TRANSSHIPMENT CONTAINER TERMINALS

by

Duygu Korkmaz

A Thesis Submitted to the
Graduate School of Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Industrial Engineering

Koç University

August, 2014

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Duygu Korkmaz

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Prof. Dr. Ceyda Oğuz (Advisor)

_____

Assoc. Prof. Dr. Emre Alper Yıldırım

_____

Assoc. Prof. Dr. Abdullah Daşçı

Date: _____

to Izmir...

ABSTRACT

The ever increasing number of containers to be transported necessitates expanding the capacity of container terminals. This can be achieved either by enlarging the container terminals physically or by managing the container terminals better. Since the first alternative is costly, and sometimes not possible due to limited land, container terminal managers focus on the second alternative. One possible way to increase the capacity of a container terminal is to reduce the vessel handling times. By that way, containers will spend little or no time at the yard storage phase, which will directly enhance the capability of container terminals to compete with each other. This setting is applicable to large transshipment hubs where many vessels can be moored and processed at the same time and discharged containers can directly go to their outbound vessels. Since the yard is a critical resource for the container terminal management, this environment will result in less cost and better utilization of the yard space. In this study, we consider a transshipment container terminal where all containers must be loaded to their outbound vessels directly after being unloaded from their incoming vessels. Our aim is to determine a schedule of unloading and loading times of containers while minimizing the makespan, which is the total completion time of all operations. This problem has not been studied before. The most related work deals with a case where there are two vessels and three types of containers, which are loaded, unloaded and directly transshipped ones. In our study, we consider just the transshipment containers but the number of vessels is not restricted to two. We develop a mathematical model to represent the system and analyze its performance through computational studies. We create instances from different sizes to understand the limits of the mathematical model. After identifying these limits for the model, we study possible solution methodologies.

ÖZET

Günümüzde artan konteyner taşımacılığı, konteyner terminallerinin birbirleriyle rekabet edebilmeleri için kapasitelerini arttırmaları gerekliliğini doğurmuştur. Kapasite arttırımı ya fiziksel olarak operasyon alanının genişletilmesi ya da varolan terminallerin daha iyi yönetilmesiyle mümkündür. Birinci seçenek çok masraflı olduğu ve fiziksel açıdan mümkün olamayabileceği için, terminal yönetimi için ikinci alternatif üzerine yoğunlaşmak daha mantıklıdır. Konteyner terminallerinin rekabet yeteneği gemilerin rıhtımda işlenme süresine fazlasıyla bağlıdır. Bu nedenle, gemilerin rıhtımda daha az süre tutulması oldukça önemlidir. Birçok geminin aynı anda demirleyebileceği ve işlenebileceği, boşaltılan konteynerlerin ise doğrudan onları gidecekleri yerlere taşıyacak olan gemilere aktarılabileceği büyük aktarma merkezleri bu amaca ulaşmak için en uygun ortamlardan biridir. Bu çalışmada, bütün konteynerlerin böyle bir ortamda boşaltılıp yüklendiği bir konteyner aktarma merkezi incelenmiştir. Çalışmada amaç, tamamlanma zamanını en azlayarak konteynerlerin tahliye ve yükleme işlemlerini gösteren bir çizelge belirlemektir. Böyle bir ortamı inceleyen bir çalışma daha önce yapılmamış olmakla birlikte, yapılan en benzer çalışma iki gemi ve doğrudan aktarılan konteynerler dışında ithal ve ihraç konteynerlerin de bulunduğu bir problemi inceleyen çalışmadır. Bizim çalışmamızda gemi sayısının iki ile kısıtlı kalmadığı ve konteynerlerin hepsinin doğrudan aktarılması gerektiği bir ortam ele alınmıştır. Böyle bir ortam için konteynerlerin tahliye ve yükleme çizelgesini belirleyecek bir tamsayılı matematiksel model geliştirilmiş ve hesaplamalı deneylerle sistemin performansı analiz edilmiştir. Modelin sınırlarını analiz edebilmek için problem örnekleri yaratılmış ve muhtemel çözüm yöntemleri incelenmiştir.

ACKNOWLEDGMENTS

I would firstly like to express my appreciation to my advisor, Prof. Dr. Ceyda Oğuz, for her engagement and guidance throughout the entire process of this thesis. She provided me with the introduction to the topic and enlightened my way with all the valuable suggestions.

I would like to thank Assoc. Prof. Dr. Emre Alper Yıldırım and Assoc. Prof. Dr. Abdullah Daşçı for taking part in my thesis committee.

I would like to acknowledge financial support from TUBITAK during my master thesis.

I owe my warmest regards to my friends Aysu, Ayşegül and Güneş, a group of wonderful people, for their irreplaceable and unforgettable friendships during my master study at Koç University.

My special thanks to Orçun, who was always there when I need. Without his love and understanding, I would not be able to make it.

Last but not the least, I thank my parents Havva and Doğan and my sister Gamze, for their endless support and faith in my efforts. To my love and family, I dedicate this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

AGV	Automated Guided Vehicle
BAP	Berth Allocation Problem
CP	Constraint Programming
CSP	Constraint Satisfaction Problem
GA	Genetic Algorithm
GRASP	Greedy Randomized Adaptive Search Procedure
LB	Lower Bound
LNS	Large Neighborhood Search
MILP	Mixed Integer Linear Programming
QC	Quay Crane
QCAP	Quay Crane Assignment Problem
QCSP	Quay Crane Scheduling Problem
SC	Straddle Carrier
TEU	Twenty Feet Equivalent Unit
TS	Tabu Search
UB	Upper Bound

Chapter 1

INTRODUCTION

Using containers for overseas shipping has caught considerable interest due to its various advantages. The main benefits of using containers for long distance transportation are that they have less damaging, require less packaging and leads to higher productivity. Containers are moved from one destination to another by large ships, called vessels. Vessels arrive at a port, occupy a place to moor and the incoming containers are unloaded while the outgoing containers are loaded. These unloading and loading operations are performed by large vehicles called Quay Cranes (QCs). It is crucial to manage these QCs efficiently since they are very expensive and are critical resources for terminals.

The increasing volume of container shipping worldwide requires managing the terminals better to stay stable in the competitive market. Terminals should find a way to increase their capacity; they either need to enlarge their sizes or reduce the vessel handling times to be able handle greater number of vessels and containers. It is a very difficult option to enlarge the sizes of the container terminals since it is a very costly investment. Therefore, the focus should be on finding and applying new methods to reduce the vessel handling times. It is important for vessels to spend as little time as possible at container terminals because the more a vessel waits to be processed, the later the containers are sent to their destinations. This is not desirable for either the terminal management, ship owners or the final customer because the containers on the vessels or in the yard storage mean money-tied-up to inventories and wasted resources. If the yard storage phase is skipped, the vessel handling time decreases. In order for containers to skip the yard storage phase, they should directly move from one vessel to another without going to yard storage places. This environment is observed in large transshipment hubs in which many vessels can be processed simultaneously. When the inbound vessel (the vessel carrying the container to the terminal) and the outbound vessel (the vessel which will take the container to its destination) of a

container are available at the same time, there is an opportunity to move that container directly to its outbound vessel after unloading it from its inbound vessel. By that way, the containers will lose no time at the yard storage phase. QCs which load and unload the containers will be better utilized and the overall vessel handling time will decrease. With this transshipment modality, the loading and unloading operations are handled together. They are not regarded as two separate processes performed on a vessel anymore; they are rather intertwined activities that should be planned accordingly. Different from classical scheduling of loading or unloading operations of a vessel, both of the operations are executed for more than one vessel at the same time for the transshipment case. The opportunity to reduce the vessel handling times in transshipment container terminals is the main drive of this study.

The solution methods utilized in this thesis are Tabu Search (TS) and Constraint Programming (CP). TS is a metaheuristic method employing local search for optimization problems. Local search moves from one solution to its adjacent ones with the hope of finding better solutions. The enhancement of TS is in its ability to guide the search by using memory techniques. The algorithm prevents moving to a previously visited non-promising solution by marking a tabu status on that solution. TS is widely used for tackling combinatorial optimization and scheduling problems. CP is a programming logic which makes use of constraints in order to represent the problem. Constraints state the relations between the variables and they specify the properties of a solution to be found. The CP approach is to find the state in which all of the constraints are satisfied at the same time.

The content of this thesis can be explained as follows:

- We develop a mixed integer linear programming (MILP) model for the complex problem encountered at transshipment container terminals. The preliminary tests are carried out to understand the characteristics of the model.

- A new instance generation scheme is proposed to test the performance of the model.

- We utilize Tabu Search (TS) and Constraint Programming (CP) approaches, the latter of which performs better in terms of solution time and quality.

The rest of the thesis is organized as follows. In Chapter 1, the main operations per-

formed at container terminals are explained along with the definitions of basic concepts. In Chapter 2, the related literature review is given. Chapter 3 provides the detailed problem description along with the MILP formulation. Chapter 4 describes applied solution strategies. Numerical experiments and computational results are presented in Chapter 5. In this chapter, data generation is also explained. Chapter 6 gives the conclusion and the future research directions.

1.1 An Overview of Container Terminals

In this section, the basic concepts related with container terminals are defined and the main decisions to be made at container terminals are explained. An overview of a container terminal can be seen in Figure 1.6. Stahlbock and Voβ (2008) provides a comprehensive survey of the operations at container terminals.

1.1.1 Container

A container (Figure 1.1) is a large reusable box which is used in intermodal transportation to carry the goods between destinations. The containers are defined according to their lengths. A standard containers is either 20-feet or 40-feet. A 20-feet equivalent container is referred to as 1 TEU (twenty feet equivalent unit). 40- feet containers are called as 2 TEU. It is important to have such standardization in containers since they are used interchangeably by both carriers and companies. Containers are flexible in the sense that they can carry a wide variety of goods such as liquids, manufactured goods, cars and perishable goods. Containerization enables effective transfer between several modes of transport, which is a major requirement for intermodal transportation.Containers are made of durable materials which help to keep the content of the containers safe and stable. Another advantage of the endurance of the containers is that they can easily be stacked. Since the space on the vessels and on the land side of the terminals are limited, the stackability of containers helps to manage the storage capacities better.

Containers are placed on the vessels according to a stowage plan. Each container is attributed with a specific bay, row and tier number. Bays are the cross-wise areas of a

Figure 1.1: A view of a container

vessel. Row is the position of the containers in bays while tier is the level on which the container is stacked. These concepts are visualized in Figure 1.2. The stowage plan of a vessel induces precedence relations among containers. There are some physical constraints which should be taken into account while planning the loading and unloading operations. The stowage plan is modified as the vessel proceeds between its destination ports with the addition and removal of containers.

1.1.2 Quay Crane

Quay cranes (QCs) (Figure 1.3) are large equipment that are used to load and unload the intermodal containers. They are very expensive and one of the main investments of container terminals. Moreover, QCs are the only equipment to perform the loading and unloading operations. Therefore, it is important to manage the QCs efficiently. Their working schedules directly affect the total waiting time of the vessels and the containers, which is the performance of the terminal. That is why there is an intensive research discussing the quay

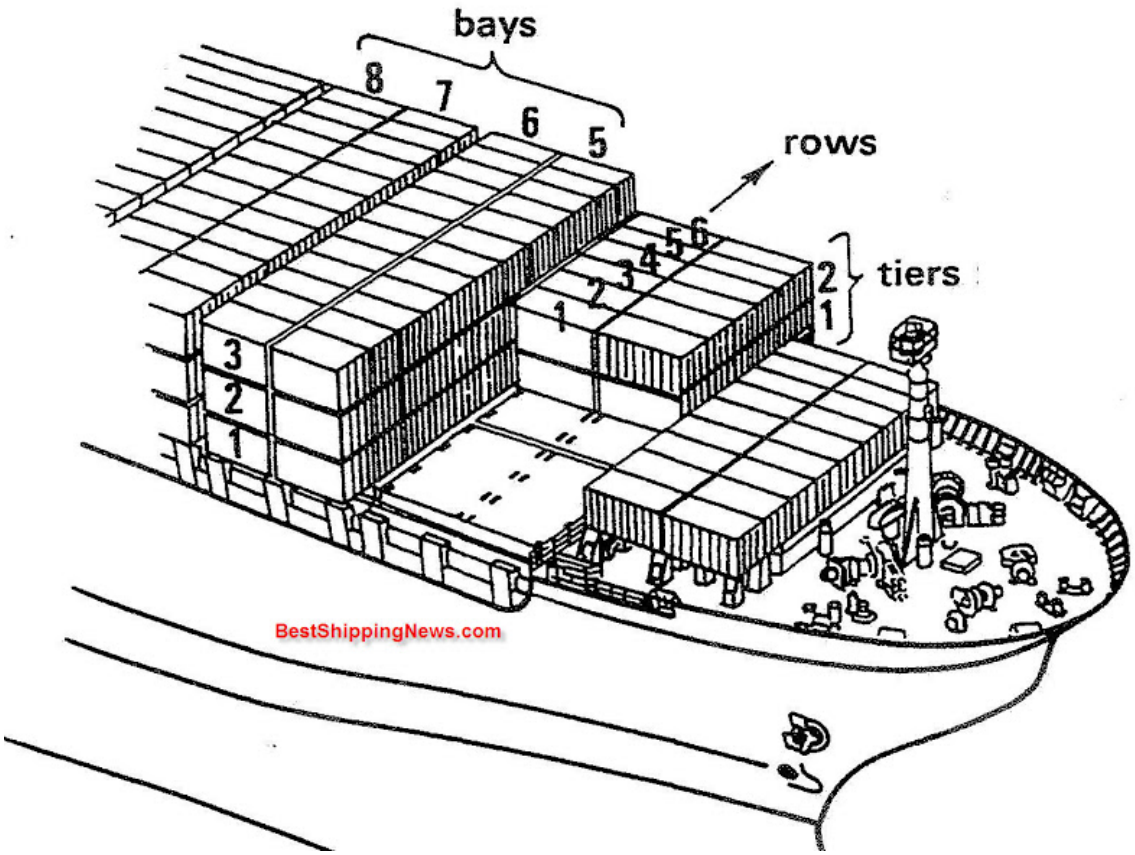

Figure 1.2: A view of container positions on a vessel

crane scheduling (Bierwirth and Meisel (2010)).

