment marine terminals, *Transportation Research Part E*. 45, 771–786.
- [46] Park, S., Seo, J., 2009. Mathematical modelling and solving procedure of the planar storage location assignment problem. *Computers and Industrial Engineering*, 57. 1062–1071.
- [47] Rojanasoonthon, S., Bard, J.F., 2003. A GRASP for parallel machine scheduling with time windows, *INFORMS Journal on Computing*, 17. 32-51.
- [48] Romeijn, H., E., and Sargut, F., Z., 2011. The stochastic transportation problem with single sourcing. *European Journal of Operational Research* 214 (2011) 262–272.
- [49] Sargut, Z., 2006. “Efficient Approaches to Integrated Requirements Planning Problem in Supply Chain Optimization”. Phd Thesis., University of Florida, USA.

- [50] Steenken, D., Voss, S., Stahlbock, R., 2004. Container terminal operations and operations research – a classification and literature review, *OR Spectrum*. 26. 3–49.
- [51] Sharma, J.K., Swarup, K., 1978. The minimization in transportation problems, *N.Z.O.R.*, 6. 75-88.
- [52] Sun, M., 2002. The transportation problem with exclusionary side constraints and two branch-and-bound algorithms, *European Journal of Operational Research* 140, 629–647.
- [53] Sun, M., 1998. A tabu search heuristic procedure for solving the transportation problem with exclusionary side constraints, *Journal of Heuristics* 3 (4), 305–326.
- [54] Williams, A.C., 1963. A stochastic transportation problem, *Ops. Res.*, 11.759-70.
- [55] Wolfe, W.J., Sorensen, S.E., 2000. Three Scheduling Algorithms Applied to the Earth Observing Systems Domain, *Management Science*, 46. 148-168.
- [56] World Trade Organization Report, 2007. http://www.wto.org/English/res_e/booksp_e/anrep_e/world_trade_report06_e.pdf, 28.08.2007.
- [57] Vis, I.F.A., Koster de, R., 2003. Transshipment of containers at a container terminal: an overview, *European Journal of Operational Research*, 147. 1–16.
- [58] Yeo, G., Roe, M., Dinwoodie, J., 2008. Evaluating the competitiveness of container ports in Korea and China, *Transportation Research Part A*, 42. 910-921.
- [59] Zhang, C., Liu, J., Wan, Y., Murty, K.G., and Linn, R.J., 2003. Storage space allocation in container terminals, *Transportation Research Part B: Methodological*, 37. Issue 10. 883-903.