1.1.3 Other Handling Equipment

Different types of material handling equipment are utilized in terminals while transferring the containers. Once the containers are unloaded from vessels, they either go to other transshipment modes or to the storage areas via trucks that travel between the vessels and the related destination. The automated version of the trucks are called Automated Guided Vehicles (AGVs) (Figure 1.4). The containers can also be moved via a Straddle Carrier (SC) (Figure 1.5). SCs can both carry the containers and store them in stacks. Stacks are storage places where the containers can be kept for a certain amount of time. A view of stacks can be seen in Figure 1.7. They consist of lanes in which containers are placed on

Figure 1.3: A view of a quay crane

top of each other.

1.1.4 Container Terminals

Since the first containerization in the 1950s, the overseas world trade has been increasing. The expanding number of cargo carried in containers necessitated investments on the special facilities which can effectively handle these containers. Maritime container terminals (Figure 1.6) are large facilities where container flow is executed between two interfaces; the waterside and the landside. The waterside operations include the loading and the unloading of the vessels by QCs. At the landside, the containers are carried from the vessels to the storage areas or to other means of inland transportation. The containers arriving at port on vessels

Figure 1.4: An automated guided vehicle

are called import containers. They need to be unloaded and then directed to other mode of inland transport like truck or train to reach their final destinations. Conversely, the containers that are subject to precarriage by trucks or trains to arrive at the port are called export containers. They are loaded on the vessels and proceed to their final destinations. There is another type of container, which we consider in this thesis; i.e. transshipment or transit containers. Transshipment containers arrive at a port on vessels and go to their final destinations via other vessels. The terminals that can handle this type of containers are called transshipment container terminals and they are relatively larger and busier than the other traditional container terminals.

The importance of container terminals has been improving as a result of globalisation and economic growth. Most of the cargo is carried via sea freight because it is cost effective and the containers have many advantages as stated in Section 1.1.1. The containerization has an increasing trend which directly raises the workload of the container terminals and necessitates larger and fully equipped container terminals that can handle more TEUs. As a result of this development, the container terminals need to be better utilized in order

to be able to compete with each other. The main performance measure of container terminals is the turnover rate of containers. Therefore, finding ways to reduce the time that the vessels spend at port is an objective of any container terminal and contributes to its competitiveness.

1.2 Main Operations at Container Terminals

The main operations at container terminals can be classified into two: the waterside and the landside operations.

1.2.1 Waterside Operations

The operations at the quayside include solving Berth Allocation Problem (BAP), Quay Crane Allocation Problem (QCAP) and Quay Crane Scheduling Problem (QCSP). In this subsection, each of these problems are explained.

Berth Allocation Problem

Vessels need to be assigned to berths to be served at container terminals. The terminal management knows the schedule of the large cargo vessels approximately one year in advance. However, it is not easy to make berthing plans so far in advance because there can be some ad-hoc or short notice vessels calling at the port. In this case, the terminal management needs to revise the berthing plans and come up with another decision. There are some important criteria while making this decision like the length of the vessel and the depth of the berth positions. By taking all the constraints into account, a berthing plan should be developed. The objectives of BAP can be minimizing the vessel handling times or minimizing the total distance between the vessels. Bierwirth and Meisel (2010) provides a comprehensive survey on BAP.

Quay Crane Assignment Problem

After a vessel is allocated a berth at the quay, the containers are moved by QCs. For this purpose, a set of QCs are assigned to the vessel, which is referred to as Quay Crane

Assignment Problem (QCAP). The number of QCs to be assigned to a vessel depends on the workload and the priority of the vessel. Moreover, the physical characteristics of the quay and the QCs like the size and accessibility play a role on the QC assignment. QCAP is solved by Daganzo (1989) for small problem instances.

Quay Crane Scheduling Problem

Quay Crane Scheduling Problem (QCSP) aims to come up with a loading and unloading plan for the containers on a single vessel while the objective is, most of the time, to minimize the makespan; i.e. the completion time of the last container. Lim et al. (2007) proved that QCSP with the objective of minimizing the makespan is NP-Hard. In order to get more insight into the QCSP, which is the most related problem with the one in the thesis, it is required to understand how the containers are treated in different problem settings. Firstly, if all of the containers on a bay are regarded as a single task, then this problem is called as QCSP with complete bays. The number of tasks is the same as the number of bays and a task is to load or unload all of the containers on a bay. Secondly, if the containers located on a bay are divided into groups and each group is treated separately, then this problem is QCSP with container groups. Here, the containers are grouped according to a common characteristic like destination. These two classes of problems are described in detail in Bierwirth and Meisel (2010).

1.2.2 Landside Operations

The operations continue on the landside once the containers are unloaded from the vessel. Conversely, if there are export containers to be loaded on the vessels, the landside is the starting point of the operations. In this subsection, the main operations at the landside of the ports are explained.

Landside Transport Operations

In order to enable the continuation of the process at container terminals, the containers need to be transported between the ship and the stack. After the containers are taken off the ship by QCs, they are located on the material handling vehicles to be carried to the storage places. These vehicles should be managed carefully in order not to face any congestion in the

yard. The vehicles can be automated like AGVs; therefore, a proper planning synchronized with QC schedules is required. The more number of transport vehicles on the landside does not necessarily reduce the vessel handling times because the congestions at the cranes and in the yard tend to increase. Also, investing more in such vehicles adds up to the total operational cost of the terminal and sea freight might lose its economical advantages. The transport operations in container terminals are discussed in detail by Carlo et al. (2014a).

Landside Storage Operations

In container terminals, the space is limited. Containers need to be organized in a smart way to facilitate the exchange between different transportation modes. Containers are kept in stacks temporarily before they are canalized to other transportation means. Similar to their stowage positions on the vessels, they are given a block, bay, row and tier. A view of containers in the stacks can be seen in Figure 1.7. Carlo et al. (2014b) give an overview of the storage operations in container terminals.

Since the land is a scarce source for the terminals, it is important to optimize the stacking operations. According to the dispatching schedules of containers to and from the stacks, suitable positions in the stack should be assigned to containers such that reshuffling jobs are minimized. Reshuffling occurs when accessing the containers that are located below others. The upper ones have to be moved to other storage locations first and then repositioned again as explained in Vis and de Koster (2003)). The organization of the blocks in the stacks is related to the attribute of the containers. The containers sharing the same destination are usually stacked close to each other. There might also be some special requirements of reefer or refrigerated containers, which implies storage in some designated areas. The characteristics and the capacity of the material handling equipment like Straddle Carriers (SCs) affect the stacking structure. The height of the stack is determined by the employed SCs. The yard storage phase adds more time to the overall vessel handling time, which should be kept at minimum to be more competitive.

Figure 1.5: A straddle carrier

Figure 1.6: An overview of a container terminal

Figure 1.7: A view of container stacks

Chapter 2

LITERATURE SURVEY

In this section, the two most related problems with the problem in this thesis are explained in detail. They are Quay Crane Scheduling Problem and Transshipment Problems. Moreover, the state of the art on the solution methodologies, TS and CP, are also examined and the recent findings are explained.

2.1 Quay Crane Scheduling Problems

It is crucial to understand the main operations performed in container terminals to get more insight into the problem discussed in this paper. Stahlbock and Voβ (2008) classify and explain the parts of the container terminal systems as handling equipment, human resources and assisting systems. Later, the authors discuss the optimization problems encountered both on the seaside and the landside. Lastly, the research on integrated systems like simulation based ones is explained. There is much research done on scheduling the QCs on a single vessel. Bierwirth and Meisel (2010) provide a survey of QCSP and berth allocation in container terminals. They first handle the planning of seaside operations and explain the Berth Allocation Problem (BAP). According to the characteristics of vessel such as expected time of arrival, estimated processing time and physical requirements, the problem is to find a berthing position and time in order to fulfill a specific objective such as minimization of makespan. The authors then explain the Quay Crane Assignment Problem (QCAP). After a berthing plan is assigned to a vessel, the QCs to load and unload container should be identified. As for Quay Crane Scheduling Problem (QCSP), the aim is to find a processing schedule for QCs working on a vessel by obeying precedence relations and completing the operations without preemption while minimizing the makespan. This problem is stated to be similar to minimum makespan scheduling problem with identical parallel machines and precedence relations in Pinedo (2002). The problem is represented as $Pm|prec|Cmax$ in three field notation and is known to be NP-hard as proven by Blazewicz et al. (1983).

The general QCSP is divided into two classes based on the definition of a task, that is defined on the basis of bays or containers. When a task shows one bay and each bay is served by one QC, the problem is then called as QCSP with complete bays. On the other hand, if a task represents a group of containers which usually have common characteristics, then this problem is referred to as QCSP with container groups. In QCSP with complete bays, each bay is treated separately by one QC, whose schedules are unidirectional, meaning that QCs always move in the same direction according to Bierwirth and Meisel (2009). For this type of the problem when the move type is unidirectional, Lim et al. (2007) shows that there is always an optimal solution. The authors show that a minimum completion time can always be found when the search is limited to crane allocations. They developed optimization algorithms to solve the problem. For large problems, a simulated annealing algorithm is proposed to determine crane and time allocations. In the simulated annealing metaheuristic, the solutions are represented as crane allocations and an initial solution is found by assigning the earliest available QC to the first position. The algorithm is found to perform well when the number of QCs and containers are of realistic sizes.

QCSP with container groups, which was introduced by Kim and Park (2004), can have more than one QC assigned to a bay. Therefore, additional crane crossing interferences emerge, which makes the problem harder. Sammarra et al. (2007) proposes a Tabu Search algorithm to handle QCSP with container groups. The authors decomposed the problem into two: a routing problem and a scheduling problem. The routing problem is solved via TS metaheuristic while the scheduling problem is handled with local search procedures. The routing problem determines the sequence of tasks on each QC. The scheduling problem attributes a starting time for each container on the sequence found by routing problem. The authors utilized TS for the routing subproblem. The neighborhood structures are obtained by swap or insertion moves, which are used in this thesis as well and the details will be explained later in Section 2.3. The findings of the paper were promising, which directed us to consider TS as a solution method for our problem.

In Meisel and Bierwirth (2011), a unified approach for evaluating the performance of solution methods of different models is presented. The authors first explain that 3 different problem classes emerge with different characteristics; QCSP with container groups, QCSP with complete bays and QCSP with bay areas. The first two of these problems were ex-

plained in the previous paragraphs. The latter problem, QCSP with bay areas, requires the vessel to be partitioned into areas of bays such that the cranes operate separately on these areas. After such classification of the problem, the authors detected a need for a basis to compare the quality of the solutions of different quay crane scheduling models. They created test data to compare the performances of different procedures tackling QCSP. They described this new instance generation scheme for QCSP with container groups in detail. The data set includes values for number of container groups on a vessel, number of bays of a vessel, capacity per bay, number of QCs, crane travel time, crane safety margin, handling rate, precedence density, density of non-simultaneity. The processing time of container groups are extracted uniformly from the handling volume of bays. The precedence relations are created randomly for each bay of a vessel with a uniform density across the bays. The data generation scheme can be adapted to create instances for QCSP with complete bays and QCSP with bay areas as well. The authors define seven instance sets to be representative of typical QCSP. These instances are reproducible via a tool called QCSPgen. The parameters used in each set are described in detail as well as benchmark solutions. After having a common platform for all of the three problem settings, the authors explain the across model evaluation between the QCSP with container groups, QCSP with complete bays and QCSP with bay areas. The findings of this paper is having a unified approach for evaluation of different problems. No single modeling is found to be strictly dominant on the others.

A recent study from Fu et al. (2014) deals with multi vessel quay crane assignment and scheduling. They approach these two problems simultaneously by considering vessel priority and safety margins of QCs. The authors develop an integer programming formulation with the objective of minimizing the makespan of each vessel. The authors assumed that QCs can move from one vessel to another when tasks are completed and their working rates are identical. They propose a Genetic Algorithm (GA) to solve this integrated problem for large instances. A comparison is made between the exact solutions from GAMS and the GA results. The findings show that GA can report solutions in a reasonable time where the exact model fails to reach any solution.

Figure 2.1: The quay cranes working on a vessel

2.2 Transshipment Problems

The transshipment is the change in the mode of transport during the travel. The cargo can be taken to an intermediate location to be consolidated with other cargo or to be transferred via another transport vehicle. Transshipment takes place in large hubs. The transshipment of containers at a container terminal means that the containers are transferred to another vessel to reach their destinations after a temporary or no yard storage.

For this transshipment modality, the problems about the storage and cross docking terminology seem related (Boysen et al. (2010)). In some of these problems, transshipment containers are considered as well as import and export ones. Nishimura et al. (2009) address the storage arrangement of transshipment containers on a container yard of a large terminal. They discuss the movement of containers from mega-containership to feeder ships using temporary yard storage. An optimization model is set to formulate the container flow and

a lagrangian relaxation based on a heuristic approach is developed to solve the problem. Another study examining the transshipment flows between mother vessels and feeders is from Lee and Jin (2013). The authors proposed a mixed integer programming model and a heuristic solution algorithm. The direct transshipment of containers is mostly studied for the transportation modes like trucks and trains. The direct ship-to-ship transfer has not been studied widely. Alicke (2002) investigates intermodal terminal concept and analyses the advantages of transshipment of containers when different modes of transport is used. C. Lian and and Gen (2011) consider the transshipment transport problem in a container terminal where a mega-ship is used. They address the berth allocation planning problem in which one-way direct transshipment between a mother vessel and a feeder is considered.

The most related paper with our study is from Monaco and Sammarra (2012). They introduce a direct ship-to-ship container transshipment problem at a maritime terminal. In addition to the classical ship-yard-ship cycle, the authors introduce a direct movement of containers between vessels. In their study, they consider a situation of two vessels. In addition to the containers which must be loaded and unloaded using traditional methods, some of the containers on these vessels must be directly transshipped from one to another. The vessels are attributed with an arrival time, a priority, a set of loading and unloading operations, a set of QCs, a set of bays and available places for transshipment containers. There are also some precedence and non-simultaneity constraints. In order to enable the transshipment of containers, the two vessels need to be berthed at the same time or their berthing schedules should at least overlap. The authors propose a mixed integer linear model for this problem with the aim of determining the working schedules of all the QCs working on these two vessels and assigning a stowage position to all the containers directly transshipped. The objective is to minimize the service time of both vessels while reducing the temporary storage of directly transshipped containers as much as possible.

2.3 Tabu Search

One of the solution methodologies utilized in this thesis is Tabu Search, proposed by Glover and Laguna (1997), since it is widely applied as a solution method for the scheduling problems. Therefore, it is helpful to mention some of the related literature where TS is proposed.

Wong and Kozan (2010) investigated container operations in multi-ship environment and proposed List Scheduling and TS algorithms. They enriched their study with a real case at Port of Brisbane. The proposed heuristic algorithms produced good results and were promising for the future research. Another study for the scheduling optimization in container terminals is from Zeng and Yang (2007). The authors developed an optimization model for scheduling of QCs and yard cranes. They utilized TS metaheuristic to tackle this problem and observed that TS was efficient in searching the solution space and finding improved solutions for the terminal operations.

A TS algorithm for QCSP is proposed by Sammarra et al. (2007). The authors divided the problem into two; a routing problem and a scheduling problem. For the routing part, they developed a TS heuristic while the scheduling problem is solved by a local search technique. The neighborhood functions utilized in the study are swap and insertion. The swap move changes the positions of adjacent jobs by taking the precedence constraints into account. The insertion move allows to move the containers to the QCs apart from the ones they are initially assigned to. This insertion move is found to improve the makespan. The search continues from one solution to another according to the neighborhood definition. If a solution is removed from the neighborhood, then it is regarded as tabu and cannot be visited again after a predetermined number of iterations. The only way to accept a previously declared tabu solution before the iteration counter reaches a specified value is that this solution satisfies the aspiration criteria. The aspiration condition is satisfied if a move yields the best solution found so far. The number of iterations a solution will be kept forbidden is referred to as short term memory. The short term memory prevents the search to go to the previously visited solutions. The authors keep track of the number of times an attribute is accepted as a solution, which is called frequency. This frequency represent the long term memory which helps to diversify the search. The diversification in TS is important since the unexplored regions may contain promising solutions. The initial solution for the algorithm is driven by two different methods. One is found by assigning each crane the same number of tasks. The second is obtained by assigning the same work load to each QC. The initial solutions are significant in metaheuristic since they directly affect the performance of the search. In order to decrease the effect of the initial solution, a search should be properly diversified. The authors compare the performance of their TS

algorithm with the results reported in Moccia et al. (2006).The TS algorithm is found to perform well in terms of solution time and quality.

2.4 Constraint Programming

Another solution technique that we used the tackle the problem proposed in this thesis is Constraint Programming (CP), which is a well-known method for combinatorial optimization problems. The logic behind CP is to represent the problem via constraints and finding the sets that satisfy all of these constraints. CP resembles Constraint Satisfaction Problem (CSP) with the addition of an objective function as stated in Lustig and Puget (2001). The constraints in CP are various like linear, nonlinear, boolean, finite or integer. The decision variables are of two different types; interval and sequence. Laborie and Rogerie (2008) and Laborie et al. (2009) define interval variable as the time interval in the planning horizon in which an activity is performed. The interval variables have a start time, an execution duration, a presence information and an end time. The authors also explain the sequence variable as a permutation of a set of interval variables.The sequence variables can be used to state specific requirements on the interval variables more effectively.

The problems in container terminals are described and classified in Rashidi and Tsang (2013). The authors formulated CSPs for different scheduling problems faced at terminals and found it to be more efficient for Integer Programming formulations for specific cases. They considered BAP, QCAP and QCSP simultaneously because the total berthing time of a vessel depends on the number of QCs assigned to it.When the number of QCs assigned to a vessel increases, total handling time of the vessel decreases. The authors give a compact formulation for these problems. A CP approach for QCSP is proposed by Ünsal and Oğuz (2013). They formulated QCSP with container groups via CP using propositional logic. They used global constraints to assign tasks to QCs without any interference. The authors found out that the results have advantages over other solution methods. The computation time was reduced significantly and effective QC schedules were found in shorter solution time. This study reveals a fruitful area where research can be intensified and influences the direction of the studies in this thesis.

Chapter 3

PROBLEM DEFINITION

In our setting of transshipment container terminals, we deal with a case where many vessels can be moored at the same time and all of the containers on each vessel are a transshipment type. The containers will go to their destinations via other vessels, not by any other transportation method such as road or rail. Therefore, no import or export containers on the vessels are handled and no yard storage is necessary. A snapshot taken from the system can show a number of vessels berthed at their corresponding places. All of the vessels to be processed are assumed to be available at the time of use with known inbound contents; i.e. stowage plans. Each container on a specific vessel will be taken to another one. A container will not stay on the same vessel which had carried it to the terminal. Moreover, the exact locations of containers on their inbound and outbound vessels are known. Exact location means the corresponding vessel, the bay number on that vessel and the position on that bay. Since containers are located in stacks on bays, certain precedence exists between containers on the same bay. A container which is located on the top needs to be unloaded before the one located below can be unloaded. The same logic is applicable to the loading case. First, the containers that should be located at the bottom of the vessel are loaded so that the others can be loaded on top of them.

Upon arrival of the vessels to the port, they are assigned a number of QCs. The containers will be unloaded from their inbound vessels and loaded to their outbound vessels without preemption by these QCs. The QC-vessel assignment is kept constant through the planning horizon. In this study, to simplify the handling of the problem, the corresponding bays on each vessel where a specific QC will work are determined in advance. This assumption scales down the complexity of the problem. If QCs are not assigned to specific bays, it becomes more difficult to come up with a schedule for QCs because the domain of the possible assignments gets much larger. Therefore, there is no decision of assigning QCs to vessels and bays. Different from other studies in the literature, the loading and

unloading operations by QCs might be performed simultaneously; a QC can first unload a container and then load another container on one of the bays assigned to it. Suppose that QC_1 takes $container_A$ down the $vessel_X$. When $container_A$ is placed on the material handling equipment, it is observed that $container_B$ to be loaded on $vessel_X$ by QC_1 has arrived. $Container_B$ has already been unloaded from $vessel_Y$ and carried to $vessel_X$ by the time QC_1 reached the ground. If the space on $vessel_X$ for $container_B$ is available, then QC_1 can load $container_B$. By this way, QC_1 is better utilized since it does not travel empty from the ground to the vessel.

The containers will move from their inbound vessels once they are unloaded to their outbound vessels to be loaded by material handling trucks. We assume that these material handling trucks are unlimited in number and always available. This assumption is important since we do not consider any yard storage process. Normally, containers go to temporary yard storage areas after being unloaded or before being loaded. In our setting, we do not deal with the storage of the containers in the stacks. We treat the material handling trucks as if they can handle all of the containers and there is no need for temporary storage between the transshipment. In order for a container to be loaded to its target bay, all of the containers existing on that bay should first be unloaded. This imposes another relationship between containers whose inbound and outbound bays are the same. For example, suppose $container_A$ arrives at the port on bay_1 of $vessel_X$ and will leave the port on bay_2 of $vessel_Y$. Initially, $container_B$ is located on bay_2 of $vessel_Y$. In order for $container_A$ to be loaded on bay_2 of $vessel_Y$, $container_B$ should first be unloaded from bay_2 of $vessel_Y$.

Each container has a processing time; that is, the time required to unload and load that container. The processing time is assumed to be the same both for the loading and loading operation of a specific container. The containers also need some time to be transferred by material handling equipment from their origin vessels to destination vessels, which is the travel time. The QCs working on specified bays are assumed to need no time to move between bays; they instantaneously go from one bay to another to process containers so there is no repositioning time. Our objective for this problem is to come up with a loading and unloading schedule for each QC while minimizing the makespan; i.e the completion time of the latest loaded container.

3.1 Mathematical Model

In this section, a mixed integer linear programming (MILP) model is developed for transshipment container terminal problem. The problem described above has not been analyzed as it is; therefore, we came up with a new mathematical model. The modeling logic is similar to the study of Sammarra et al. (2007), which is based on Kim and Park (2004) and including the updates from Moccia et al. (2006). Sammarra et al. (2007) take the crane interferences into account in their model. Since we are given initial assignments of QCs to bays, we do not consider such interference.

3.1.1 Modeling Assumptions

The following assumptions are made while formulating the model.

- All of the containers on each vessel are transshipment type; there are no import or export containers.
- All of the vessels are available at the same time at the beginning of the planning period with known inbound and outbound content. Therefore, we know exactly which vessels and which bay a container comes from and where it will go.
- All of the QCs are identical.
- The QCs are assigned to specific bays of a vessel. We know which QC will load or unload the containers located on the bays of a vessel. Therefore, the cranes do not interfere with each other.
- The QCs move instantaneously between the bays assigned to them; there is no repositioning time for QCs.
- The material handling vehicles carrying containers between vessels are unlimited in number and always available.

These assumptions can be valid in very large transshipment hubs, which we accept as our setting. There can be both import and transit containers on a vessel as well as the export

ones to be loaded. We assume that we only deal with the transit containers on the vessels, that is why the other two types are taken out of the scope of this study. Additionally, the larger ports can handle many vessels and enormous number of containers at the same time, which makes it worth to deal with the transshipment containers separately. Having all the vessels available at time zero enables the aggregate planning of all loading and unloading operations. If this assumption is removed, the system would need to handle the dynamism and this will make the problem harder. All of the QCs are regarded as identical. Otherwise, we would need to differentiate their working capabilities and speed, which would affect the performance of the terminal. Also, known QC-bay assignment removes the necessity to consider QC interferences in the model, inclusion of which would complicate the model. As for the next assumption, the repositioning time for QCs are not considered. QCs are supposed to move instantaneously between the bays meaning that the repositioning time is so small with respect to the processing times. The last assumption is required since we do not consider any yard storage. The unloaded containers are supposed to be waiting on the material handling vehicles before being loaded. Otherwise, we would need to deal with the temporary storage in the terminal, which would again make the problem more complex.

3.1.2 Model Formulation

In the proposed MILP model, the objective is to determine a schedule which states the loading and unloading times of containers while minimizing the makespan. The following indices, sets, triplets and parameters are defined to represent the model.

Indices

i,j: Containers

k,l: Quay cranes

Parameters

p_i: Processing time of container i

t_i: Travel time of container i from its inbound vessel to its outbound vessel

Sets and Triplets

Q: Set of quay cranes

Ω_k: Set of all containers that quay crane k should move, $\forall k \in Q$

With the addition of dummy containers 0 and T with zero processing times in order to be able to represent the first and the last containers moved by the QCs;

Ω_k^0=$\Omega_k \cup \{0\}$

Ω_k^T=$\Omega_k \cup \{\mathrm{T}\}$

$\bar{\Omega}_k$=$\Omega_k \cup \{0,\mathrm{T}\}$

prec: The triplet of quay crane and containers showing which containers have precedence relations when moved by a specific quay crane. When quay crane k should move container i before container j, $(k,i,j) \in prec$.

order: The triplet of container and quay cranes showing the quay cranes by which each container should be unloaded and then loaded. When container i should first be unloaded by quay crane k and then should be loaded by quay crane l, $(i,k,l) \in order$.

Here, the triplet *prec* emerges from the stowage plan of the containers. If a container is located on top of another container, the QC unloading them will first move the one on the top. Conversely, the QC will first load the one that should be at the bottom of the stack while loading. This triplet has an element for each pair of such containers for both the loading and the unloading cases. The triplet *order* represents the inbound and outbound vessels of a container. Since we know which QC works on which vessel, we can directly show the origin and destination of a container by QCs. This triplet has an element for every container meaning that each container is addressed from its origin to its destination.

Decision Variables

A_{ik}: Start time of container i on quay crane k, $A_{ik} \geq 0, \forall i \in \Omega_k, \forall k \in Q$.

$$X_{ij}^k = \begin{cases} 1 & \text{if container } j \text{ is performed immediately after container } i \text{ by crane } k, \\ & \forall k \in Q, \forall i,j \in \Omega_k; \\ 0 & \text{otherwise.} \end{cases}$$

w: Makespan, $w \geq 0$.

Formulation

$$\min w \tag{3.1}$$

s.t.

$$w \geq A_{ik} + p_i \qquad \forall i \in \bar{\Omega}_k, \forall k \in Q \tag{3.2}$$

$$\sum_{j \in \Omega_k^T} X_{0j}^k = 1 \qquad \forall k \in Q \tag{3.3}$$

$$\sum_{j \in \Omega_k^0} X_{jT}^k = 1 \qquad \forall k \in Q \tag{3.4}$$

$$\sum_{j \in \Omega_k^T} X_{ij}^k = 1 \qquad \forall i \in \Omega_k, \forall k \in Q \tag{3.5}$$

$$X_{ij}^k + X_{ji}^k = 1 \qquad \forall i, j \in \Omega_k, \forall k \in Q \tag{3.6}$$

$$\sum_{j \in \Omega_k^0} X_{ji}^k - \sum_{j \in \Omega_k^T} X_{ij}^k = 0 \qquad \forall i \in \Omega_k, \forall k \in Q \tag{3.7}$$

$$A_{ik} + p_i \leq A_{jk} + M(1 - X_{ij}^k) \qquad \forall i, j \in \bar{\Omega}_k, \forall k \in Q \tag{3.8}$$

$$A_{ik} + p_i + t_i \leq A_{il} \qquad \forall (i, k, l) \in order \tag{3.9}$$

$$A_{ik} + p_i \leq A_{jk} \qquad \forall (k, i, j) \in prec \tag{3.10}$$

$$X_{ij}^k \in \{0, 1\} \qquad \forall i, j \in \bar{\Omega}_k, \forall k \in Q \tag{3.11}$$

$$A_{ik} \geq 0 \qquad \forall i \in \bar{\Omega}_k, \forall k \in Q \tag{3.12}$$

The objective (3.1) is to minimize the makespan. Minimizing the makespan means minimizing the vessel completion time, which is the common objective for most of the container terminals. Each container has a start time at the respective QCs which load them to their outbound vessels. When the time required to process a container is added to this start time, we have a completion time for each container. Then, the makespan value is the greatest completion time of all containers. Constraint set (3.2) calculates that makespan value. Constraint set (3.3) determines the first container on each QC. The last container on each QC is determined by constraint set (3.4). The processing order of containers in between are determined by constraint set (3.5). Constraint set (3.6) ensures that either $container_i$ is moved before $container_j$ or vice versa; only one of the containers can be preceding the other. Constraint set (3.7) states that each unloaded container is then loaded. Here, every container on each QC is evaluated. First sum in this constraint set shows the container

that precedes $container_i$ and the seconds sum determines the following. Constraint sets (3.3), (3.4), (3.5),(3.6) and (3.7) are the traditional routing constraints that are useful in scheduling settings. Constraint set (3.8) determines the start time of consecutive containers on each QC. Here, the big M notation is used to ensure that the start time of the containers is calculated when they are processed one after another; i.e. when the X_{ij}^k variables take the value 1. Constraint set (3.9) states that a container should be unloaded and carried to its outbound vessel before it can be loaded. Constraint set (3.10) ensures the precedence relations among containers. If QC k should perform container i before container j, they belong to a specific triplet *prec* and their start times are considered in this constraint set. The constraint sets (3.11) and (3.12) are the integrality and the nonnegativity constraints.

Chapter 4

SOLUTION APPROACHES

In this chapter, the solution approaches to tackle the problem are explained. First, Tabu Search (TS), which is known to be an advantageous meta-heuristics for scheduling problems, is applied to the problem. Having observed that there is room for improving the solution quality and the solution time of TS, Constraint Programming (CP) is tried as another solution methodology. The details of both procedures will be given in this chapter.

4.1 Tabu Search

Tabu Search (TS) is one of the mostly used metaheuristics for combinatorial optimization problems. It is first proposed by Glover (1986) and perceived to work well with scheduling problems. The basic idea behind TS is to guide the search process by using the history of the search. TS takes an initial feasible solution and tries to improve it in the next iterations. The neighborhood of a solution is the set of candidate solutions which are reachable by local transformations in the existing solution. TS evaluates all of the neighboring solutions of the current solution and the best solution in the neighborhood is chosen as the current solution. TS uses a type of memory, called *tabu list*, to escape from local optima and to avoid cycling. The recently visited solutions are kept in the tabu list and these solutions are banned to be admitted in the neighborhood of the new solution. By that way, tabu list helps to explore the areas that would be left unvisited by any other local search methods. The solutions in the tabu list are left out of the neighborhood for a certain number of iterations during the search. The number of solutions to be kept in the tabu list is called as *tabu tenure*. When a new element is added to the tabu list, one of the already existing solutions in this list is removed following a predetermined rule. The tabu status of a solution is overwritten when it provides an objective function value better than the best known so far. This cancellation of tabu status is called as aspiration condition. The algorithm terminates when a stopping condition is satisfied.

The main steps of TS can be summarized as follows:

TS Algorithm

Notation

s_0: the initial schedule,

s: the current schedule,

s^*: the best known schedule,

f_s: makespan of schedule s,

$N(s)$: the neighborhood of schedule s,

s^{iter}: the schedule $\in N(s)$.

Algorithm 1 TS algorithm

Require: p_i, t_i, Q, Ω_k, *prec*, *order*

1: Generate s_0.
2: $s \leftarrow s_0$.
3: Initialize $TabuList$
4: **while** termination criterion is not met **do**
5: Calculate f_s.
6: Construct $N(s)$.
7: Generate $s^{iter} \in N(s)$.
8: **if** $s^{iter} \notin TabuList$ **then**
9: $s \leftarrow s^{iter}$.
10: Update the $TabuList$.
11: **else**
12: **if** $f_{s^{iter}} \leq f_s$ **then** ▷ aspiration condition is met
13: $s \leftarrow s^{iter}$.
14: **end if**
15: **end if**
16: **end while**

4.2 Proposed TS Algorithm

We applied TS to solve the problem defined in this thesis. The reason why we prefer TS is its success in several scheduling problems, as can be seen in Sammarra et al. (2007) and Cesaret et al. (2012). TS is applied with different neighborhood structures and various additional characteristics. The details of these elements are described in the following subsections.

4.2.1 Solution Representation

A solution to the problem is represented by a N_{QC} X N_{max_c} matrix where N_{QC} shows the number of QCs in the problem and N_{max_c} is the maximum number of containers a QC needs to process. The i^{th} row of the matrix shows the processing order of the i^{th} QC. The j^{th} column in the i^{th} row, i.e. the $(i,j)^{th}$ element of the matrix, indicates the $(i,j)^{th}$ container's sequence on i^{th} QC. As an example, a solution to the problem where there are3 QCs and 3 containers and the 1^{st} QC must process only container 1, the 2^{nd} QC must handle containers 2 and 3, the 3^{rd} QC must process all of the containers can be represented as $\begin{bmatrix} 1 & 0 & 0 \\ 2 & 3 & 0 \\ 1 & 2 & 3 \end{bmatrix}$. Here, we understand that the 2^{nd} QC first processes the 2^{nd} container and then the 3^{rd} container. According to the initial positions of the containers on the vessels, we can infer whether the 2^{nd} QC loads or unloads these containers. The zeros in the rows state that there are no more containers for this QC to process. This representation keeps the loading and unloading information of each QC at the same time. We do not need to consider these operations separately.

4.2.2 Initial Solution

An initial solution is created by randomly assigning a processing order of containers to each QC. This random order may not always be feasible. When a container to be loaded by a specific QC is assigned an order before unloading the one that is already existing in its place, we face infeasibility. In order to avoid this issue, we perform a feasibility check to this initial random solution. The sequence for each QC is checked to detect if there is a container that violates the precedence requirements. If so, it is moved to the nearest place where it does not cause any infeasibility. This process continues until we obtain an initial feasible solution.

4.2.3 Move Operators and Neighborhood Definition

Two different move operators are tested in this thesis, they are swap and insertion. These operators are known to be successful in creating efficient neighborhoods for meta heuristic approaches as mentioned in Sammarra et al. (2007) and Cesaret et al. (2012). First, the

swap move randomly selects a QC. This selection simply means choosing a row from the solution matrix. After randomly determining the row; i.e. the QC, two random elements are swapped. Here, in order to be able to apply the swap operator, a QC must process more than one container. The chosen elements correspond to the containers whose processing positions will be swapped. The final solution after swap is checked for feasibility to see if any precedence relations are violated.

The insertion operator has a similar procedure with swap. A QC is randomly selected, then a randomly selected container is inserted to a randomly selected position on this QC. After insertion, a feasibility check is again performed and if any violations exist, they are fixed following the precedence relationships.

4.2.4 Tabu List and Tabu Tenure

Tabu list is a special characteristic of TS which helps us avoid moving in similar regions with the help of its memory. Tabu list keeps the record of a predetermined number of last visited solutions and prevents cycling. This predetermined number of solutions to be kept in the tabu list is referred as tabu tenure. In our study, preliminary tests are conducted to determine the tabu tenure and they will be presented in next sections. In the tabu list, the complete solutions are kept in order not to visit those solutions again. It is a good implementation of TS to keep some attributes of solutions in the tabu list instead of the complete solutions. Our setting requires determining the schedules of several QCs at the same time. Some containers can appear together on different QCs meaning that some moves may affect more than one QC's schedules. If we keep a part of the solution in tabu list instead of the whole solution, there is a risk of loss of information and unvisited solutions can obtain tabu status.

4.2.5 Aspiration and Termination Criteria

For the aspiration criteria, the widely used one is selected. If a solution in the tabu list gives an objective function value better than that of the current best solution, its tabu status is canceled. As for the termination criteria, the TS algorithm stops after either a fixed number of iterations or a predetermined number of solutions are visited without improvement.

4.2.6 Additional Procedures in Tabu Search

In order to enhance the performance of TS, some additional characteristics are utilized. Frequency based memory is added to the algorithm to detect promising areas for the search. It keeps the record of how many times a solution is visited, then after a number of iterations, the search is started with the most frequently visited solution to better exploit this area. This type of memory is applied with the neighborhood structures obtained by both swap and insertion operations.

In our study, we also analyzed the effects of allowing infeasible solutions through the search. For a certain number of iterations, the feasibility check is not performed and the algorithm is iterated with infeasible solutions. When the iteration counter reaches a certain value, the solution on hand is converted into a feasible solution by checking the precedence rules.

4.3 Constraint Programming

Constraint Programming (CP) approach has been widely and successfully used for scheduling problems. The problem is represented by using constraints in this method. The values for variables that are violating these constraints are removed from the model, which leads to reduced domains of variables, i.e. search space. New constraints are also inferred from the existing ones and the search continues. According to (Lustig and Puget (2001)), CP resembles a computer program that explains a method for solving a particular problem.

Constraint Satisfaction Problems (CSP) are an important class of combinatorial optimization problems and they help to better understand the CP logic. CSP is a feasibility problem over a set of domains for the variables, and a set of constraints over these variables. The aim is to find a set of values for the variables from their domains such that all of the constraints are satisfied. In CP concept, the problem also involves an objective function. CP can find the optimal solution by first checking the feasibility of the current solution and then adding a new constraint to the model stating that next solutions need to be better than this current solution.

There are several ways to represent real world problems in CP. The logical constraints can be easily embedded into CP, which makes it flexible with different problem types.

Moreover, the type of variables and constraints enable the declarative modeling of different settings in a compact way. The definitions of these characteristics are given in the following subsections.

4.3.1 Variable Types in Constraint Programming

The problem in this thesis requires assigning a time window for each container on the respective QCs that will unload and load it. That means containers are activities and the QCs are the resources. Each container is attributed with a start time on its resources, i.e. QCs. Each QC can handle only one container at a given time. This kind of activity-resource relation infers disjunctive scheduling problem. In disjunctive scheduling, the operations must be processed without interruption on scarce resources as described in Dorndorf et al. (2000). The disjunctive modeling considers two general constraints: precedence and disjunctive restrictions. The precedence relations ensures a specified processing order for the activities. The disjunctive relations prevents the overlapping assignments of activities to the resources. In CP context, it is very practical to represent these relations using an *interval variable*.

An *interval variable* shows the interval of time in which an activity is processed. Each activity should be assigned to an interval in which it will be processed without preemption. Each interval variable has a start time, a processing time, and an end time. If all of the activities need to be processed and none of them should be left unprocessed; then, this requirement is ensured by declaring the *presence* of these activities to $True$ or 1. We utilized an *interval variable* both for the loading and the unloading operations in this study. An *interval variable* is defined for each container that represents the time interval during which it is loaded by the respective QC. Similarly, another *interval variable* shows the interval for each container in which it is unloaded by the respective QC. The *interval variables* helped to model our problem in an effective way without defining any other variables.

4.3.2 Constraint Types in Constraint Programming

In some problems, there are several constraints valid for most of the variables that can easily be identified. These constraints can be stated via a conjunction of logical relations in order to have stronger statements in the model. Such high level constraints are referred to as *global constraints* in CP context. A good use of *global constraints* can enhance the

performance of the model. *Global constraints* help to reduce the domain for the variables faster by effective filtering algorithms as explained in Bordeaux et al. (2006). They allow to represent the real world problems more easily. The constraints we utilized in this study are *presence*, *disjunctive* and *precedence*.

Presence constraint is a *Boolean* constraint which takes a value of 0 or 1. It states whether a certain interval, depicted via an *interval variable*, must be present or not. In our problem, *presence* constraints are used to ensure a loading and unloading time interval for each container.

Disjunctive constraint describes how to model disjunctive resources and activities in CP context. It ensures non-overlapping activities for *interval variables*. An activity on an *interval variable* should finish before another activity can start. In our modeling, we utilize *disjunctive* constraints to guarantee that the loading and the unloading windows for each container do not coincide.

Precedence constraint ensures the relation between the start and the end time of intervals between which a certain precedence relation exists. It states that a certain activity should be completed before the next activity can start. *Precedence* constraints help to reduce the domain of the variables; thus it becomes easier to reach at good solutions. In our context, *precedence* constraints are used to model the precedence relations emerging from the stowage plan of the containers on the vessels. For example, if $container_a$ is located on top of $container_b$, then while unloading, $container_a$ should be removed first. Such relationships are modeled with the help of *precedence* constraints.

4.3.3 Systematic Search and Backtracking

CP utilizes systematic search throughout its procedure. The possible assignments of values in their domains to the variables are evaluated systematically. It is for sure that a solution will be found or the problem will be proven to be insoluble. One of the most common systematic search algorithms is backtracking. Backtracking is an incremental method which completes a partial solution with the consistent values from the domain of the variables. The unassigned variables are repeatedly attributed with a value and then a complete solution is formed. In backtracking, the constraints are checked after each realization of a partial solution. If a partial solution violates any of the constraint, the most recently added variable

becomes subject to backtracking again. This property of backtracking enables CP algorithm to eliminate a subspace from the variable domains (Ginsberg (1993)).

The most commonly used search technique is Large Neighborhood Search (LNS) (Shaw (1998)). The main idea is to use a tree based backtracking search technique to evaluate the feasibility of the move in the search. The important points are the relaxation and the reoptimization parts. First, the solution on hand is relaxed and then reoptimized with the hope of reaching every part of the neighborhood. Shaw (2005) improves LNS by adapting the relaxation part to the scheduling problems. The CP solver ILOGs CP Optimizer 12.3, which we used in this thesis as well, uses the logic from Shaw (2005) plus a learning scheme, which was proposed by Laborie and Godard (2007). Learning is the key point in any algorithm since it helps to diversify the search in the efficient neighborhoods and improve the robustness of the algorithms.

4.3.4 Consistency Techniques and Constraint Propagation

The domains of the variables are reduced by filtering the constraints in CP. This filtering is performed by consistency techniques and the most common consistency algorithm is arc-consistency. In arc-consistency, the domain of the variables is investigated and the search space is pruned for the inconsistent areas as explained in Bessiere et al. (2005). An assignment is inconsistent if at least one constraint is violated. Suppose that there are two constraints on some variables and the third constraint can easily be extracted from the existing two constraints. Then, certainly the domain of the variables will be reduced after checking the consistencies with these three constraints. As a result, some inconsistent values are removed from the search by constraint filtering and the search continues on these reduced domains. Consistency techniques help to direct the search in a shorter way to the solution and improves the efficiency of the CP algorithm. Moreover, global constraints in CP help to filter the constraints and handle the search space more effectively. Therefore, it is recommended to use as many number of global constraints as possible in CP modeling.

Backtracking and consistency techniques alone are not efficient ways to solve constraint satisfaction problems (CSP). A method which combines these procedures to enhance the search is constraint propagation and it can completely solve the CSP. The integration of backtracking and consistency techniques produced more efficient constraint programming

algorithms.

4.4 Proposed CP Algorithm

We propose Constraint Programming approach to tackle the problem in this thesis. In our CP model, the QCs are regarded as resources and the containers to be loaded and unloaded by these QCs are considered as activities. An *interval variable* is defined for each of the loading and unloading operations to show the interval of time that these activities are executed. Since all of these operations must be performed without preemption, it is assured by forcing the *presence* of these operations to $True$ or 1 in the CP model.

4.4.1 CP Model Formulation

The corresponding CP model is given here.

Indices

i,j: Containers

k,l: Quay cranes

Parameters

p_i: Processing time of container i.

t_i: Travel time of container i from its inbound vessel to its outbound vessel.

Sets

Q: Set of quay cranes.

Ω: Set of containers.

prec: The triplet of quay crane and containers showing which containers have precedence relations when performed by a specific quay crane. When quay crane k should perform task i before task j, $(k,i,j) \in prec$.

PU: The set of unloading processes by quay cranes. When container i should be unloaded by quay crane k, $(i,k) \in PU$.

PL: The set of loading processes by quay cranes. When container i should be loaded by quay crane k, $(i,k) \in PL$.

$notPU$: The set indicating the quay cranes by which the containers will not be unloaded. When container i should not be unloaded by quay crane k, $(i,k) \in notPU$.

$notPL$: The set indicating the quay cranes by which the containers will not be loaded. When container i should not be loaded by quay crane k, $(i,k) \in notPL$.

Decision Variables

L_{ik}: An interval variable that represents the interval in which container i is loaded by quay crane k.

U_{ik}: An interval variable that represents the interval in which container i is unloaded by quay crane k.

The sizes of the interval variables L_{ik} and U_{ik} are the *processing time* of $container_i$, namely p_i. By being able to define sizes for interval variables, we do not need to again consider p_i values in the constraints of the model.

Formulation

$$\min \max_{i \in \Omega, k \in Q} (end(L_{ik})) \tag{4.1}$$

s.t.

$$Presence(L_{ik}) = 1 \qquad \forall (i,k) \in PL \tag{4.2}$$

$$Presence(L_{ik}) = 0 \qquad \forall (i,k) \in notPL \tag{4.3}$$

$$Presence(U_{ik}) = 1 \qquad \forall (i,k) \in PU \tag{4.4}$$

$$Presence(U_{ik}) = 0 \qquad \forall (i,k) \in notPU \tag{4.5}$$

$$Disjunctive((L_{ik}, U_{ik}) | \forall i \in \Omega) \qquad \forall k \in Q \tag{4.6}$$

$$Precedence(L_{ik}, L_{jk}) \qquad \forall (k,i,j) \in prec \tag{4.7}$$

$$Precedence(U_{ik}, U_{jk}) \qquad \forall (k,i,j) \in prec \tag{4.8}$$

$$Precedence(U_{ik}, L_{il}, t_i) \qquad \forall i \in \Omega, \forall (k,l) \in Q \tag{4.9}$$

The objective is to minimize the completion time of latest loaded container. $end(L_{ik})$ depicts the end of the *interval variable* L_{ik}, where L_{ik} shows the interval in which $container_i$

is loaded by QC_k. Since $container_i$ is attributed with a *processing time*, p_i, the end of interval L_{ik} is the start time of the loading operation plus the *processing time*. Constraint sets (4.2) and (4.3) ensure that the containers are loaded by the QCs assigned to them.The set PL clearly specifies which containers should be loaded by which QCs. The set $notPL$ is added to the model to ensure that the decision variables L_{ik} are forced to be $False$ or 0 when $container_i$ should not be loaded by QC_k. Constraint sets (4.4) and (4.5) guarantee the same logic for the unloading case. Constraint set (4.6) states that the loading and unloading activities of a task should not overlap. Here *disjunctive* constraints force the loading and the unloading intervals of a container to be assigned to different points in the planning horizon. Constraint sets (4.7) and (4.8) force the precedence relations between the containers while loading and unloading, respectively. Constraint set (4.9) ensures that a container is unloaded and carried to its outbound vessel before it can be loaded. Here $Precedence(U_{ik}, L_{il}, t_i)$ means that end of interval U_{ik} plus t_i time units is less than or equal to the start time of interval L_{il}.

Chapter 5

COMPUTATIONAL STUDIES

This chapter explains the computational experiments conducted to test the performance of the TS and CP algorithms. In Section 5.1, a new data generation scheme is proposed and described in detail. The parameter settings for the TS algorithm is given in Section 5.2. In Section 5.3, the computational results of both TS and CP algorithms are presented.

5.1 Data Generation

To the best of our knowledge, the problem proposed in this thesis has not been studied before. Therefore, we need to test the capability of the mathematical model given in Section 3.1 and the proposed solution methods via instances that would reflect different scenarios for the setting. For this purpose, we developed a new instance generation scheme for the transshipment container terminal problem. The existing instances in the literature do not truly reflect our problem setting since our problem is defined for the first time in this thesis.

First, to test the computational limits of our exact model, we generated instances with fixed number of vessels and containers from different sizes (fixed case). Then, to have an idea on the capability of the model to handle randomness, we produced instances with varying sizes (variable case). As for the fixed case, the number of vessels is 5, 7 and 10 while the number of containers in total for all vessels is 30, 50 and 80. The number of QCs assigned to each vessel is chosen uniformly between [1, 3]. Also, the number of bays a QC will serve on a vessel is determined uniformly between [1, 4]. The containers are assigned randomly on vessels. According to their positions on the vessels, a certain precedence relation is observed while loading and unloading the containers. Moreover, the travel time of the containers from their inbound vessels to their outbound vessels depends on their assignment to the vessels too. The processing times of the containers; i.e. the time required to load and unload a container, is drawn randomly between the interval [1, 20] and the processing time is the same for both loading and unloading case. For each combination of vessel and container

sizes, 5 instances are generated. An example instance for this fixed case is shown below:

Number of vessels: 7

Number of containers: 50

Number of QCs: 15

Processing times= [3 13 10 17 7 5 7 18 16 14 9 13 11 17 12 11 17 15 19 17 20 7 1 18 6 10 19 10 9 8 4 6 3 15 16 16 16 13 7 9 9 7 13 18 6 9 7 6 6 10]

Travel times= [4 6 14 4 8 5 2 5 7 11 12 6 10 6 3 6 2 6 5 4 8 8 4 11 4 5 3 5 9 13 15 6 10 6 14 11 2 3 10 2 11 6 3 2 13 14 6 14 8 5]

For the variable case, the intervals that the number of vessels and the number of containers are chosen from are seen in Table 5.1. The generation of other parameters like the number of QCs or processing times of the containers follows the same procedure as the fixed case explained above. An example instance for this variable case is shown below:

Number of vessels: 10

Number of containers: 27

Number of QCs: 10

Processing times= [7 8 12 8 18 3 14 7 18 8 12 12 2 7 9 7 15 15 4 6 19 9 1 10 5 17 7]

Travel times= [13 52 16 34 47 31 5 33 42 17 41 43 12 36 4 38 37 56 49 25 18 41 54 7 3 9 59]

Table 5.1: The distributions of the number of vessels and containers for the variable case

	Number of vessels (n_v)	Number of containers (n_c)
Instance Set 1	U[2,5]	U[n_v,15]
Instance Set 2	U[2,10]	U[n_v,30]
Instance Set 3	U[2,15]	U[n_v,40]
Instance Set 4	U[3,5]	U[2n_v,30]
Instance Set 5	U[3,7]	U[2n_v,50]

These instances are used for preliminary tests on the exact model and the proposed solution methodologies, whose results will be presented in the following sections. Later, the instance creation procedure is modified to reflect similar characteristics with the study given in Meisel and Bierwirth (2011). In this study, Meisel and Bierwirth give parameter settings for different sizes of the problem for only one vessel. We duplicate this scheme for more

than one vessel. We take the parameters which are used in our model as they are defined in Meisel and Bierwirth (2011). The parameters are defined as depicted in Table 5.2 in Meisel and Bierwirth (2011). For the rest of the parameters for which Meisel and Bierwirth (2011) do not describe a method, we follow our previously defined procedure. For example, Meisel and Bierwirth state the number of containers, number of bays and number of QCs on a vessel. They give the interval for the number of containers and state the corresponding bay and QC number required for these containers. We create a predefined number of vessels as described in their study and take this setting as if they reflect the inbound content of the vessels. For the case when the vessels leave from the terminal, i.e. the outbound content of the vessels, we randomly assign containers to other vessels on random bays and extract the corresponding precedence relations and the travel times as defined earlier.

Table 5.2: The parameters used for the generation of test instances in Meisel and Bierwirth (2011)

Number of containers (n_c)	Number of bays (n_b)	Number of QCs (n_{qc})
$[10, 15, ..., 40]$	10	2
$[45, 50, ..., 70]$	15	4
$[75, 80, ..., 100]$	20	6

5.2 Parameter Settings for TS Algorithm

We performed preliminary tests in order to decide on the neighborhood structure, the tabu tenure and the termination criterion.

Neighborhood Structure: As previously explained, we tried two different neighborhood structures; swap and insertion. In order to decide on the best structure, we compared maximum, average and minimum percentage deviations from the optimum solution if exists or from the upper bound after 3 hours run of MILP model. We analyzed the results over 10 instances of each set for variable size case, whose characteristics were described in Table 5.1. We used a special setting of our TS algorithm where tabu tenure is 5. The algorithm terminates when the number of iterations without improvement reaches 2000 or the total number of iterations is 5000. This termination condition will be explained in the following paragraphs. In Table 5.3, the comparison between the swap and insertion neighborhood

structures is given.

Table 5.3: Preliminary test results for neighborhood structure

	Percentage deviations from optimum or UB					
	Swap			Insertion		
Instance Set	Max	Avg.	Min	Max	Avg.	Min
1	4.1	0.7	0	4.1	0.7	0
2	33.3	16.7	0	45	14.3	0
3	41.9	14.9	0	46.7	16.3	0
4	12.8	1.4	0	12.8	2.1	0
5	6.2	0.8	0	3.5	0.3	0
Overall	45	6.9	0	46.7	6.7	0

As seen in Table 5.3, the average percentage deviation is slightly smaller in the insertion neighborhood structure. Using insertion neighborhood structure produces better maximum percentage deviations for instance sets 1, 2, 4 and 5. The worst maximum percentage deviation we get from swap neighborhood, which is 45%, is better than that of insertion neighborhood. However, since we decided to deal with the average performance of the algorithm, we opt for the insertion neighborhood procedure for further analysis.

Tabu Tenure: Tabu tenure is one of the most critical parameters of TS algorithm as it directly affects cycling and restriction. If tabu tenure is small, there is a risk of cycling; on the other hand, if it is too large, the search might be restricted too much. To detect the best tabu tenure value, we tested three values of tabu tenure, *namely* $3, 5, 7$, over 10 instances of variable size instance sets. The insertion neighborhood structure is used as decided in the previous parameter setting. The number of iterations without improvement is 2000 and termination criterion is 5000 iterations in order to be consistent in the parameter tuning. The average percentage deviations of TS results over these 10 replications can be seen in Table 5.4.

Table 5.4: Percentage deviations of preliminary test results for tabu tenure

Instance Set	Tabu Tenure	Max	Avg.	Min
1	3	12.8	4.3	0
	5	4.1	0.7	0
	7	10.9	3.4	0
2	3	33.3	14.8	0
	5	33.3	14.3	0
	7	41.3	17.4	0
3	3	60.1	20.1	0
	5	46.7	16.3	0
	7	60.1	19.6	0
4	3	20.1	20.2	0
	5	12.8	2.1	0
	7	20.9	2.9	0
5	3	3.5	0.4	0
	5	3.5	0.3	0
	7	3.5	0.3	0

According to the results in Table 5.4, the best average performance of the algorithm realizes when the tabu tenure is set to 5.

Termination Condition: We decided to stop the algorithm when the iteration counter reaches a specified number or a predetermined number of iterations without any improvement is observed. The iterations that cannot improve the existing solution are rejected by the algorithm. Therefore, the number of iterations without any improvement is called *max rejects*. As for the total number of iterations, we examined 5000 and 7000. After experiencing the convergence as in Figure 5.1, 5000 was chosen to be the total number of iterations. The average percentage deviations were similar in the both case; therefore, the computation time helped us to choose the value for this parameter. The same test is applied for fixing the number of non improving solutions. The results of these analysis can be seen in Table 5.5.

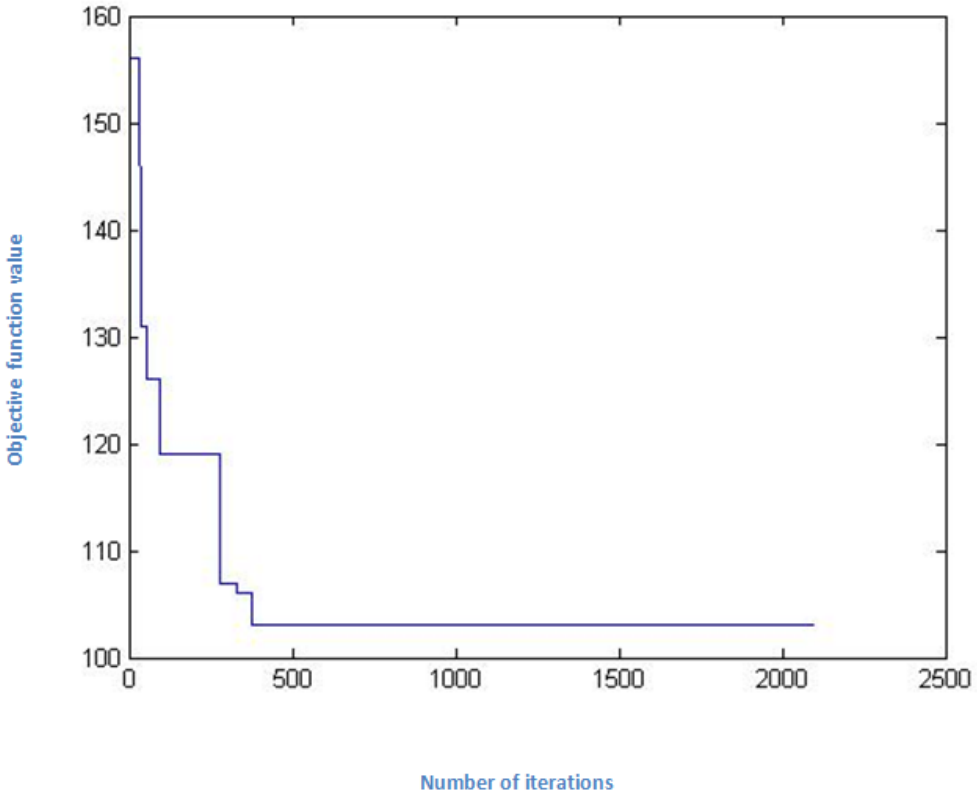

Figure 5.1: Convergence of the TS algorithm for an instance in Set 5

Table 5.5: Preliminary test results for termination condition

Instance Set	Max rejects	Max iterations	TS time (sec)	% deviations
1	2000	5000	10	0.7
	5000	7000	27	0.7
2	2000	5000	12	14.3
	5000	7000	35	14.3
3	2000	5000	25	16.3
	5000	7000	78	16.3
4	2000	5000	21	2.1
	5000	7000	56	1.5
5	2000	5000	35	0.3
	5000	7000	101	0.3

Additional Procedures: TS has some additional characteristics which might enhance the search. We wanted to see if the performance of TS can be improved via including frequency based memory and infeasible solutions in the algorithm. After having decided on the neighborhood structure as insertion, the tabu tenure as 5,the number of non improving solutions as 2000 and the total number of iterations as 5000, we performed tests to observe the behavior of our TS algorithm with these extra procedures.

Frequency based memory records how many times a solution is visited and helps TS to intensify the search. In our TS algorithm, the solutions that are frequently visited are periodically identified and the search is differentiated accordingly. When frequency based memory approach is applied, it is experienced that the maximum percentage deviation from optimum or upper bound decreased while average percentage deviation increased as seen in Table 5.6. Another way to diversify the search is to let infeasible solutions. For a predetermined number of iterations, the search is performed without any feasibility checks. Then, the solution on hand is converted into a feasible solution and the search continues from that feasible solution. When our TS algorithm is run by letting infeasible solutions, the average performance deviation is increased as seen in Table 5.6. After further analyzing the results in Table 5.6, it is seen that TS with infeasible solutions produces worse results

than TS with frequency based memory. Therefore, TS with infeasible solutions is excluded from further analysis. When pure TS is compared to TS with frequency based memory, the results indicate that the maximum percentage deviation in overall decreases with the inclusion of frequency based memory whereas the same effect cannot be seen on the average performance deviation. The average deviation is greater in TS with frequency based memory than the pure TS. Since we mentioned that we take the decisions on the characteristics of the model according to the average performance deviations, our final TS algorithm does not contain frequency based memory.

Table 5.6: Percentage deviations with additional procedures in Tabu Search

	Insertion			Insertion+Frequency			Insertion+Infeasibility		
Instance Set	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min
1	4.1	0.7	0	11.1	5.2	0	4.1	0.7	0
2	3.3	14.3	0	33.3	17.5	0	45	23.5	0
3	46.7	16.3	0	29.4	16.1	0	41.9	27.6	9.22
4	12.8	2.1	0	32.3	0.3	0	3.2	0.3	0
5	3.5	0.3	0	3.5	0.3	0	3.5	0.4	0
Overall	46.7	6.7	0	33.3	7.9	0	45	10.5	0

5.3 Results of the Computational Experiments

This section explains the results driven from the computational studies and gives an analysis on these results.

5.3.1 Computational Platform

The computational studies in this thesis are performed on a computer with an Intel(R) Core(TM) processor, 2.50 GHz speed and 4GB of RAM. The mathematical model as well as the CP algorithm proposed here are coded in CPLEX Studio IDE 12.5. The TS algorithm is run in MATLAB R2010b.

5.3.2 Computational Limits

We performed additional tests to detect the computational capacity of our model. As for the MILP model described in Section 3.1, the computational limit is examined through the fixed size instances and the variable size instances separately. These instances are run in CPLEX for 3 hours. If the optimal value is not found in 3 hours, then the solution found so far is recorded as the upper bound value to be compared later with the results of TS and CP. By keeping the number of vessels and containers as deterministic as in the fixed size instances, it is possible to identify the maximum problem size that the model can handle in a reasonable computational time. The number of optimal solutions found in 3 hours computational time is stated in Table 5.7.

Table 5.7: Number of optimal solutions out of 5 instances for the fixed size

Number of vessels	Number of containers	Number of optimal solutions
5	30	5
	50	4
	80	2
7	30	5
	50	3
	80	0
10	30	5
	50	5
	80	1

There are 5 instances for each combination of the values in the Table 5.7. The column "Number of optimal solutions" points out how many of these 5 instances are solved to the optimality. For example, we see on the fifth row of Table 5.7 that 3 instances out of 5 are solved to the optimality in the time limit. Out of 45 instances in total, the model finds the optimal solutions for 30 of them. For example, we see from the table that out of 5 instances which have 5 vessels and 50 containers, optimal solution is found for 4 of them in the computational time limit. For a deeper consideration of this result, we can say that when the average number of the containers on a vessel increases, it becomes hard to solve this problem.

As for the variable case, there are 10 instances of each size as shown in rows of Table 5.1. In Table 5.8, the number of optimal solutions found in each set of instances is presented.

Table 5.8: Number of optimal solutions out of 10 instances for the variable size

Instance Set	Number of optimal solutions
1	10
2	8
3	6
4	10
5	8

The number of optimal solutions is found to decrease when the number of vessels and containers increase, which is an expected case because the problem gets harder. Since the numbers are chosen uniformly between the specified intervals, there is a chance to get a higher number from the interval [2,10] than [2,5]. Therefore, Instance Set 2 might have more number of vessels and containers than Instance Set 1, thus Instance Set 2 is somehow more difficult and we have less number of optimal solutions found in computational time limit than that of Instance Set 1. Similar comparisons can also be made between other instance sets too. For example, the upper limit for number of containers is the same for Instance Set 2 and 4. However, the number of vessels comes from different limits in this case. The interval for Instance Set 2 is [2,10] while the one for Instance Set 4 is [3,5]. We see that Instance Set 2 may have some realizations that have more number of vessels than any instance in Instance Set 4. That probability of having a greater number of one parameter while others are the same makes one problem harder than the other. This explains why the number of optimal solutions found for Instance Set 2 is less than that of Instance Set 4.

5.3.3 Performance Measures

The MILP model is run for 1 hour for the instances which are created as explained in Meisel and Bierwirth (2011) to identify the upper bounds, the lower bounds and the gaps. These bounds are used to interpret the performance of the solution algorithms. The values for the UBs, the LBs and the corresponding gaps for each instance at the start and at the end of the computation period are shown separately in Table 5.9. In the same table, the percentage deviations of the TS results from the optimal values obtained from CP algorithm are also given. The instances for which no feasible solution can be found in 1 hour computational time limit are indicated as "-" in Table 5.9. The percentage gap cannot be calculated for

these instances. Table 5.9 includes instances where $n_v = 3$ and $n_c = 10$, $n_v = 5$ and $n_c = 10$, $n_v = 3$ and $n_c = 20$. MILP model fails to report any feasible solutions when $n_v = 3$ and $n_c = 20$, which indicates that no solution will be found for larger instances. That is why the comparisons in Table 5.9 are limited for these three set of instances.

Table 5.9: The performance comparison of the MILP model, CP and TS algorithms

Parameters					MILP						CP	TS	
					Root			Termination					
n_v	n_c	n_b	n_{qc}	Instance	UB	LB	% Gap	UB	LB	% Gap	Optimum	Z	% Gap
3	10	10	2	1	270	108	60	221	108	51.13	218	220	0.91
	10	10	2	2	291	105	63.92	210	105	50	210	210	0
	10	10	2	3	260	131	49.62	190	131	31.05	183	199	8.04
	10	10	2	4	298	81	72.82	202	81	54.46	202	207	2.42
	10	10	2	5	240	92	61.67	186	92	50.54	186	190	2.11
	10	10	2	6	289	151	47.75	215	151	29.77	200	204	1.96
	10	10	2	7	298	116	61.07	267	116	56.55	267	271	1.48
	10	10	2	8	296	103	65.20	211	103	51.18	211	231	8.66
	10	10	2	9	238	108	54.62	194	108	44.33	188	194	3.09
	10	10	2	10	235	119	49.36	184	119	35.33	182	191	4.71
5	10	10	2	1	-	84	-	-	88	-	153	192	20.31
	10	10	2	2	-	90	-	-	112	-	152	159	4.40
	10	10	2	3	-	89	-	-	89	-	195	251	22.31
	10	10	2	4	-	80	-	-	93	-	146	175	16.57
	10	10	2	5	-	73	-	-	73	-	137	162	15.43
	10	10	2	6	256	81	68.36	153	81	47.06	153	173	11.56
	10	10	2	7	209	69	66.99	139	69	50.36	139	164	15.24
	10	10	2	8	251	84	66.53	161	84	47.83	161	165	2.42
	10	10	2	9	-	90	-	-	90	-	157	180	12.78
	10	10	2	10	-	90	-	-	90	-	143	173	17.34
3	20	10	2	1	-	117	-	-	117	-	288	331	12.99
	20	10	2	2	-	104	-	-	104	-	319	319	0
	20	10	2	3	-	99	-	-	99	-	285	318	10.38
	20	10	2	4	-	127	-	-	127	-	263	338	22.19
	20	10	2	5	-	92	-	-	92	-	254	254	0
	20	10	2	6	-	83	-	-	83	-	250	273	8.42
	20	10	2	7	-	108	-	-	108	-	290	290	0
	20	10	2	8	-	109	-	-	109	-	295	295	0
	20	10	2	9	-	105	-	-	105	-	279	281	0.71
	20	10	2	10	-	115	-	-	115	-	309	309	0

Table 5.10: Run time comparison of MILP model, CP and TS algorithms

Parameters				Run times (sec)		
n_v	n_c	n_b	n_{qc}	MILP	CP	TS
3	10	10	2	-	5	26
	20	10	2	-	15	48
	30	10	2	-	30	75
	40	10	2	-	40	101
	50	15	4	-	118	126
	70	15	4	-	148	192
	90	20	6	-	568	247
	100	20	6	-	1241	275
5	10	10	2	-	6	42
	20	10	2	-	17	85
	30	10	2	-	20	127
	40	10	2	-	45	180
	50	15	4	-	1106	231
	70	15	4	-	3618	332
	90	20	6	-	-	720
	100	20	6	-	-	813

Table 5.11: Performance summary of the TS algorithm

Parameters				% deviations from optimum			Run times (sec)	
n_v	n_c	n_b	n_{qc}	Max	Ave.	Min	CP	TS
3	10	10	2	8.66	3.34	0	5	26
5	10	10	2	22.31	13.84	2.42	6	42
3	20	10	2	22.19	5.47	0	15	48
Overall				22.31	7.55	0		

The average run times of the MILP model, CP and TS algorithms for the instances which are created as explained in Meisel and Bierwirth (2011) are reported in Table 5.10. The run time is analyzed with more number of instances than the percentage deviation analysis. The "parameters" columns of Table 5.10 show the corresponding instance sizes. The "-" in the cells of this table shows that no solutions are found for these instances in the computational time limit.

Table 5.11 summarizes the performance of the TS algorithm in terms of percentage deviation and run time. Since we include percentage deviations in this summary, the instances in Table 5.9 are included here.

5.4 Analysis of the Results

In this section, the results presented in Tables 5.9, 5.10, and 5.11 are interpreted.

5.4.1 Run Time Analysis

We tackled the problem proposed in this thesis with MILP model, CP and TS methods. Therefore, we firstly compared the computational times of these algorithms. The average run times over 10 instances of the corresponding sizes are shown in Table 5.10. It is anticipated that MILP model could only solve small size instances since the problem described in this study tries to find schedules for several QCs, which is alone known to be NP-hard. Previously, the number of optimal solutions that MILP could find in 3 hours computational time was reported in Section 5.3.2 for the preliminary test instances. As for the instances used for performance evaluations (Meisel and Bierwirth (2011)), the MILP model runs for 1 hour and does not report an optimal solution for any of the instances. There are even instances for which MILP model cannot find a feasible solution within this computational time limit. For all of the ten instances whose parameters are given in the first row of Table 5.10, where $n_v = 3$ and $n_c = 10$, MILP model could find a feasible solution in 1 hour. MILP could also detect feasible solutions for three of the instances, where $n_v = 5$ and $n_c = 10$. However, no feasible solutions are reported for further instances within 1 hour.

It is observed from Table 5.10 that CP model finds the optimal solution in a very short time for the instances where the number of containers per vessel (n_c) is up to 40. After 40 containers per vessel, the run time of the CP model increases. For the instances where there are 3 vessels, the CP model can find the optimal solution for each of the instances. The run time of the algorithm is below 100 seconds when n_c is less than 50. It takes CP model longer to converge into a solution when n_c increases. As for the instances where the number of vessels (n_v) is 5, the same behavior is observed. The CP model cannot find the optimal solution where n_c is 90 and 100 in the computational time limit. Moreover, the run time increases dramatically for 50 and 70 containers. We can induce from Table 5.10

that the average run time of the CP model shows a very sharp rise when the total number of containers in the model is more than 250, as in the case where $n_v = 3$ and $n_c = 90$ or $n_v = 5$ and $n_c = 50$.

From Table 5.10, we see that the run time of TS algorithm is worse than CP for the instances where the total number of containers ($n_v \times n_c$) is less than 200. For the other instances, TS can converge to a solution quicker than CP. We can state that for the larger instances, TS algorithm stops before the half of the computational time of CP. It is important to point out that run time is not a sufficient performance measure since it is highly affected by coding skills and the computing platform. Therefore, the quality of the solutions obtained by the algorithms is analyzed in the following section.

5.4.2 Percentage Deviations Analysis

We first analyzed the performance of the MILP model through the upper and the lower bounds obtained in 1 hour computational time limit. The MILP columns in Table 5.9 show the bounds at the root and the termination of the algorithm. At the root, i.e. the start of the computation, the UB and the LB values are recorded. From these values, the percentage gap at root is calculated for each instance. Similarly, the UB and the LB at termination; i.e. the end of the computation, are used to calculate the percentage gap at termination. It is observed from Table 5.9 that the MILP model cannot find an optimal solution for any of the instances in 1 hour computational time limit. For the instances where $n_v = 3$ and $n_c = 10$, the MILP model can find a feasible solution, thus we have the corresponding UB values in Table 5.9. When $n_v = 5$ and $n_c = 10$, MILP model is able to find feasible solutions for 3 instances out of 10. For the rest of 7 instances, no feasible solutions are recorded in 1 hour, which are shown as "-" in this table. Additionally, no feasible solutions are reported by MILP for the instances where $n_v = 3$ and $n_c = 20$. It is also seen in Table 5.9 that the LB values for the instances where $n_v = 3$ and $n_c = 10$ stayed the same in the computational time limit. The average gap at root is realized as 58.60% whereas it decreased to 45.98% at termination. The LB values are improved for the 1^{st}, 2^{nd}, 3^{rd} and 4^{th} instances where $n_v = 5$ and $n_c = 10$. However, no feasible solutions are found for these 4 instances, which makes it impossible to calculate the gaps. As for the 6^{th}, 7^{th} and 8^{th} instances of the set where $n_v = 5$ and $n_c = 10$, feasible solutions are found and the corresponding gaps are

calculated. The MILP model cannot find a feasible solution for any of the instances where $n_v = 3$ and $n_c = 20$. Therefore, no gaps are shown for these instances. The LB values also stayed the same at root and termination, which implies that no improvement can be observed within 1 hour computational time for these instances.

The CP model is able to obtain optimal solutions for each instance in Table 5.9. The CP column in this table shows the corresponding optimum values. When we have a deeper look at the instances where $n_v = 3$ and $n_c = 10$, we can see that the MILP model can find the same solution with CP, which is in fact the optimal solution, for the 2^{nd}, 4^{th}, 5^{th}, 7^{th} and 8^{th} instances. However, the MILP model fails to prove the optimality of these solutions within the given time limit. The solutions at termination for these instances where $n_v = 3$ and $n_c = 10$ are on average 1.7% far from the optimal solutions. For the instances of $n_v = 5$ and $n_c = 10$ where a feasible solution is found (6^{th}, 7^{th} and 8^{th} instances), the feasible solutions are exactly the optimal solutions, but their optimality cannot be proven within the computational time limit. As for the case where $n_v = 3$ and $n_c = 20$, no instance can be compared with the optimal solutions obtained by CP since no feasible solutions can be found by MILP.

The performance of the TS algorithm can be seen in the last column of the Table 5.9. We compare the solutions obtained by TS with the optimum solutions from CP. It is observed that TS can detect the optimal solution for the 2^{nd} instance where $n_v = 3$ and $n_c = 10$. The average percentage deviation for these instances is 3.34% as summarized in Table 5.11. For the instances where $n_v = 5$ and $n_c = 10$, the TS algorithm cannot report any optimal solution. We also see from Table 5.11 that the worst maximum, average and minimum percentage deviations are observed when $n_v = 5$ and $n_c = 10$. TS algorithm can find the optimum solutions for 5 instances of the set where $n_v = 3$ and $n_c = 20$, which are depicted as 0% deviations in Table 5.9. Here, the maximum, the average and the minimum percentage deviations from the optimum values are shown. For the instances where $n_v = 3$ and $n_c = 10$, the performance deviation of TS algorithm is at its best. Since the problem gets harder with the increase in n_v or n_c, it is an expected outcome that the performance of the algorithm will worsen as well. It becomes more difficult for TS heuristic algorithm to handle the precedence relations and the feasibility checks as the problem size gets bigger. Therefore, the algorithm reports inferior solutions.

Needless to say that, CP algorithm performs better than TS since CP can detect the optimum solutions in a very short time for the instances considered in Table 5.9. The solution quality and the solution time of CP make it a more convenient method for the problem in this thesis.

It is observed that none of the instances in the analysis can be solved to the optimality by the exact MILP model. The reason is that our problem has some features which make the problem harder. When the characteristics of the problem are considered in detail, one potential reason is found to be the precedence relations emerging from the stowage plan of the inbound and the outbound vessels. The terminal management takes the precedence constraints of the inbound vessels as constant and start the planning from there. However, the stowage plan of the outgoing vessels could be modified if the containers have more than one candidate locations on the vessels. Sometimes, the containers are flexible and can be stacked on several bays of the vessels. This is because the characteristics of the containers are similar to each other, which makes it possible for them to be stacked interchangeably between the locations. In our problem, we assign the containers to the random bays of the outbound vessels and extract the corresponding precedence relations for the loading phase. We now want to observe if these precedence relations are so restrictive such that they prevent the model from being solved to the optimality. Therefore, the precedence constraints in the outbound vessels, which are seen in the loading stage of the vessels, are relaxed such that no strict stacking order of containers is valid. By relaxing the precedence constraints for the loading case, we somehow assume that the containers can be stored on any bay of the vessel without any restrictions and there is available storage space for the containers on the vessels. Each instance considered in Table 5.9 is modified such that the precedence relations observed while loading the containers to their outbound vessels are removed. The specific bay that a container needs to be loaded is the same; however, the location on that bay is not specified for this modified case. If two containers need to be loaded on the same bay, one of them can be located on top of the other; the loading order is not strictly constrained. The other parameters and the precedence relations are kept the same. We analyze the 1 hour run of the MILP model with the modified instances. Table 5.12 shows the results of this analysis.

Table 5.12: Performance comparison of the original and the modified instances

Parameters				Original instances		Modified instances	
				# of feasible solutions	Ave. % gap at termination	# of feasible solutions	Ave. % gap at termination
n_v	n_c	n_b	n_{qc}				
3	10	10	2	10	45.98	10	57.37
5	10	10	2	3	48.41	8	51.80
3	20	10	2	0	-	2	49.09

We compare the MILP results of the modified instances with the original ones. The MILP model cannot find any optimal solution for the modified instances also. The number of feasible solutions increases when we remove the precedence constraints for the loading case as seen in Table 5.12. Since the solution space becomes less restricted with the removal of precedence relations, it becomes easier for the model to detect a solution. The average gap at termination is recorded for these instances for which a feasible solution could be found as well. The average gap of the model without precedence relations in the loading phase, i.e. the modified instances, is slightly higher than that of original instance sets where $n_v = 3$ and $n_c = 10$ or $n_v = 5$ and $n_c = 10$. The reason of this increase is that there are more number of feasible solutions in the model and it becomes harder for the model to converge into one. The same comparisons cannot be made for $n_v = 3$ and $n_c = 20$ because no feasible solutions are found for the original instances of this size. After these comparisons of performance of two different settings, it can be stated that the difficulty of the problem does not only lie in the precedence constraints of the loading phase. When some of the precedence constraints are excluded from the model, it can still not detect the optimal solutions within the given computational time limit. The problem itself requires managing the unloading and the loading operations of several containers on several QCs at the same time. Although some of the precedence constraints are relaxed, the problem remains difficult. Therefore, we can conclude that stowage plans are not the only characteristics in the model which make the problem harder.

Chapter 6

CONCLUSIONS AND FUTURE RESEARCH

6.1 Conclusions

In this thesis, we consider a transshipment container terminal and analyze the operations need to be carried out on the vessels which moor at this type of container terminals. The aim of this study is to find a working schedule for the quay cranes simultaneously working on the vessels while minimizing the total completion time of all operations, which is makespan. To the best of our knowledge, the problem studied in this thesis has not been studied before. Therefore we propose a compact setting along with a new modeling, an instance generation scheme and solution methods for this problem.

We propose a mixed integer linear programming model for this complex problem faced at transshipment container terminals by assuming that all the containers on each vessel are a transshipment type. The mathematical model is tested in order to verify that it truly reflects the setting discussed in this study. For the test purposes and for the performance comparisons, a new instance generation scheme is developed. This instance generation logic can be utilized for the similar problems which contain transshipment modality. In order to have representative problem realizations, different test scenarios have been set and interpreted.

After analyzing the limits of the mathematical model, it is required to come up with more practical ways to tackle the problem. Therefore, we develop a Tabu Search metaheuristic algorithm, which is known with its successful performance for combinatorial optimization problems. We decide to use insertion neighborhood structure in TS rather than swap namely because the insertion neighborhood produced better average performance results. Both of the moves for neighborhood structure is performed randomly, enabling the diversification in the algorithm. Moreover, additional procedures are tried in order to see if they will positively affect the performance of the algorithm.

Another approach used as a solution procedure for the problem described here is Con-

straint Programming. The proposed CP model makes use of propagation and search technique and easily detects the values for the interval variables described in the model. The robustness of CP model is proven through the computations and the results emphasize the effectiveness of the algorithm. The success of CP approach lies in its capability to search the solution space after reducing the domains for the variables, which is also referred to as backtracking. By incrementally constructing the solution from the reduced domains, CP model can find consistent and complete solutions very efficiently. When we compare the performance of CP model with that of TS algorithm, we see that CP outperforms TS in both solution time and quality for the respective instances. Therefore, the proposed CP algorithm is a good candidate to tackle the real life problems.

6.2 Future Research

For future research directions, the enhancements on the mathematical model can be considered. The bounds obtained from the MILP model has room for improvement. A proper relaxation method can be applied to get stronger bounds on the model. Linear programming relaxation and Lagrangian relaxation methods should be further analyzed to see if they deserve future research. Furthermore, these improved lower bounds can be embedded into the solution algorithms to help them reach better solutions.

BIBLIOGRAPHY

K. Alicke. Modeling and optimization of the intermodal terminal mega hub. *OR Spectrum*, 24:1–17, 2002.

C. Bessiere, J. C. Regin, R. H. C. Yap, and Y. Zhang. An optimal coarse-grained arc consistency algorithm. *Artificial Intelligence*, 165(2):165–185, 2005.

C. Bierwirth and F. Meisel. A fast heuristic for quay crane scheduling with interference constraints. *Journal of Scheduling*, 10.1007/s:10951–009–0105–0, 2009.

C. Bierwirth and F. Meisel. A survey of berth allocation and quay crane scheduling problems in container terminals. *European Journal of Operational Research*, 202(3):615–627, 2010.

J. Blazewicz, J. K. Lenstra, and A. H. G. Rinnooy Kan. Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, 5:11–24, 1983.

L. Bordeaux, Y. Hamadi, and L. Zhang. Propositional satisfiability and constraint programming: A comparative survey. *ACM Computer Survey*, 38:1–54, 2006.

N. Boysen, M. Fliedner, and A. Scholl. Scheduling inbound and outbound trucks at cross docking terminals. *OR Spectrum*, 32:135–161, 2010.

H. Hwang C. Lian and and M. Gen. A berth allocation planning problem with direct transshipment consideration. *Journal of Intelligent Manufacturing*, 23:2207–2214, 2011.

H. J. Carlo, I. F. A. Vis, and K. J. Roodbergen. Transport operations in container terminals: Literature overview, trends, research directions and classification scheme. *European Journal of Operational Research*, 236:1–13, 2014a.

H. J. Carlo, I. F. A. Vis, and K. J. Roodbergen. Storage yard operations in container terminals: Literature overview, trends, and research directions. *European Journal of Operational Research*, 235:412–430, 2014b.

B. Cesaret, C. Oğuz, and F. S. Salman. A tabu search algorithm for order acceptance and scheduling. *Computers and Operations Research*, 39:1197–1205, 2012.

C. F. Daganzo. The crane scheduling problem. *Transportation Research-B*, 23B(3):159175, 1989.

U. Dorndorf, E. Pesch, and T. Phan-Huy. Constraint propagation techniques for the disjunctive scheduling problem. *Artificial Intelligence*, 122:189–240, 2000.

Y. Fu, A. Diabat, and I. Tsai. A multi-vessel quay crane assignment and scheduling problem: Formulation and heuristic solution approach. *Expert Systems with Applications*, 41:6959–6965, 2014.

M. L. Ginsberg. Dynamic backtracking. *Journal of Artificial Intelligence Research 1*, 1993.

F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13:533–543, 1986.

F. Glover and M. Laguna. Tabu search. *Kluwer Academic Publishers, London*, 1997.

K. H. Kim and Y. M. Park. A crane scheduling method for port container terminals. *European Journal of Operational Research*, 156(3):752–768, 2004.

P. Laborie and G. Godard. Self-adapting large neighborhood search: Application to single-mode scheduling problems. *Proceedings of the 3rd Multidisciplinary International Conference on Scheduling: Theory and Applications*, 2007.

P. Laborie and J. Rogerie. Reasoning with conditional time-intervals. *Proceedings of the 21st International FLAIRS Conference, Coconut Grove, Florida, USA*, 2008.

P. Laborie, J. Rogerie, P. Shaw, and P. Vilim. Reasoning with conditional time-intervals - part ii: an algebraical model for resources. *Proceedings of the 22nd International FLAIRS Conference, Sanibel Island, Florida, USA*, 2009.

D. H. Lee and J. G. Jin. Feeder vessel management at container transshipment terminals. *Transportation Research Part E*, 49:201216, 2013.

A. Lim, B. Rodrigues, and Z. Xul. A m-parallel crane scheduling problem with a non-crossing constraint. *Naval Research Logistics*, 54(2):115–127, 2007.

I. J. Lustig and J. F. Puget. Program does not equal program: constraint programming and its relationship to mathematical programming. *Interfaces*, 31(6):29–53, 2001.

F. Meisel and C. Bierwirth. A unified approach for the evaluation of quay crane scheduling models and algorithms. *Computers and Operations Research*, 38:683–693, 2011.

L. Moccia, J.-F. Cordeau, M. Gaudioso, and G. Laporte. A branch-and-cut algorithm for the quay crane scheduling problem in a container terminal. *Naval Research Logistics*, 53: 45–59, 2006.

M. F. Monaco and M. Sammarra. The direct ship-to-ship container transshipment problem at a maritime terminal. *The International Conference on Logistics and Maritime Systems, Bremen, Germany*, pages 79–89, 2012.

E. Nishimura, A. Imai, G. K. Janssens, and S. Papadimitriu. Container storage and transshipment marine terminals. *Transportation Research Part E*, 45:771–789, 2009.

M. Pinedo. Scheduling-theory algorithms and systems 2nd ed. *Prentice-Hall, Englewood Cliffs, NJ*, 2002.

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

M. Sammarra, J. F. Cordeau, G. Laporte, and M. F. Monaco. A tabu search heuristic for the quay crane scheduling problem. *Journal of Scheduling*, 10(4-5):327–336, 2007.

P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. *Lecture Notes in Computer Science*, 1520:417–431, 1998.

P. Shaw. Randomized large neighborhood search for cumulative scheduling. *ICAPS 2005*, 2005.

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

O. Ünsal and C. Oğuz. Constraint programming approach to quay crane scheduling problem. *Transportation Research Part E*, 59:108–122, 2013.

I.F.A. Vis and R. de Koster. Transshipment of containers at a container terminal: An overview. *European Journal of Operational Research*, 147(1):1–16, 2003.

A. Wong and E. Kozan. Optimization of container process at seaport terminals. *Journal of the Operational Research Society*, 61:658–665, 2010.

Q. C. Zeng and Z. Z. Yang. A two-phase tabu search approach to scheduling optimization in container terminals. *Journal of Marine Science and Application*, 6(2):44–50, 2007.

VITA

Duygu Korkmaz was born in Izmir, Turkey on March 28, 1988. She graduated from Karşıyaka Anatolian High School in 2006. She received her B.Sc. degree in Industrial Systems Engineering with double major in Economics from Izmir University of Economics, Izmir, Turkey in 2011. Same year, she became M.Sc. student in Industrial Engineering at Koç University. She is recently working as a Logistics Engineer in tobacco industry.